Universitat
de les Illes Balears

DOCTORAL THESIS

2022

NEW TOOLS TO PROTEIN INTERACTION NETWORKS ALIGNMENT AND PREDICTION

Adrià Alcalá Mena

Universitat
de les Illes Balears

DOCTORAL THESIS
2022

Doctoral Programme in Information and Communications Technology

NEW TOOLS TO PROTEIN INTERACTION NETWORKS ALIGNMENT AND PREDICTION

Adrià Alcalá Mena

Thesis Supervisor: Maria de la Mercè Llabrés Segura
Thesis Tutor: Maria de la Mercè Llabrés Segura

Doctor by the Universitat de les Illes Balears

A na Neus,
sense tu res seria possible.

Agraïments

Aquests agraïments no existirien si no fos per la incansable labor de na Mercè. Gràcies a ella i a la seva constant lluita per ajudar-me a no desistir i mantenir l'esperança teniu entre mans aquesta tesi. Així doncs no puc deixar d'agrair-li l'esforç que ha fet tots aquest anys que m'ha permès presentar aquesta treball.

Els començaments són sempre difícils, i més quan es tracta d'elaborar una investigació tan complexa. Això no obstant, he de donar les gràcies a n'en Ricardo per haver-me ajudat a fer l'arrancada d'aquest projecte i per la seva col·laboració comprovant que tot funcionava perfectament.

Estic enormement agraït a n'en Cesc, qui no només em va presentar el món de la bioinformàtica, sinó perquè a més em va donar l'oportunitat d'entrar-hi com a membre al grup d'investigació.

Agrair també a Gabriel Valiente per haver-me acollit a Barcelona i per la seva inestimable col·labaració a AligNet

Per continuar, voldria destacar en Biel Cardona, gràcies a ell i als seus comentaris, he estat capaç de créixer i ara el meu codi ara ja és més pythonic i no tan *gintonic*. En Biel Riera també és mereixedor d'unes paraules gràcies a la seva cooperació amb la implantació de l'eina de PinaWeb. Així com a tot el grup de BioCOM, per la seva ajuda i acollida.

Sense cap dubte, els membres de la Comissió de Doctorat, mereixen un lloc important en aquests agraïments. Oferint tota l'ajuda necessària i facilitant al màxim tots els trámits.

A na Patricia per la seva ajuda lingüística i perquè sense ella aquesta pàgina no hagués quedat tant bé.

I ara, fent referència a la part més personal, vull agrair als *gasolineros*, els meus amics, el seu afany a ajudar-me i obligar-me a desconnectar sempre que ho he necessitat. Gràcies a ells aquests anys han estat, no només plens de dates de lliurament i feines a fer, sinó que també hi ha hagut dies de riure i moments divertits per recordar.

Per últim, però no menys important a la família que sense la seva insistència i suport aquesta tesi no s'hagués acabat.

Contents

Abstract

All molecular functions and biological processes are carried out by groups of proteins that interact with each other. Proteins interactions are modeled by simple networks called protein-protein interaction networks (PPINs) whose nodes are proteins and whose edges are the protein-protein interactions. PPINs are broadly accepted to model the protein's functional relations, and their analysis has become a key ingredient in the study of protein functions.

Similar to the sequence alignment scenario, network alignment presents a comprehensive way to compare two or more biological networks. It considers not only biological interactions but also topological similarity of the neighborhood of biological nodes, and a most straightforward application of network alignment is to transfer known biological knowledge from well-studied species to unknown ones. With this purpose, several pairwise alignment algorithms have been proposed in the last years. Most PPINs aligners take into account both, the network topology and the biological features of the proteins, in the definition of "similarity." However, this is a trade-off that is hard to achieve. In addition, most aligners are quite complicated and time-consuming to use in practice, and many researchers are faced with selecting an aligner without being able to compare its performance with other aligners. Motivated by this lack of well-balanced and efficient algorithms, we have designed AligNet, a parameter-free PPINs aligner capable to achieve a good balance between topological and biological matching. Besides AligNet, and motivated by the lack of a "one-size-fits all" PPINs aligner, we have developed PINAWeb, a user-friendly web-based tool that obtains and compares the results produced by 5 different aligners.

Regarding the protein-protein interactions (PPIs) information, it must be obtained with high-throughput technology from metaproteomic data. Retrieving protein-protein interaction data experimentally is a very high time-consuming and labor-intensive task. Consequently, in the last years, the biological community has been looking for computational methods to correctly predict PPIs. The amount of new sequence data that is created every day, and its lack of *a priori* information, makes those methods that predict PPIs based on protein sequence information very popular. Under the requirement of predicting interactions based only on protein sequence information together with the idea to also exploit the PPIs information stored in the databases, we have designed Prots2Net, a PPIs predictor. To train the model, Prots2Net explores the PPIs retrieved from the STRING database of two selected species. The tests, reported on the Yeast and the Human datasets, show that Prots2Net performs better than the previous prediction methods.

Resum

Les proteïnes són les encarregades de dur a terme les funcions moleculars i els processos biològics de les cèl·lules, i per fer-ho, s'organitzen en grups que interactuen entre si. Aquestes interaccions es modelen amb grafs o xarxes anomenades xarxes d'interacció proteïna-proteïna (PPIN), on els nodes són les proteïnes i les arestes són les interaccions proteïna-proteïna (PPI). Les PPIN, s'han convertit en un model àmpliament acceptat per modelar les relacions funcionals de les proteïnes i la seva anàlisi és un ingredient clau en l'estudi de les seves funcions moleculars.

De manera similar al cas de l'alineament de seqüències, l'alineament de xarxes proporciona un mètode de comparació de dues o més xarxes biològiques. Però en aquest cas, es consideren tant les funcions biològiques dels nodes alineats com la seva similitud topològica, per la qual cosa una aplicació directa de l'alineament d'aquestes xarxes és transferir coneixements biològics d'espècies ja estudiades a altres espècies noves. En els darrers vint anys han sorgit diversos algorismes d'alineament de PPIN. La majoria d'aquests alineadors es basen en la definició d'una "similitud" entre els nodes a alinear on es considera tant la topologia de la xarxa com les característiques biològiques de les proteïnes. Ara bé, obtenir un bon balanç entre similitud topològica i similitud biològica és una tasca molt difícil que encara no s'ha assolit, i que a més, alguns alineadors deixen en mans dels mateixos usuaris en forma de parámetres a escollir. Això, juntament amb l'esforç considerable per a la seva utilització que requereixen alguns alineadors, fa que molts d'investigadors es vegin abocats a seleccionar un alineador sense saber si és el més convenient i sobretot sense poder comparar-ne els resultats amb altres alineadors. Per tot això, en aquesta tesi hem dissenyat un nou alineador, AligNet, sense paràmetres a escollir per part de l'usuari i capaç d'aconseguir un bon equilibri entre la coincidència topològica de les xarxes a alinear i la biològica dels nodes alineats. A més, hem desenvolupat PINAWeb, una eina web que proporciona, compara i analitza els alineaments obtinguts pels cinc alineadors més ben considerats ara com ara.

Finalment, pel que fa a les PPI, com que la seva obtenció de forma experimental és una tasca molt costosa, en els darrers anys s'ha optat pel disseny de mètodes computacionals que prediguin aquestes interaccions. Aquests predictors per ser d'utilitat, han de predir la interacció de les proteïnes en la mínima informació disponible d'aquestes, que normalment és la seqüència de les proteïnes. Sota el requisit de predir les interaccions a partir únicament de la informació de la seqüència de la proteïna juntament amb la idea d'explotar també la informació de les PPI disponible en les diferents bases de dades, hem dissenyat Prots2Net, un predictor de PPI. Aquest model de predicció utilitza les PPI de dues espècies conegudes obtingudes de la base de dades STRING i les seqüències de les proteïnes en el seu entrenament. Les proves duites a terme per avaluar Prots2Net que aquí es presenten, mostren que funciona millor que els mètodes de predicció coneguts.

Resumen

Las proteínas son las encargadas de llevar a cabo las funciones moleculares y los procesos biológicos de las células, y para hacerlo, se organizan en grupos que interactúan entre sí. Estas interacciones se modelan con grafos o redes llamadas redes de interacción proteína-proteína (PPIN), donde los nodos son las proteínas y las aristas son las interacciones proteína-proteína (PPI). Las PPIN, se han convertido en un modelo ampliamente aceptado para modelar las relaciones funcionales de las proteínas y su análisis es un ingrediente clave en el estudio de las funciones moleculares de las proteínas.

De manera similar al caso del alineamiento de secuencias, el alineamiento de redes proporciona un método de comparación de dos o más redes biológicas. Pero en este caso, se consideran tanto las funciones biológicas como la similitud topológica de los nodos alineados, por lo que una aplicación directa del alineamiento de estas redes es transferir conocimientos biológicos de especies ya estudiadas a otras especies nuevas. En los últimos 20 años han surgido varios algoritmos de alineamiento de PPIN. La mayoría de estos alineadores se basan en la definición de una "similitud" entre los nodos a alinear, donde se considera tanto la topología de la red como las características biológicas de los nodos. Ahora bien, obtener un buen balance entre similitud topológica y similitud biológica es una tarea difícil que no se ha logrado todavía, y que incluso algunos alineadores dejan en manos de los mismos usuarios. Además, los alineadores requieren un esfuerzo considerable por parte del usuario para su utilización, por lo que muchos investigadores se ven obligados a seleccionar un alineador sin saber si es el más conveniente y sobre todo sin poder comparar los resultados con otros alineadores. Por este motivo, en esta tesis hemos diseñado un nuevo alineador, AligNet, sin parámetros a escoger por parte del usuario y que consigue un buen equilibrio entre la coincidencia topológica de las redes a alinear y la biológica de los nodos alineados. Además, hemos desarrollado PINAWeb, una herramienta web que proporciona, compara y analiza los alineamientos obtenidos por los cinco alineadores mejor considerados por ahora.

En cuanto a las PPI, su obtención de forma experimental es una tarea muy costosa, y últimamente se opta por el diseño de métodos computacionales que predigan estas interacciones. Estos predictores, para ser útiles, deben predecir la interacción de las proteínas con la mínima información disponible de estas, que normalmente es la secuencia de las proteínas. Bajo el requisito de predecir las interacciones a partir únicamente de la información de la secuencia de la proteína, junto con la idea de explotar también la información de las PPI disponible en las diferentes bases de datos, en esta tesis hemos diseñado Prots2Net, un predictor de PPI. Este modelo de predicción utiliza, para su entrenamiento, las PPI de dos especies conocidas obtenidas de la base de datos STRING y las secuencias de las proteínas. Las pruebas realizadas, que aquí se presentan, muestran que Prots2Net obtiene mejores resultados que los métodos de predicción conocidos.

Introduction

The work developed in this thesis is framed within the field of bioinformatics, which is a branch of computer science born in the early 1970s. As Wikipedia says, bioinformatics is an interdisciplinary field that develops methods and software tools for understanding biological data, in particular when the data sets are large and complex. Until recently, biologists did not have access to very large amounts of data, which has now become commonplace.

Biology, as a scientific study of life, is an ancient branch of science that can be traced to Ancient Egypt and Mesopotamia. This life study was mainly based on environment observation and new discoveries were obtained with more detailed examinations. In the early 19th century, a number of biologists pointed out the central importance of the cell. The term molecular biology, the branch of biology that studies the cells and all living processes at the cell level, was already used in 1945, but it was in 1953 that two young men, named Francis Crick and James Watson, made a double helix model of deoxyribonucleic acid (DNA) which changed the whole research scenario. The research lead to finding DNA material in other microorganisms, plants, and animals. Since then, the central dogma of molecular biology describes the process in which DNA is transcribed into RNA, which is then translated into proteins. Molecular biology also plays a critical role in the understanding of structures, functions, and internal controls within individual cells, all of which can be used to efficiently target new drugs, diagnose disease, and better understanding cell physiology [33]. Some clinical research and medical therapies arising from molecular biology are covered under gene therapy, whereas the use of molecular biology or molecular cell biology in medicine is now referred to as molecular medicine. Computational biology involves the development and application of data-analytical and theoretical methods, mathematical modelling and computational simulation techniques to the study of biological, ecological, behavioral, and social systems [13]. Computational biology has been used to help sequence the human genome, create accurate models of the human brain, and assist in modeling biological systems [40]. The field is broadly defined and includes foundations in biology, applied mathematics, statistics, biochemistry, chemistry, biophysics, molecular biology, genetics, genomics, computer science, ecology, and evolution [23]. Bioinformatics has offered insights into all aspects of life on earth far beyond biomedicine. This has ranged from understanding how plants remove CO_2 from the atmosphere during photosynthesis to how microbes alter sewage wastes.

The mathematical analysis and computational modeling of complex biological systems are called systems biology. In these biological systems, proteins are the active agents that regulate most of the biological processes. Proteins play many vital roles in living organisms, from forming the structural fibers of muscles to the enzymes that digest food to synthesizing and replicating DNA. In addition, other kinds of proteins include an-

tibodies that protect an organism from infection, and hormones that send important signals throughout the body. These functions are performed by sets of proteins that interact with each other. Proteomics is the large-scale study of proteins [8, 11]. Proteomics enables the identification of ever-increasing numbers of proteins. This varies with time and the distinct requirements, or stresses, that a cell or organism undergoes [7]. Proteomics is an interdisciplinary domain that has benefited greatly from the genetic information of various genome projects, including the Human Genome Project [37]. It covers the exploration of proteomes from the overall level of protein composition, structure, and activity, and it is an important component of functional genomics.

One of the most difficult problems in systems biology is to understand how proteins interact with each other to carry out their biological functions. Protein–protein interactions (PPIs) are physical contacts of high specificity established between two or more protein molecules as a result of biochemical events steered by interactions that include electrostatic forces, hydrogen bonding, and the hydrophobic effect. Many are physical contacts with molecular associations between chains that occur in a cell or in a living organism in a specific biomolecular context. Proteins rarely act alone, as their functions tend to be regulated. Many molecular processes within a cell are carried out by molecular machines that are built from numerous protein components organized by their PPIs. These physiological interactions make up the so-called interactomics of the organism, while aberrant PPIs are the basis of multiple aggregation-related diseases, such as Creutzfeldt–Jakob and Alzheimer's diseases.

PPIs have been studied with many methods and from different perspectives: biochemistry, quantum chemistry, molecular dynamics, signal transduction, among others [95, 35]. Protein interactions are modeled as node relations in what is called protein-protein interaction networks, PPINs for short. PPINs are undirected graphs whose nodes are proteins and whose edges are protein-protein interactions. PPINs are broadly accepted to model the protein's functional relations, and their analysis has become a key ingredient in the study of protein functions and comparative proteomics. Indeed, in the post-genomic era, with more and more omics data being generated, networks become a more appropriate representation to describe complicated biological systems, such as PPINs, gene regulatory networks, and transcription factor networks. Comparative proteomics is the comparison of protein features of different organisms. As in the sequence alignment scenario, network alignment presents a comprehensive way to compare two or more biological networks in systems biology. It considers not only biological interactions but also topological similarity of the neighborhood of biological nodes, and a most straightforward application of network alignment is to transfer known biological knowledge from well-studied species to unknown ones. In particular, the alignment of PPINs has become a key ingredient to obtain functional orthologs as well as evolutionary conserved assembly pathways of protein complexes. The goal of the biological network alignment problem is to map nodes across different networks based on their biological (sequence) similarity and the interaction patterns of their neighboring communities (i.e. topology similarity). For this purpose, several pairwise alignment algorithms have been proposed in the last 20 years. A PPINs alignment consists of a function that injectively maps the nodes in one network (source network) to nodes in the other network (target network). That is, it establishes a one-to-one relation between the nodes in the source network and the nodes in the target network. Network alignment can be classified into the following categories: local and global alignments, pairwise or multiple alignments.

Local alignments only match a subset of nodes from the source network to the target network, while global alignments match every node in the source network to a node in the target network. This implies that, the source network is smaller (in number of nodes) than the target network. A pairwise alignment (local or global) finds a match between two networks, while multiple alignments look for conserved regions in multiple networks at the same time. The early aligners [45, 49, 54, 55, 69] were aimed at finding *local* alignments between regions with similar structure in the networks under comparison. But since local alignments between regions of the pair of PPINs could be mutually inconsistent, it could be impossible to merge the alignments between regions into an alignment of the whole network. In contrast, a *global* alignment algorithm is aimed at finding the best overall alignment between the whole PPINs [19]. Several such global PPINs aligners have been proposed during the last years [55, 1, 34, 71, 76, 90, 62, 79]. We refer to [59] for a recent review on PPINs alignment.

Most PPINs aligners are based on the idea that "two nodes are similar when their corresponding neighbors are so", taking into account both the network topology and the biological features of the proteins in the definition of "similarity." The problem is that attaining the right balance between network topology and biological information is one of the most difficult and key points in any PPIN alignment algorithm. As it is shown in [16, 63], when an alignment process is guided by topological information only, it produces alignments with a high topological coherence but a low biological coherence, while when it is guided by sequence information only, the resulting alignments have a high biological coherence but a low topological coherence. This becomes specially inconvenient in those aligners where the user has to choose the value of a parameter that specifies the desired balance between the topological and the sequence similarities. In addition, most aligners are not efficient from the computational point of view.

Several scores have been established in order to evaluate and assess the aligners. The correctness of an alignment is measured by the preservation of the network structure and the protein's functional information. See surveys [5, 16, 19, 60, 21] for a review of implemented aligners and their evaluation. However, it is very difficult to fulfill both requirements. Indeed, the authors in [16] conclude that:"*we find dramatic differences between existing algorithms in the quality of the alignments they produce. Additionally, we find that many of these tools are inconvenient to use in practice, and there remains a need for easy-to-use, cross-platform tools for performing network alignment"*. In addition, the authors in [60] state that: "*in the network alignment problem, unfortunately there is no gold standard for evaluation; that is, a best alignment is unknown from a biological perspective"*. Therefore, when the topological similarity between two networks wants to be captured rather than the biological function similarity of the matched proteins, an alignment tool with a high ratio of edges preserved by the alignment must be considered. On the other hand, when the function similarity of the matched proteins is more relevant than the edge preservation, a suitable alignment tool is an aligner providing a high functional coherence score.

Motivated by this lack of well-balanced and efficient algorithms, the first goal in this thesis was to design a PPINs aligner to fulfill these requirements. Thus, we designed AligNet, a parameter-free pairwise PPINs alignment algorithm aimed at filling the gap between efficient topologically and biologically meaningful matchings. The overall idea of the algorithm is to obtain many local alignments that are combined and extended into a meaningful global alignment. The final alignment captures the benefits of considering

both types of alignments: with the local alignments we capture the topological similarity between the networks, and we speed up the running time of the algorithm, while with the final global alignment we solve the inconsistencies among the local alignments and yield an overall alignment of the pair of input PPINs.

As more omics data is generated, new proteins are collected every day and their functional relations must be obtained with high-throughput technology. Retrieving protein-protein interaction data experimentally is a very high time-consuming and labor-intensive task. Consequently, in the last years, the biological community is looking for computational methods to correctly predict PPIs. Predicting new protein functions and protein-protein interactions is a relevant topic in computational biology [82, 91, 51, 42]. This topic has played a key role during the COVID-19 pandemic to find a vaccine for the coronavirus disease [31, 46, 58]. Indeed, understanding the mechanism of viral infections is a crucial step towards the discovery of antiviral drugs and vaccines. In this line of research, virus-host protein-protein interaction networks, a particular form of protein-protein interaction networks (PPINs), have become appropriate to analyze virus-host relationships. Specifically, information on well-known and studied virus-host protein-protein interaction networks can be transferred to new ones through protein-protein interaction network comparison and alignment. As mentioned above, the same rule applies in the general setting of protein-protein interaction networks, where biological knowledge of well-known and studied protein-protein interaction networks is transferred to new ones through a network alignment. A popular approach in this framework is the "network-based" approach [86]. With this approach, new interactions between proteins are understood as "missing links" in protein-protein interaction networks. New functional information is obtained either from the characteristics of the protein's position in the network, or it is transferred from annotated proteins sharing a similar position in the PPINs of other organisms. To transfer functional information between proteins from different organisms, it is necessary to relate the proteins sharing a similar position in the PPINs. This relation between proteins sharing a similar position is obtained by a pairwise PPINs alignment. Within this alignment setting, aligners with a high ratio of preserved edges match proteins that share a similar position in the networks. On the other hand, aligners with high functional coherence values match proteins sharing some biological function. Hence, the following question arises again: which is the best aligner one can use to achieve the prediction of new protein functions and protein-protein interactions via a PPINs alignment?

The current lack of a "one-size-fits all" PPINs aligner, together with the computational problems that arise when installing the relevant software, motivated the second goal of this thesis, which was to develop a user-friendly web-based tool to obtain and compare the results produced by different aligners. The resulting program, PINAWeb, is a web-based tool that performs a pairwise alignment of two PPINs, considering the aligners: AligNet [3], HubAlign [34], L-GRAAL [62], PINALOG [79] and SPINAL [1], which are the most recent and show the best performance as evaluated [16]. For every aligner and pair of networks, PINAWeb returns the corresponding alignment, the topological and biological performance scores, and also a visualization of the comparison of the alignments achieved (agreement/differences) when several aligners are considered. In addition, as running an aligner on its own data represents a major problem for most researchers, PINAWeb has been devised such that PPINs can be uploaded either from a standard database or from the researcher's own data. Finally, it is an open tool, and new aligners, new databases, and new metrics for scoring alignments can be integrated

posteriorly.

Regarding the PPI data which is the basis of PPINs, different methodologies to experimentally retrieve them have been developed in the last two decades. These methodologies are yeast two-hybrid screens [41, 48], protein chips [105], tandem affinity purificaton [80], immunoprecipitation [72] and spectrometric protein complex identification [36] among others. As a result, a considerable amount of PPI data has been generated and stored in different databases [93, 100] which allows the study and analysis of PPIs. Such analysis can be performed under different computational techniques, and inference statistics has been proved to be successful at PPI prediction. This fact, together with the acceptance that experimental methods to detect PPIs are too expensive in terms of time and labor, has generated an explosion of computational methods to predict PPIs. These computational methods rely on different machine learning technologies, such as deep neural networks [18], convolutional neural networks [96], rotation forest [97, 53, 74], support vector machine [102, 32, 104, 74], extreme learning machine with principal component analysis [103], k-nearest neighbors [101], and random forest [17, 74] among others, and different data types to train the models, such us literature mining knowledge [15], gene fusion [20], phylogenetic profiles [92], gene ontology annotations [61], gene neighborhood [28], and co-evolution analysis of interacting proteins [44]. However, the amount of new sequence data that is created every day and its lack of *a priori* information, makes those methods that predict PPIs based on protein sequence information very popular [38, 52, 53, 6]. Despite the simplicity of predicting interactions based only on protein sequence information, the results reported by those methods prove that, they perform well and achieve good measures of error prediction [53].

Under the requirement of predicting interactions based only on protein sequence information together with the idea to also exploit the PPIs information stored in the databases, the third goal of this thesis was to design a PPIs predictor, based on the protein sequences and two PPINs. Hence, we developed Prots2Net, a multilayer perceptron neural network that uses protein sequence information only from the input proteins and interaction information from the STRING database [93]. The training data of our model are two PPINs from the STRING database selected by the user. To build the prediction model, PPIs in one of the selected species are predicted through PPIs of orthologous proteins in the other species, and the PPIs information from the second species is used as the true output in the training. The tests carried out to evaluate the performance of Prots2Net compared to other 12 existing methods, reported values above 95% in accuracy, sensitivity, and precision error measures on the Yeast and Human datasets. Thus improving the results obtained by the others methods.

Overall, we present in this thesis AligNet, a new aligner of PPINs, PINAWeb, a comparator of aligners of PPINs, and Prots2Net, a PPIN predictor. All these tools contribute to gather and analyze protein-protein interaction data and move one step forward to protein's function discovery.

Organization of this thesis

This Introduction is followed by a chapter where we gather a series of definitions on graph theory and used algorithms, neural networks, binary classification problem and evaluation as well as a brief explanation of all aligners and prediction methods considered in this thesis. Then, in Chapter 2 we present AligNet, a PPIN aligner as well as the results

obtained when comparing AligNet with the already existing aligners. In Chapter 3, we present PINAWeb, a user-friendly web-based tool that obtains and compares the results produced by the following aligners: AligNet, HubAlign, L-GRAAL, PINALOG, and SPINAL.

Finally, in Chapter 4, we present Prots2Net, a tool designed to predict the PPIs of a proteome or a metaproteome sample. We also reported some tests on the Yeast and the Human datasets, that shows that Prots2Net performs better than the previous prediction methods.

The Thesis ends with a short Conclusion chapter and future work, where we summarize the results as well as the utility of the tools, together with a brief discussion about future lines of research.

Publications

The results reported in this memoir are supported by:

- AligNet was presented in the XIV Symposium on Bioinformatics (JBI2018) and in the 15th International Symposium on Bioinformatics Research and Applications (ISBRA 2019).
- AligNet was published in BMC Bioinformatics [3].
- PINAWeb was presented in the 22nd Annual International Conference on Research in Computational Molecular Biology (RECOMB 2018) and in the Hbioinfo bridge 2021.
- PINAWeb was published in bioRxiv [2] as a preprint and has been submitted to BMC Bioinformatics.
- Prots2Net was published in bioRxiv [4] as a preprint and has been submitted to the Special Issue of the International Journal of Molecular Sciences titled Advance in Computational Protein Structural Biology.

Acronyms

AC	Auto Covariance. 19
ACC	Accuracy. 9
ANNs	Artificial Neural Networks. 3
API	Application Programming Interface. 21
AUC	Area Under Curve. 9
BLAST	Basic Local Alignemt Search Tool. 10
cel	Caenorhabditis elegans. 41
CFC	Complex Functional Coherence. 12
CMTNNs	Complementary Neural Networks. 6
CNNs	Convolutional Neural Networks. 6, 16
DCT	Discrete Cosine Transform. 19
DEG	Database of Essential Genes. 14
DIP	Database of Interacting Proteins. 13
dme	Drosophila melanogaster. 41
EC	Edge Correctness ratio. 11
ELM	Extreme Learning Machine. 18
EMBL-EBI	European Molecular Biology Laboratory - European Bioinformatics Institute. 14
FC	Functional Coherence value. 11
FFNNs	FeedForward Neural Networks. 6
FN	False Negative. 8
FP	False Positive. 8
FPR	False Positive Rate. 8
GD	Gradient Descent. 5
GO	Gene Ontology annotation. 11
GRNNs	General Regression Neural Networks. 6
GUI	Graphical User Interface. 20
HAP	Hypergraph Assignment Problem. 3
hsa	Homo sapiens. 41

HTML	HyperText Markup Language. 51
HTTP	HyperText Transfer Protocol. 20
Isobase	Isobase. 14
KNNs	K-Nearest Neighbors. 18
LDA	Latent Dirichlet Allocation. 19
LPQ	Local Phase Quantization. 15
MCC	Matthews Correlation Coefficient. 9
MIPS-CORUM	MIPS CORUM. 14
MIPS-FunCat	MIPS Functional Catalogue. 14
MLP	Multilayer Perceptron. 6
MMI	Multivariate Mutual Information. 19
mus	Mus musculus. 41
NCBI	National Center for Biotechnology Information. 72
NMBAC	Normalize Moreu-Broto AutoCorrelation. 19
NNs	Neural Networks. 3
NP-hard	Non-deterministic Polynomial-time hardness. 3
OLPP	Orthogonal Locality Preserving Projection. 18
PCA	Principal Component Analysis. 18
PIR	Protein Information Resource. 14
PNNs	Probabilistic Neural Networks. 6
PPI	Protein-Protein Interaction. 10
PPIN	Protein-Protein Interaction Network. 10
PPV	Positive Predictive Value. 8
PseeAAC	Pseudo Amino Acid Composition. 19
PSI-BLAST	Position-Specific Iterated Basic Local Alignment Search Tool. 18
PSSM	Position-Specific Scoring Matrix. 18
RBFNNs	Radial Basis Function Neural Networks. 6
ReLU	Rectified Linear Unit. 4
RF	Random Forest. 19
RNNs	Recurrent Neural Networks. 6
ROC	Receiver Operating Characterisc. 9
RoF	Rotation Forest. 15
sce	Saccharomyces cerevisiae. 41
SGD	Stochastic Gradient Descent. 5
SIB	Swiss Institute of Bioinformatics. 14
SMR	Substitution Matrix Representation. 19

SNNs	Simulated Neural networks. 3
SRC	Sparse Representation based Classifier. 19
SVM	Support Vector Machine. 16
TLU	Threshold Logic Unit. 6
TN	True Negative. 8
TNR	True Negative Rate. 8
TP	True Positive. 8
TPR	True Positive Rate. 8
UniProt	Universal Protein Resource. 13
WSRC	Weighted Sparse Representation based Classifier. 19

CHAPTER 1

Preliminaries

We introduce here some basic notions and notations that will be used throughout this manuscript.

1.1 Graph theory

In mathematics, and more specifically in graph theory, a graph or a network is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related". The objects correspond to mathematical abstractions called nodes (also called vertices or points) and each of the related pairs of nodes is called an edge (also called link or line). Typically, a graph is depicted in diagrammatic form as a set of dots or circles for the nodes, joined by lines or curves for the edges. We denote a graph by $G = (V; E)$ where $V = V(G)$ is its set of nodes and $E = E(G)$ its set of edges.

Throughout this manuscript, we will use the notation listed below:

- $n = |V|$ is the *order* of the graph, and it is its number of nodes.
- $m = |E|$ is the *size* of the graph, and it is its number of edges.
- $e = \{u, v\} = uv$ is an edge that joins u and v.
- u and v are adjacent if there exists and edge e that joins u and v, we also say that u and v are neighbors.
- Given $u \in V(G)$, $N_G(u)$ is the set of nodes adjacent to u, i.e., $N_G(u) = \{v \in V \,|\{u, v\} \in E\}$.
- $deg(u) = |N_G(u)|$ is the degree of u, namely the number of its neighbors.
- A path from a node u to a node v is an ordered sequence of nodes, where each node is adjacent to the next one in the list, i.e. if (u_0, u_1, u_2, u_3) is a path between

u_0 and u_3 means that u_0 and u_1 are adjacent, u_1 and u_2 are adjacent and u_2 and u_3 are adjacent.

- u and v are connected if there is a path between u and v.
- We define the distance between two connected nodes $u, v \in V$, as the number of edges contained in the shortest path between u and v. We denote by $d(u, v)$ the distance between the nodes u,
- The diameter of a graph G is the maximum distance between two connected nodes, $D(G) = max_{u,v \in V} d(u, v)$.
- The eccentricity of a node is the maximum distance between these nodes and all the other connected nodes in the graph, $exc(u) = max_{v \in V} d(u, v)$.

There are many kinds of graphs in the literature. We define below those that are used in this work:

- **Null graph**, also called an empty graph, is a graph with no edges, i.e., $E = \emptyset$.
- **Connected graph**, is a graph in which every pair of nodes are connected.
- **Bipartite graph**, is a graph where its set of nodes V can be split into two subsets, V_1 and V_2 such that there is no edge between a pair of nodes in one subset, i.e., for every edge $uv \in E$, $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$.
- **Weighted graph**, is a graph where every edge has assigned a numerical weight.
- **Directed graph**, is a graph where the edges are directed edges or arrows, indicating that the relationship represented by the edge, only applies from a source node to a target node, but not the other way around. Notice that, in a directed graph, the edges are ordered pairs of nodes. Hence, we denote a directed edge by $e = (u, v) \in E$.

A *hypergraph* is a generalization of a graph where the edges link arbitrary sets of nodes. Formally, a hypergraph is a pair $H = (V, E)$, where V is a set of nodes and its set of hyperedges or hyperarcs is $E \subseteq \{\{AB\} \,|A, B \in \mathcal{P}(V)\}$ where $\mathcal{P}(V)$ denotes the power set of V, that is, the set of subsets of V. For instance, if $V = \{a, b, c, d, e, f\}$, then $e = \{\{a, b, c\}\{d, b, f\}\}$ is the hyperarc between the sets $\{a, b, c\}$ and $\{d, b, f\}$.

1.1.1 Graph clustering

Detecting graph elements with "similar" properties is of great importance, especially in large networks or graphs, where it is crucial to identify specific patterns or structures quickly. The process of grouping together elements/entities based on some similarity measure is called cluster analysis. In our graph setting, a clustering of a graph or network, is a partition of its set of nodes. Every element of this partition is called a *cluster* and every node belongs to one, and only one cluster. An *overlapping clustering* is a set of subsets of nodes (also called clusters) such that every node belongs to one cluster. In contrast to a clustering, in an overlapping clustering the clusters may share some nodes.

1.1.2 Algorithms on graphs

The algorithms and implemented tools on graphs that we use in this thesis are the following:

1. The Hungarian matching algorithm, also called the Kuhn-Munkres algorithm, is an $O(|V|^3)$ algorithm that can be used to find a maximum-weight matching in bipartite graphs. It is also called the assignment problem [50].

2. The Hypergraph Assignment Problem (HAP) generalizes the assignment problem from bipartite graphs to bipartite hypergraphs. The HAP is NP-hard, (i.e., an algorithm for solving it can be translated into one for solving any nondeterministic polynomial time problem) and APX-hard (i.e., a nondeterministic polynomial time problem that allows polynomial-time approximation algorithms with approximation ratio bounded by a constant) even for small hyperedge sizes and hypergraphs with a special partitioned structure [12].

1.2 Neural networks

Neural Networks (NNs) also known as Artificial Neural Networks (ANNs) or Simulated Neural networks (SNNs), are a subset of machine learning techniques located at the heart of deep learning algorithms. Their name and structure was inspired by the human brain, mimicking the way that biological neurons signal to one another. Thus, their nodes are called neurons.

Briefly, each neuron receives a multiplied version of inputs and random weights, which are added to a static bias value (unique to each neuron layer). This information is passed to an appropriate activation function, which decides the final value to be given out of the neuron. There are various activation functions available as per the nature of input values. Once the output is generated from the final neural net layer, a loss function (expected vs output) is calculated, and a back propagation is performed such that the weights are adjusted to minimize the loss function. Finding optimal values of weights is what the overall operation is focusing around. The concepts involved in neural networks are the following:

- *Weights* are numeric values which are multiplied with inputs. In back propagation, they are modified to reduce the loss. In simple words, weights are machine learned values from Neural Networks. They self-adjust depending on the difference between predicted outputs vs expected outputs.

- *Activation function* is a mathematical formula which helps the neuron to switch ON or OFF.

- *Input layer* is the first layer of the neural network and represents the input of the model.

- *Hidden layers* are the intermediary nodes, they take a set of weighted input values and produces an output value through an activation function.

- *Output layer* is the last layer of the neural network and represents the output of the model.

- *Loss function* is a function defined on a data point, prediction and label, that measures the penalty (difference between output and expected output).
- *Cost Function* is a function used to evaluate the performance of the neural network, it might be a sum of loss functions over the training set plus some model complexity penalty (regularization).
- *Optimization algorithm* is used to optimize the cost function.
- *Regularization* is used to address over-fitting and feature selection.

1.2.1 Architecture of a neural network

Before training a neural network, the following aspects must be decided:

- Number of hidden layers and their size (number of neurons).
- Activation function.
- Loss function.
- Cost function.
- Optimization algorithm.
- Regularization method.

Activation function

An activation function is a very important feature of an artificial neural network, they basically decide whether a neuron should be activated or not. Some examples of activation functions are: linear function, binary step function, sigmoid, hyperbolic tangent, Rectified Linear Unit (ReLU), leaky ReLU, exponential LU. We refer to [88] for all definitions and explanations.

In our algorithm Prots2Net presented in this thesis, we use ReLU as activation function, defined as

$$R(z) = max(0, z).$$

Loss function

The loss function in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model. From the loss function, we can derive the gradients which are used to update the weights. Some examples of loss functions are: mean squared error, mean squared logarithmic error, binary cross-entropy, hinge, multi-class cross-entropy, Kullback Leibler divergence, Huber. We refer to [89] for all definitions and explanations. The one used in our algorithm Prots2Net is binary cross entropy, defined as

$$(y * log(p) + (1 - y) * log(1 - p)),$$

where y is the expected value and p the probability to assign y to class 1.

Cost Function

A cost function is a single value, not a vector, since it rates how good the neural network did as a whole. Generally, cost and loss functions are synonymous, but cost function can contain regularization terms in addition to loss function, although it is not always necessary. Some examples of cost functions are: mean, mean squared, mean absolute, root mean squared, categorical cross entropy cost, binary cross entropy cost. We refer to [89] for all definitions and explanations.

The one used in Prots2Net is binary cross entropy, defined as

$$\frac{1}{N}\sum_{i=1}^{N} -(y_i * log(p_i) + (1 - y_i) * log(1 - p_i)),$$

where y_i is the expected value and p_i the probability to assign y_i to class 1.

Optimization algorithms

Optimization algorithms or strategies are responsible for reducing the losses and to provide the most possible accurate results. There are different types of optimizers [83]:

- *Gradient Descent (GD)* is the most basic though the most used optimization algorithm. Gradient descent is an iterative first-order optimization algorithm for finding a local minimum of a differentiable function, which in this case is the loss function. It calculates the weights of neurons so that the loss function can reach a minimum.
- *Stochastic Gradient Descent (SGD)*, is a variant of the Gradient Descent that updates the model's parameters more frequently. For example, if the dataset contains 100 rows, SGD will update the model parameters 100 times in one cycle of the dataset instead of one time as in the Gradient Descent.
- *Mini-Batch Gradient Descent*, is an improvement on both SGD and standard gradient descent. It updates the model parameters after every batch. A batch is a subset of the dataset of a fixed size. So, the dataset is divided into various batches and after every batch, the parameters are updated.
- *Momentum* accelerates the convergence of the differentiable function towards the relevant direction and reduces the fluctuation to the irrelevant direction.
- *Nesterov Accelerated Gradient*, a variant of Momentum developed to resolve the issue that when the momentum is too high, the algorithm may miss the local minimum and may continue to rise up.
- *Adagrad* a type of second order optimization algorithm since it works on the derivative of an error function.
- *AdaDelta* an extension of AdaGrad which tends to remove the decaying learning rate problem of it.
- *Adaptive Moment Estimation* (Adam) works with momentum of first and second order. The intuition behind Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a bit for a careful

search. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta, Adam also keeps an exponentially decaying average of past gradients.

The one used in Prots2Net is Adam, and we have explained all previous algorithms in order to better understand the Adam algorithm.

Regularization

Regularization is a technique that makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well. The most used techniques for regularization are *L1* and *L2*, that owes its name to L1 and L2 norm of the weights vector w respectively.

$$Costfunction = Loss + Regularizationterm$$

Due to the addition of this regularization term, the values of the weight matrices decrease because a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

1.2.2 Types of neural networks

There are different types of neural networks, some of them are: perceptron, FeedForward Neural Networks (FFNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Radial Basis Function Neural Networks (RBFNNs), General Regression Neural Networks (GRNNs), Probabilistic Neural Networks (PNNs), Complementary Neural Networks (CMTNNs). We refer to [87] for all definitions and explanations.

The one used in Prots2Net is a Multilayer Perceptron (MLP), which we briefly explain below.

Multilayer Perceptron

A Multilayer Perceptron (MLP) is a class of feedforward neural networks where the network is composed of perceptrons (with threshold activation).

Perceptron [66] also known as Threshold Logic Unit (TLU) is one of the simplest and oldest models of neural networks, it has only an input layer and an output layer. A perceptron is a binary classifier that separates, by a hyperplane, the input space into two categories. In Figure 1.1 we show the schema of a basic perceptron (left) and a linear hyperplane (right):

A. Schema of a basic perceptron, it consists of a bias node x_0, the input nodes x_j and the output node y. The bias node is set to $+1$, and it is an additional parameter used along with the sum of the product of weights and inputs to produce an output.

B. Linear hyperplane. Represents a classification in two classes, class 1 (C1) shown as green hexagons and class 2 (C2) shown as orange circles. The hyperplane is shown as a blue line, separating both classes.

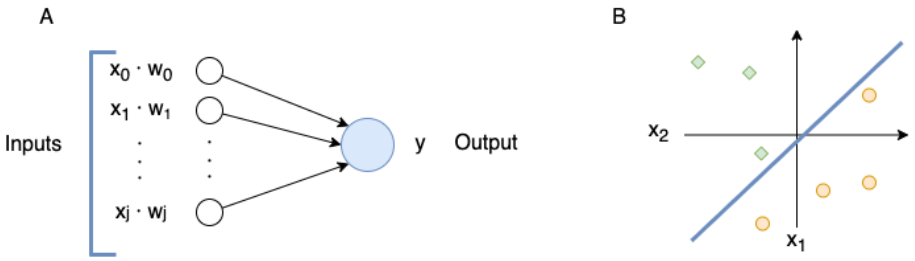


Figure 1.1: Basic perceptron and linear hyperplane.

Feedforward neural networks are called feedforward because the information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are feedback into itself. The goal of a feedforward network is to approximate some function f^*. For example, for a classifier, $y = f^*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation [29].

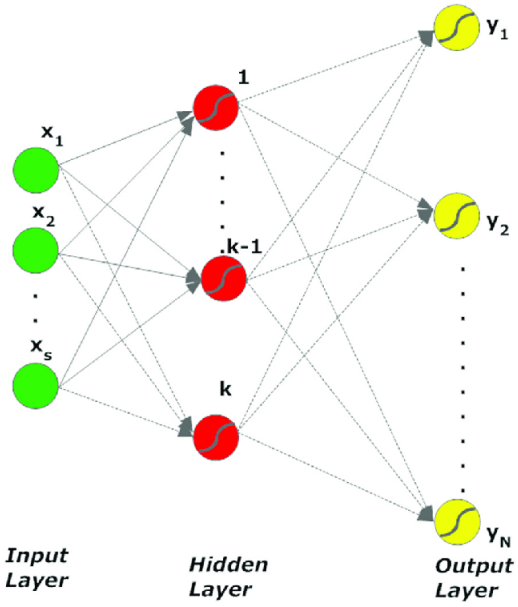


Figure 1.2: Example of a 3 layers feedforward neural network.

Except for the input nodes, each node is a neuron that uses an activation function, the two historically common activation functions are both sigmoids, and are described by

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}.$$

1.3 Binary classification problem & Evaluation

Binary classification is the task of classifying the elements of a set into two groups or classes on the basis of a classification rule. Generally, the two groups are called *positive*

class and *negative class*. Under this assumption, we define the following notions:

- *Positive* (P): it is the number of elements in the positive class.
- *Negative* (N): it is the number of elements in the negative class.
- *True Positive* (TP): it is the number of elements correctly assigned to the positive class by a binary classifier.
- *True Negative* (TN): it is the number of elements correctly assigned to the negative class by a binary classifier.
- *False Positive* (FP): it is the number of elements wrongly assigned to the positive class by a binary classifier.
- *False Negative* (FN): it is the number of elements wrongly assigned to the negative class.

The *confusion matrix* is the matrix whose entries are the values of TP, TN, FP and FN obtained with a binary classifier. Under the notations above, we use the following statistical measures to evaluate the binary classifiers'performance.

Specificity

Also called *True Negative Rate* (TNR), it measures the probability that an element classified as negative is indeed negative:

$$TNR = \frac{TN}{N}.$$

Sensitivity

Also called *True Positive Rate* (TPR) or *recall*, it measures the probability that an element classified as positive is indeed positive:

$$TPR = \frac{TP}{P}.$$

Fall out

Also called *False Positive Rate* (FPR), is the ratio between the number of negative elements wrongly classified as positive and the total number of negative elements:

$$FPR = \frac{FP}{N}.$$

Precision

Also called *Positive Predictive Value (PPV)* (PPV), it measures the proportion of correctly positive classified elements:

$$PPV = \frac{TP}{TP + FP}.$$

Accuracy

The Accuracy (ACC) measures the proportion of correctly classified elements:

$$ACC = \frac{TP + TN}{P + N}.$$

F1 score

The *F1 score* F_1 (F1), is the harmonic mean between precision and sensitivity:

$$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}.$$

Matthews Correlation Coefficient

The term MCC is widely used in the field of bioinformatics and machine learning. This coefficient is a balanced measure between true and false positives and negatives and can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient between the observed and predicted binary classifications. It is defined by

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

ROC curve

A Receiver Operating Characterisc curve, or ROC curve, is a graphical plot that displays the performance of a classification model within a range of classification thresholds from 0 to 1. This curve plots two parameters, the true positive rate (specificity) and the false positive rate. Lowering the classification threshold, the classification model classifies more elements as positive, thus increasing both, the FP and the TP values.

The area under the ROC curve is called AUC (Area Under Curve), that is, AUC measures the entire two-dimensional area underneath the entire ROC curve from $(0, 0)$ to $(1, 1)$. AUC provides an aggregate measure of the performance of a classification model across all possible classification thresholds.

1.4 Bioinformatics & Computational biology

Bioinformatics was born when computers were used in biology to understand biological data. Computational biology is the science of using biological data to develop algorithms or models in order to understand biological systems and relationships.

1.4.1 Genomics (metagenomics) & proteomics (metaproteomics)

Genomics is the study of the complete genome of an organism. Metagenomics is the study of the metagenome, i.e., the collective genome of microorganisms from an environmental sample, to provide information on the microbial diversity and ecology of a specific environment.

Proteomics is the large-scale study of proteins [8, 11]. The proteome is the entire set of proteins produced or modified by an organism or system. A metaproteome is the entire set of proteins produced or modified by a set of microorganisms from an environmental sample.

Comparative genomics and proteomics is the comparison of the genomic or protein features of different organisms. Comparative proteomics analysis may reveal the role of proteins in complex biological systems, including reproduction. Alignment of genome or protein sequences is one of the methodologies introduced in comparative genomics and proteomics. The Basic Local Alignemt Search Tool (BLAST) [94], is the most used tool to sequence alignment. BLAST finds regions of local similarity between sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. BLAST can be used to infer functional and evolutionary relationships between sequences, as well as help identify members of gene families. The score of an alignment, S, is calculated as the sum of substitution and gap scores. Substitution scores are given using a Blocks Substitution Matrix, which is a substitution scoring matrix in which scores for each position are derived from observations of the frequencies of substitutions in blocks of local alignments in related proteins. Each matrix is tailored to a particular evolutionary distance, in the BLOSUM62 matrix, for example, the alignment from which scores were derived was created using sequences sharing no more than 62% identity. Sequences more identical than 62% are represented by a single sequence in the alignment, to avoid over-weighting closely related family members. Gap scores are typically calculated as the sum of G, the gap opening penalty and L, the gap extension penalty. For a gap of length n, the gap cost would be $G + L_n$. The choice of gap costs, G and L is empirical, but it is customary to choose a high value for G $(10 - 15)$ and a low value for L $(1 - 2)$.

The bit score, S', is derived from the raw alignment score, S, taking the statistical properties of the scoring system into account. Because bit scores are normalized with respect to the scoring system, they can be used to compare alignment scores from different searches, and the *normalized bit score* B is the rescaled version which is independent of the size of the search space [14].

One of the most used formats to use BLAST is *FASTA*. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line (defline) is distinguished from the sequence data by a greater-than (">") symbol at the beginning. It is recommended that all lines in the text are shorter than 80 characters in length.

1.4.2 Protein-Protein Interaction Networks (PPINs)

Protein–protein interactions (PPIs) are physical contacts of high specificity established between two or more proteins. PPIs are modeled as pairwise relations in networks called protein–protein interaction network, PPIN for short. A PPIN is an undirected graph whose nodes are proteins and there is an edge between two proteins when they interact.

PPINs alignment

Alignment of PPINs is one of the methodologies used to comparative analysis of molecular networks, in particular the comparative analysis of PPINs or Interactomes. Every

alignment provides a mapping between the nodes of the graphs (proteins) and the corresponding edges (protein–protein interactions). Similarly to the sequence alignment setting, related proteins, or ortholog proteins, are supposed to be matched as well as their interactions. Hence, aligned proteins are supposed to share some biological function.

Formally, a *mapping* $\mu : V \to V'$, is a relation $\mu \subseteq V \times V'$ such that a node $v \in V$ is related with at most one node $v' \in V'$. It is denoted by $\mu(v) = v'$ and v' is called the image of v by μ. A mapping is *injective* when different nodes are mapped to different nodes. More precisely,

$$\forall u, v \in V,\ u \neq v \Rightarrow \nu(u) \neq \nu(v).$$

An *alignment* between two PPINs is an injective mapping, $\mu : V \to V'$ where V and V' are their corresponding sets of nodes (proteins). The domain of μ, denoted as Dom_μ, are the nodes $v \in V$ that have an image. That is,

$$Dom_\mu = \{v \in V \,|\exists\, v' \in V',\ \mu(v) = v'\}.$$

When $Dom_\mu = V$, i.e., the domain is the entire set of nodes in V, it is called a *global alignment*, otherwise it is called a *local alignment*.

1.4.3 Alignment evaluation

There are several measures introduced in the literature to evaluate the quality of a PPINs alignment [16]. The most used quality measures for PPINs alignment that we will use in the next chapters are: the *Edge Correctness ratio* (EC) and the *Functional Coherence value* (FC).

The EC ratio quantifies the amount of network structure preserved by the alignment. It is a topology quality measure defined as:

$$EC(\mu) = \frac{\left|\{\{u, v\} \in E\ :\ \{\mu(u), \mu(v)\} \in E'\}\right|}{min\{|E|, |E'|\}},$$

where μ is the alignment between $G = (V, E)$ and $G' = (V', E')$.

The FC value measures the functional similarity of the aligned proteins by comparing their *Gene Ontology annotation* (GO). A GO annotation is a statement about the function of a particular gene. GO annotations are created by associating a gene or gene product with a GO term. Together, these statements comprise a "snapshot" of current biological knowledge. Hence, GO annotations capture statements about how a gene functions at the molecular level, where in the cell it functions, and what biological processes (pathways, programs) it helps to carry out [9].

Given an alignment, μ, we define the $FC(\mu)$ as

$$FC(\mu) = \frac{\sum_{u \in V} FS(u, \mu(u))}{|V|},$$

where the similarity score FS is the Jaccard score of the GO terms defined by

$$FS(u, u') = \frac{|GO(u) \cap GO(u')|}{|GO(u) \cup GO(u')|},$$

with $GO(u)$ and $GO(u')$ are the sets of GO annotations of the proteins u and u', respectively.

Alternatively, the FC can be defined using the $HRSS$ score defined by

$$FS_{HRSS_{BMA}}(u,u') = \frac{\sum\limits_{g\in GO(u)} \max\limits_{g'\in GO(u')} HRSS(g,\, g') \sum\limits_{g'\in GO(u')} \max\limits_{g\in GO(u)} HRSS(g,\, g')}{|GO(u)| + |GO(u')|},$$

where $HRSS(g,\, g')$ is the *Hybrid Relative Specificity Similarity* of individual GO annotations [99].

Besides this two standard quality measures, the *Complex Functional Coherence* (CFC) is also used to measure the quality of an alignment in terms of its behavior on protein complexes. A protein complex is a group of polypeptide chains linked by non-covalent protein-protein interactions (PPIs), protein complexes play important roles in biological systems and perform numerous functions, such as DNA transcription, mRNA translation, and signal transduction.

The CFC is defined as the ratio of complexes that are "correctly" aligned with respect to the aligned complexes. In this way, a complex is correctly aligned if the complex and its image share some biological function. A pair of complexes, one in each network, is *coherent* when they share some biological function and *incoherent* otherwise. Then CP and NCP are defined as the number of coherent and incoherent pairs of aligned complexes. Finally, CFC is defined by:

$$CFC = \frac{CP}{CP + NCP} \times 100.$$

1.4.4 PPINs aligners

We introduce here the pairwise PPINs aligners used in Chapter 3 and Chapter 4.

PINALOG

PINALOG [79] combines information from protein sequence, function and network topology. The design of the PINALOG alignment method is based on the observation that proteins are not uniformly distributed throughout PPINs. Instead, there are some proteins that form well-connected subnetworks. PINALOG is a global alignment method and comprises three main steps:

1. Community detection: identifies dense subnetworks of input networks using CFinder [73].

2. Community mapping: maps similar communities that have high similarity scores.

3. Extension mapping: maps proteins in the neighbourhood of the core protein pairs which are then added to the core, extension mapping is repeated until no more pair is added.

PINALOG is available at `http://www.sbg.bio.ic.ac.uk/~PINALOG/`.

SPINAL

SPINAL [1] is a polynomial time heuristic algorithm which consist in two main phases:

1. Coarse-grained alignment phase, where all pairwise initial protein similarity scores based on pairwise local neighborhood matchings are constructed.

2. Fine-grained alignment phase that produces the final one-to-one mapping by iteratively growing a locally improved solution subset.

Both phases make use of the construction of neighborhood bipartite graphs and the contributors as a common primitive.

SPINAL is available at `http://code.google.com/p/spinal/`.

HubAlign

HubAlign [34] is a global network alignment algorithm, that makes use of both network topology and sequence homology information. HubAlign consist in three main phases:

1. Estimation of the topological and functional importance of a protein from the global network topology using a minimum-degree heuristic algorithm.

2. Alignment of topologically important proteins.

3. Extension of the alignments to the whole network.

HubAlign is available at `http://ttic.uchicago.edu/~hashemifar/software/HubAlign.zip`.

L-GRAAL

Lagrangian GRAphlet-based ALigner (L-GRAAL) [62] is a global network alignment algorithm which directly optimizes an objective function that takes into account both sequence-based protein conservation and graphlet-based interaction conservation, using an alignment search heuristic based on integer programming and Lagrangian relaxation.

In a first step, L-GRAAL use sequence and graphlet degree similarities to select a subset of the node mappings on which L-GRAAL will optimize seed alignments considering only the node mappings $u \leftrightarrow v$ that u and v are similar enough (considering topology and homology). This seed alignments are generated defining an integer programming problem and solved using a Lagrangian relaxation. In a second step, a greedy heuristic extends the seed alignments using all possible node mappings.

L-GRAAL is available at `http://bio-nets.doc.ic.ac.uk/L-GRAAL/`.

1.4.5 Database resources

There are different sources for PPINs, some of them are:

1. STRING.

2. DIP.

3. UniProt.

4. PIR.
5. MIPS-FunCat.
6. MIPS-CORUM.
7. Isobase.
8. DEG.

STRING

STRING [93] is a database of known and predicted protein-protein interactions. The interactions include direct (physical) and indirect (functional) associations; they stem from computational prediction, from knowledge transfer between organisms, and from interactions aggregated from other (primary) databases.

DIP

The Database of Interacting Proteins (DIP) [100] is a database that documents experimentally determined protein-protein interactions. Because the reliability of experimental evidence varies widely, methods of quality assessment have been developed and utilized to identify the most reliable subset of interactions.

UniProt

The Universal Protein Resource (UniProt) [10] is a comprehensive resource for protein sequence and annotation data. The UniProt databases are the UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef), and the UniProt Archive (UniParc). The UniProt consortium and host institutions EMBL-EBI, SIB and PIR are committed to the long-term preservation of the UniProt databases.

PIR

The Protein Information Resource (PIR) [98] is an integrated public bioinformatics resource to support genomic, proteomic and systems biology research and scientific studies.

FunCat

The Functional Catalogue (FunCat), a hierarchically structured, organism-independent, flexible and scalable controlled classification system enabling the functional description of proteins from any organism [85].

CORUM

The CORUM database provides a resource of manually annotated protein complexes from mammalian organisms. Annotation includes protein complex function, localization, subunit composition, literature references and more. All information is obtained from individual experiments published in scientific articles, data from high-throughput experiments is excluded [84].

IsoBase

Isobase (IsoRank PPI Network Alignment Based Ortholog Database) is a database of functionally related orthologs, which they term "isologs", developed from the multiple alignment of five major eukaryotic PPI networks, as computed by the global network alignment tools IsoRank & IsoRankN - the "iso-" being motivated by the connection of their work to graph isomorphism [75].

DEG

The Database of Essential Genes (DEG) contains all the essential genes that are currently available. The functions encoded by essential genes are considered a foundation of life and therefore are likely to be common to all cells. Users can BLAST the query sequences against DEG [57].

1.4.6 Protein-protein interaction prediction

Experimentally, physical interactions between pairs of proteins can be inferred from a variety of techniques, including yeast two-hybrid systems, protein-fragment complementation assays (PCA), affinity purification/mass spectrometry, protein microarrays, fluorescence resonance energy transfer (FRET), and Microscale Thermophoresis (MST). Efforts to experimentally determine the interactome of numerous species are ongoing.

Experimentally determined interactions usually provide the basis for computational methods to predict interactions, e.g. using homologous protein sequences across species. However, there are also methods that predict interactions de novo, without prior knowledge of existing interactions. We briefly introduce here the prediction methods used in Chapter 5.

DeepPPI

Deep neural networks for Protein–Protein Interactions prediction (DeepPPI) [18], employs deep neural networks to effectively learn representations of proteins from common protein descriptors. The framework of this algorithm consists on five steps:

1. Obtain protein interaction data from the DIP public database.
2. Obtain the protein sequences from PIR or UniProt.
3. High quality features of protein sequence are extracted.
4. Obtain negative samples by random matching of protein in different subcellular locations.
5. Create training and test sets from the positive and negative samples, and feed a training set into the deep neural network.

Rotation Forest & Local Phase Quantization

The method proposed in [97] use a Rotation Forest (RoF) classifier and the Local Phase Quantization (LPQ) descriptor from the Physicochemical Property Response (PR) Matrix of protein amino acids.

The RoF classifier [81] is a method for generating classifier ensembles based on feature extraction. To create the training data for a base classifier, the feature set is randomly split into K subsets (K is a parameter of the algorithm) and Principal Component Analysis (PCA) is applied to each subset in order to reduce its dimension.

The Local Phase Quantization (LPQ) [68] method is a common and efficient texture descriptor that adopts the Fourier transform to analyze the information in a matrix.

To borrow the feature extraction techniques from image processing, it is necessary to preprocess each amino acid sequence by transforming them into a matrix. The method, named Physicochemical Property Response Matrix (PR) [68], is used to represent the protein sequence.

Bio2Vec

Bio2Vec [96] model was constructed based on a feature representation method for biological sequences called bio-to-vector (Bio2Vec) and a Convolutional Neural Networks (CNNs). The Bio2Vec obtains protein sequence features by using a "bio-word" segmentation system and a word representation model used for learning the distributed representation for each "bio-word". Bio2Vec workflow consists of two stages:

1. Generate a fixed-length feature representation for each protein sequence. First segment the protein sequences into protein words, and then the protein words were transformed to a vector by Skip-Gram model.

2. Use a convolutional neural network with multiple convolution kernels for predicting PPIs. Given a pair of protein sequences, represent them using Bio2Vec and then concatenate them to form a feature pair.

3-mers

In [96] a different bio word segmentation system is compared to Bio2Vec, consisting on split a sequence through a sliding window with stride s, where K is the size of window, in particular the 3-mers bio word segmentation is tested.

MCD + SVM

The model presented in [102] is a combination of Multi-scale Continuous and Discontinous (MCD) feature representation and Support Vector Machine (SVM). On this model, the feature selection employed to construct an optimized and more discriminative feature set by excluding redundand features was mRMR. The workflow to predict the PPIs consist of three steps:

1. Represent protein sequences as a vector by using the proposed multiscale continuous and discontinuous (MCD) feature representation.

2. Minimum redundancy maximum relevance (mRMR) is utilized to do the feature selection.

3. SVM predictor is used to perform the protein interaction prediction tasks.

LCPSSMMF

LCPSSMMF [6] is a sequence-based feature extraction which combine local coding position-specific scoring matrix (PSSM) with multifeatures fusion. The workflow of this model consist on three main steps:

1. Use a local coding method based on PSM to build PSSM (CPSSM), which incorporates global and local feature extraction to account for the interactions between residues in both continuous and discontinuous regions of amino acid sequences.

2. Adopt 2 different feature extraction methods (LAG and BP) to capture multiple key feature information by using the evolutionary information embedded in CPSSM.

3. Acquire feature vectors using the multifeatures fusion method.

AC and ACC

In [32] a method combining feature representation using auto cross covariance (ACC) and support vector machine (SVM) is present. AC accounts for the interactions between residues a certain distance apart in the sequence, so this method adequately takes the neighbouring effect into account. This method consist of three main steps:

1. The amino acid residues are translated into numerical values representing physicochemical properties.

2. The numerical sequences are analysed by ACC based on the calculation of covariance.

3. The SVM model is constructed using the vectors of AC variables as input.

ACC results in two kinds of variables, AC between the same descriptor, and cross covariance (CC) between two different descriptors. In [32] the results to use only the AC variables (AC) and the AC plus CC variables (ACC) are analyzed.

LD

In [104] a representation of local protein sequence descriptors and support vector machine (SVM) are combined. To represent the protein sequence, the following steps are done:

1. For each protein sequence, every amino acid is replaced by the index depending on its grouping.

2. Split the amino acid sequences into ten local regions of varying length and composition to describe multiple overlapping continuous and discontinuous interaction patterns within a protein sequence.

3. For each local region, three local descriptors, composition (C), transition (T) and distribution (D), are calculated.

4. The amino acids are divided into seven groups.

5. The descriptors for all local regions were combined, representing the general characteristics of the protein sequence.

6. The vector representing each protein pair is used as a feature vector for input into SVM.

PCA-EELM

In [103] a method of assembling ELM and PCA is presented. The method workflow is:

1. Four kinds of useful sequence-based features such as Auto Covariance (AC), Conjoint triad (CT), Local descriptor (LD) and Moran autocorrelation (MAC) are extracted from each protein sequence to mine the interaction information in the sequence.

2. A feature reduction method, PCA, is employed to extract the most discriminative new feature subset.

3. ELM classifier is constructed using the vectors of resulting feature subset as input.

LD + KNN

In [101] a method using local descriptors and K-Nearest Neighbors (KNNs) learning system is presented with the following workflow:

1. Represent each protein sequence as a vector using a local protein sequence descriptors.

2. Characterize a protein pair in different feature vectors by coding the vectors of two proteins in this protein pair.

3. Construct a KNNs model using the feateature vectors of the protein pair as input.

OLPP + RF

In [53] a computational method for predicting PPIs based on combining Orthogonal Locality Preserving Projection (OLPP) and rotation forest (RoF) models is presented. The method consist on the following steps:

1. Convert the protein sequence into position-specific scoring matrices (PSSMs) containing protein evolutionary information by using the Position-Specific Iterated Basic Local Alignment Search Tool (PSI-BLAST).

2. Characterize the proteins as a fixed length feature vector by applying OLPP to PSSMs.

3. Train the RoF classifier.

MMI + NMBAC

In [17] two different methods to extract features from protein sequence information are used:

- Use k-gram feature representation calculated as Multivariate Mutual Information (MMI).
- Extract additional feature by Normalize Moreu-Broto AutoCorrelation (NMBAC).

These two approaches are used independently or combined to transfrom protein sequence into feature vectors. Then, a Random Forest model is feed with this feature vectors to distinguish interaction pairs from non-interaction pairs.

PseAAC | AC | LDA + RF | RoF | SVM

In [74] three different feature extraction methods are used:

- Pseudo Amino Acid Composition (PseeAAC) that calculates the correlations between residues in the whole protein sequence.
- Auto Covariance (AC) that takes the neighboring effect into account and describes the average interactions between residues with a certain distance apart.
- Latent Dirichlet Allocation (LDA) which is a generative probabilistic model.

Then these methods are combined with these three classifiers

- Support Vector Machine (SVM).
- Random Forest (RF).
- Rotation Forest (RoF).

In [74] the proposed model is the method that combines the generative LDA and discriminative RF models.

DCT + SMR + WSRC

The method proposed in [39] consists on using Discrete Cosine Transform (DCT) on Substitution Matrix Representation (SMR) and then use a Weighted Sparse Representation based Classifier (WSRC), a variant of traditional SRC, which integrates both sparsity and locality structure data.

Discrete cosine transform is a popular linear separable transformation in the lossy signal and image compression processing for its powerful energy compaction property.

In SMR, a $N \times 20$ matrix is generated to represent a N´length protein sequence based on a substitution matrix.

1.5 Software basics

In this section, we briefly introduce some basic topics in software engineering, as well as the programming languages and packages used in this thesis.

1.5.1 Microservices

Microservices - also known as the microservice architecture [70] - is an architectural style that structures an application or a tool as a collection of services that are

- Highly maintainable and testable.
- Loosely coupled.
- Independently deployable.
- Organized around business capabilities.
- Owned by a small team.

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack. To explain the microservice style, it is useful to compare it to the monolithic style: a monolithic application built as a single unit. Enterprise Applications are often built in three main parts: a client-side user interface (consisting of HTML pages and JavaScript running in a browser on the user's machine) a database (consisting of many tables inserted into a common, and usually relational, database management system), and a server-side application. The server-side application will handle HTTP requests, execute domain logic, retrieve and update data from the database, and select and populate HTML views to be sent to the browser. This server-side application is a monolith - a single logical executable. Any changes to the system involve building and deploying a new version of the server-side application [26].

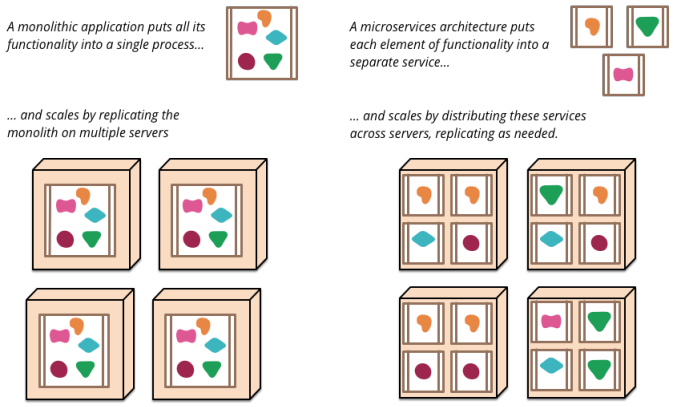


Figure 1.3: Microservices and monoliths [26]

1.5.2 Graphical User Interface

A Graphical User Interface (GUI) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

The GUI was first developed at Xerox PARC by Alan Kay, Douglas Engelbart, and a group of other researchers in 1981. Later, Apple introduced the Lisa computer with a GUI on January 19, 1983.

1.5.3 Programming languages & packages

The programming languages used to implement the tools developed in this thesis, as well as to analyze their performance, are the following:

R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R [25].

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed [24].

To implement Prots2Net, two main libraries has been used:

- Scikit-learn [77].
- tkinter.

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings.

The *tkinter* package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. Tk is a graphical user interface toolkit that takes developing desktop applications to a higher level than conventional approaches. Tk is the standard GUI not only for Tcl, but for many other dynamic languages, and can produce rich, native applications that run unchanged across Windows, Mac OS X, Linux and more.

1.5.4 Databases and other tools used

Celery

Celery is a simple, flexible, and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system. It is a task queue with focus on real-time processing, while also supporting task scheduling [22].

RabbitMQ

RabbitMQ is a message-queueing software also known as a message broker or queue manager. It is a software where queues are defined, to which applications connect in order to transfer a message or messages [43].

Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables to separate the applications from the infrastructure to delivery software quickly. Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows to run many containers simultaneously on a given host [65].

MongoDB

MongoDB is a document database that allow querying and indexing. MongoDB stores the data in flexible, JSON-like documents, so that fields can vary from document to document and the data structure can be changed over time [67].

REDIS

Redis is an open source (Berkeley Software Distribution licensed), in-memory data structure store, used as a database, cache, message broker, and streaming engine. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams [56].

CHAPTER 2

AligNet

In this chapter, we present AligNet, a parameter-free pairwise PPINs alignment algorithm aimed at filling the gap between efficient topologically and biologically meaningful matchings. The overall idea of the algorithm is to obtain many local alignments that are combined and extended into a meaningful global alignment. The final alignment captures the benefits of considering both types of alignments: with the local alignments we capture the topological similarity between the networks, and we speed up the running time of the algorithm, while with the final global alignment we solve the inconsistencies among the local alignments and yield an overall alignment of the pair of input PPINs.

2.1 The structure of the AligNet algorithm

AligNet receives as input two graphs $G = (V, E)$ and $G' = (V', E')$ representing two PPINs (in particular, each node of them is injectively identified with a protein) and it produces, as output, a similarity score for them and a local and a global alignment between them. Figure 2.1 shows the pipeline of our algorithm AligNet. The main steps in AligNet are:

1 The computation of overlapping clusters $C(G)$ and $C(G')$, respectively, of the input networks G and G'.

2 The computation of alignments between pairs of clusters in $C(G)$ and $C(G')$.

3 The computation of a matching between $C(G)$ and $C(G')$.

4 The computation of a local alignment of the input networks G and G'.

5 The extension of this local alignment to a meaningful global alignment.

Figure 2.2 displays two toy PPINs that will be used as a running example throughout this section. The first network, that we will call dme, consists of 8 nodes and 9 edges, while the second network, hsa from now on, consists of 9 nodes and 17 edges.

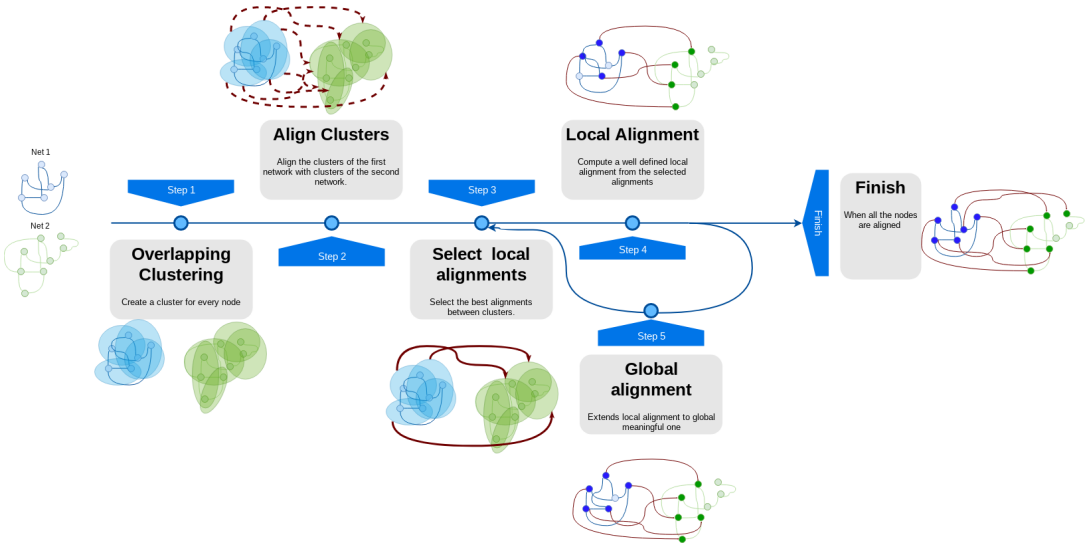


Figure 2.1: Pipeline of AligNet algorithm.

Step 1. Overlapping clusterings.

The first step in AligNet consists in computing an overlapping clustering of each input network. These clusterings are based on the similarity score between pairs of proteins (nodes) u, v in a PPIN G, $s(u, v)$, defined as follows. If u, v are not connected by a path, then $s(u, v) = 0$, and if they are connected,

$$s(u, v) = \frac{B(u, v) + \frac{D(G)+1-d_G(u,v)}{D(G)+1}}{2},$$

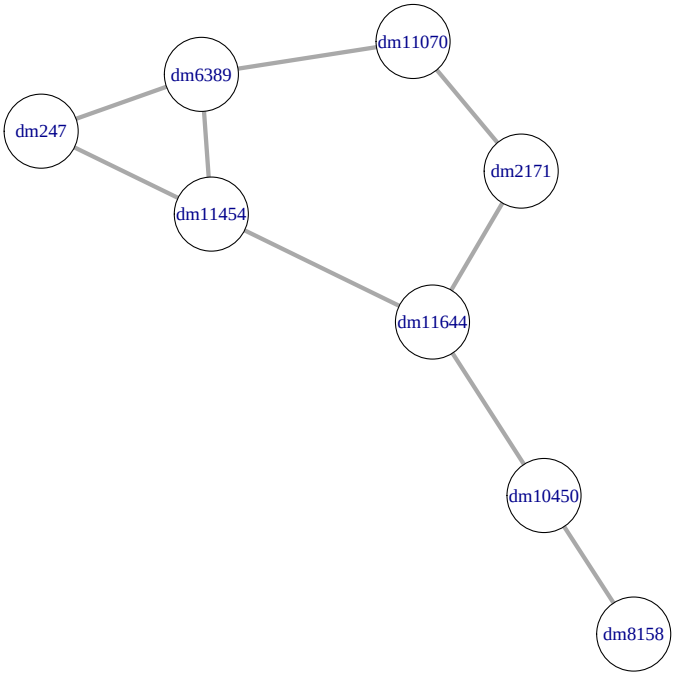
where $B(u, v)$ is the *normalized bit score* of the proteins u and v, that is, the rescaled version of their alignment score obtained with BLAST+, which is independent of the size of the search space (see section 1.4 in the Preliminaries chapter). The intuition behind this similarity score is that two proteins are similar if they have similar sequences of nucleotides and they are relatively close to each other in the graph.

To obtain the overlapping clustering of an input network, we define a cluster centered at each node. To avoid the choice of a fixed and arbitrary cluster size, we considered the similarity score distribution and define the cluster centered at each node as follows. Let α be the third quartile of the distribution of the similarity score values of pairs of nodes, so that only 25% of the pairs of nodes (u, v) are such that $s(u, v) > \alpha$. Then, for every node $u \in V$, the *cluster* C_u in G *centered* at u is

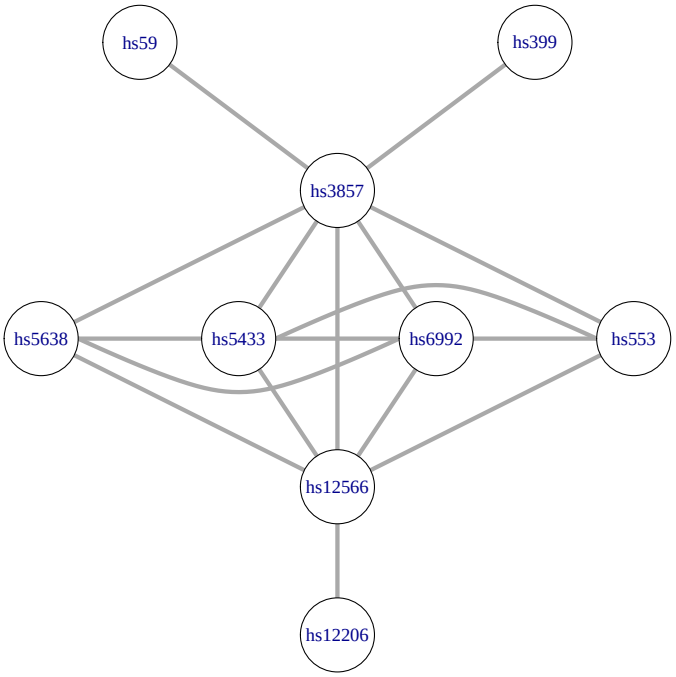
$$C_u = \{v \in V \mid s(u, v) > \alpha\}.$$

Let $C(G) = \{C_u \mid u \in V\}$ and $C(G') = \{C_{u'} \mid u' \in V'\}$.

Table 2.1 shows the normalized bit scores, $B(dm6389, v)$, the distance $d_G(dm6387, v)$, and the similarity score $s(dm6389, v)$ between the node $dm6389$ and all other nodes in the first network, G, in our running example. After calculating the similarity scores for every pair of nodes in the first network, we obtained that the third quartile of their value



(a)



(b)

Figure 2.2: (a) A subnetwork of the *Drosophila melanogaster* PPI network. (b) A subnetwork of the *Homo Sapiens* PPI network.

distribution is 0.43. Hence $\alpha = 0.43$ and the cluster centered at $dm6389$ is

$$C_{dm6389} = \{dm6389, dm247, dm11454, dm11070, dm2171, dm11644\}.$$

Regarding the overlapping clustering of the networks, Figure 2.3 displays all the clusters in our running example. The center of every cluster is highlighted in blue. Since we have considered two small pieces of a PPIN, we obtain here that, the first cluster on the left is the entire piece of network. In the right, we obtain also the entire piece of network in the second cluster on the right. Notice that we obtain the whole piece of the network when we consider the cluster of a node that is in the center of the network.

Table 2.1: Similarity scores obtained between the node $dm6389$ and all other nodes in the first network of our running example.

Protein	B	d_G	s
$dm247$	0.461	1	0.63
$dm11454$	0.462	1	0.63
$dm11070$	0.462	1	0.63
$dm2171$	0.423	2	0.51
$dm11644$	0.423	2	0.51
$dm10450$	0.385	3	0.39
$dm8158$	0.385	4	0.29

Step 2. Alignments between pairs of clusters.

In this second step, AligNet computes an alignment between every pair of clusters $C_u \in C(G)$ and $C_{u'} \in C(G')$ such that $B(u, u') > 0$. These alignments define an alignment score between every such a pair of clusters that will be used in the third step to compute a matching between $C(G)$ and $C(G')$.

Formally, for every $u \in V$ and $u' \in V'$ such that $B(u, u') > 0$, the alignment between $C_u \in C(G)$ and $C_{u'} \in C(G')$ is obtained as follows:

(i) Match u with u'. Set $L_{u,u'} = \{(u, u')\}$, $L^{(1)}_{u,u'} = \{u\}$ and $L^{(2)}_{u,u'} = \{u'\}$.

(ii) For every $v \in C_u \cap N_G(u)$ and for every $v' \in C_{u'} \cap N_{G'}(u')$, let

$$F(v, v') = |\deg(v) - \deg(v')| - B(v, v') + 1.$$

Compute a matching $M_{u,u'} \subseteq (C_u \cap N_G(u)) \times (C_{u'} \cap N_{G'}(u'))$ that minimizes $\sum_{(v,v')\in M_{u,u'}} F(v, v')$ using the Hungarian algorithm (see subsection 1.1.2 in the Preliminaries chapter). Sort the pairs in $M_{u,u'}$ in decreasing order of their F value, and concatenate them to $L_{u,u'}$. Add their first coordinates to $L^{(1)}_{u,u'}$ and their second coordinates to $L^{(2)}_{u,u'}$.

(iii) Iterate step (ii), replacing (u, u') by the rest of the pairs in $L_{u,u'}$ and removing from C_u and $C_{u'}$ the nodes already aligned.

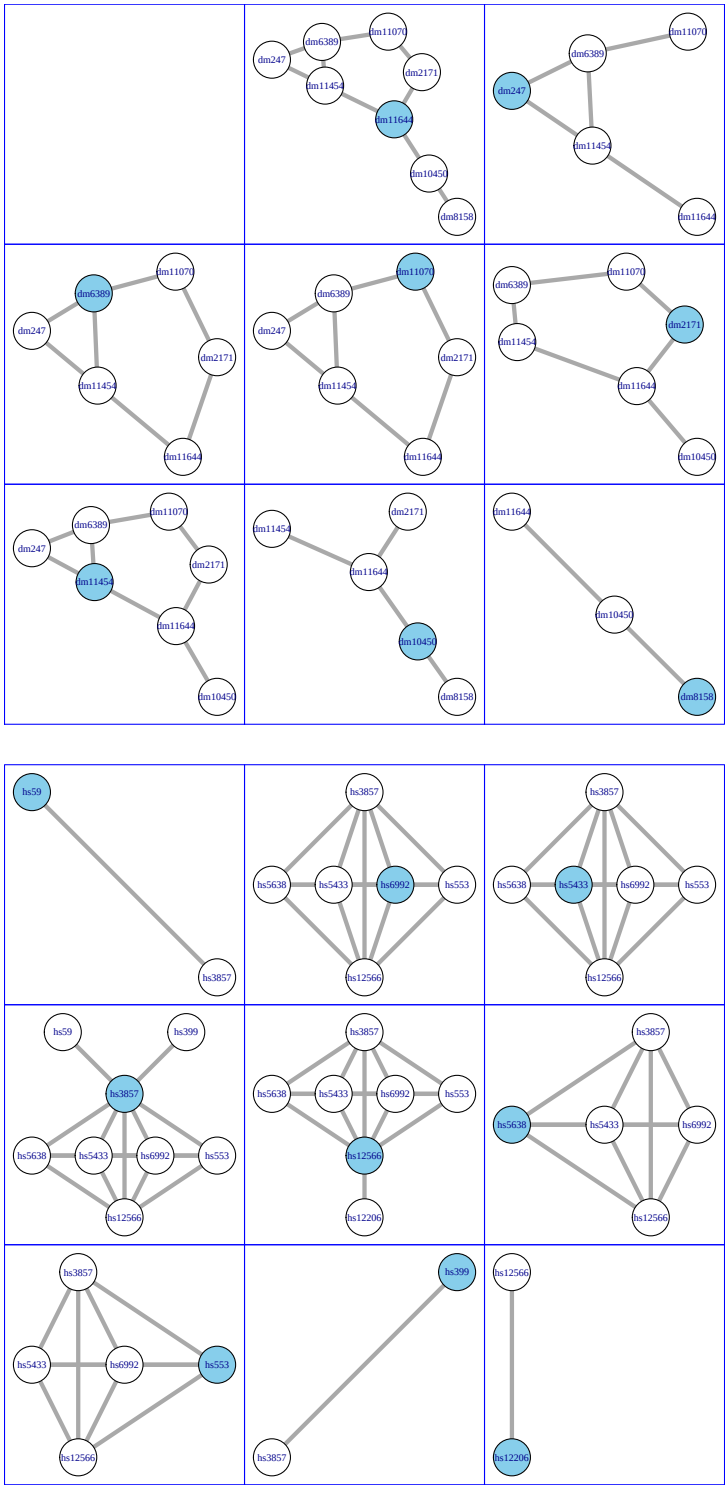


Figure 2.3: Overlapping clusterings. This figure shows the overlapping clustering on the PPINs in Figure 2.2 obtained by AligNet. We can see here the 8 clusters in the dme network on the top, and the 9 clusters in the hsa network on the bottom.

More specifically, in the k-th iteration, take the k-th element (v_0, v_0') of $L_{u,u'}$. For every $w \in (C_u \setminus L_{u,u'}^{(1)}) \cap N_G(v_0)$ and every $w' \in (C_{u'} \setminus L_{u,u'}^{(2)}) \cap N_{G'}(v_0')$, compute $F(w, w')$. Then, compute a matching

$$M_{v_0,v_0'} \subseteq \big((C_u \setminus L_{u,u'}^{(1)}) \cap N_G(v_0)\big) \times \big((C_{u'} \setminus L_{u,u'}^{(2)}) \cap N_{G'}(v_0')\big),$$

that minimizes $\sum_{(v,v') \in M_{v_0,v_0'}} F(v, v')$. Sort the pairs forming $M_{v_0,v_0'}$ in decreasing order of their F value, and concatenate them to $L_{u,u'}$. Add their first coordinates to $L_{u,u'}^{(1)}$ and their second coordinates to $L_{u,u'}^{(2)}$.

The resulting alignment $L_{u,u'}$ defines a partial injective mapping $\eta_{u,u'} : C_u \to C_{u'}$. The nodes in C_u that are matched to nodes in $C_{u'}$ form the domain of the mapping $\eta_{u,u'}$, which is denoted by $Dom\, \eta_{u,u'}$.

The general idea behind this alignment procedure is that u is matched to u' and then a node $v \in C_u$ should be matched to a node v' in $C_{u'}$ when they have similar sequences and similar degrees, provided that, furthermore, there exist paths connecting u with v and u' with v' such that their intermediate nodes are already aligned in sequential order along the paths. The alignment procedure gives priority to matching neighbors of nodes x, x' at the possible shortest distance of the respective cluster centers and with $F(x, x')$ as large as possible among those pairs already matched at the same iterative step.

Figure 2.4, 2.5, 2.6, 2.7 displays the alignment between the clusters $C_{dm11644}$ and C_{hs5433}. The alignment procedure is as follows:

1. Match $dm11644$ with $hs5433$, set $L_{dm11644,hs5433} = \{(dm11644, hs5433)\}$ (see Figure 2.4).

2. Compute the matching that minimizes $\sum_{(v,v') \in M_{dm11644,hs5433}} F(v, v')$, where $M_{dm11644,hs5433}$ are the pairs in the table below:

dm	hs
$dm11454$	$hs3857$
$dm11454$	$hs553$
$dm11454$	$hs12566$
$dm11454$	$hs5638$
$dm11454$	$hs6992$
$dm2171$	$hs3857$
$dm2171$	$hs553$
$dm2171$	$hs12566$
$dm2171$	$hs5638$
$dm2171$	$hs6992$
$dm10450$	$hs3857$
$dm10450$	$hs553$
$dm10450$	$hs12566$
$dm10450$	$hs5638$
$dm10450$	$hs6992$

3. The sorted pairs in $M_{dm11644,hs5433}$ in decreasing order of their F value are,

$$\{(dm2171, hs5638), (dm11454, hs553), (dm10450, hs3857)\}.$$

Hence, as we can see in Figure 2.5,

$$L_{dm11644,hs5433} = \Big\{(dm11644, hs5433), (dm2171, hs5638) \\ (dm11454, hs553), (dm10450, hs3857)\Big\}.$$

4. Repeat step 2 with the pair $(dm2171, hs5638)$. Thus, compute the matching that minimizes $\sum_{(v,v')\in M_{dm2171,hs5638}} F(v, v')$, where

$$M_{dm2171,hs5638} = \{(dm11070, hs3857), (dm11070, hs12566)\}.$$

Since $F(dm11070, hs3857) > F(dm11070, hs12566)$, we add $(dm11070, hs12566)$ to $L_{dm11644,hs5433}$ (see Figure 2.6).

5. Repeat the previous step with $(dm11454, hs553)$, then the assignment to add is $(dm6389, hs553)$ (see Figure 2.7).

6. Since all the nodes in C_{hs5433} are assigned to a node in $C_{dm11644}$, the alignment $L_{dm11644,hs5433}$ is done

$$L_{dm11644,hs5433} = \Big\{(dm11644, hs5433), (dm2171, hs5638) \\ (dm11454, hs553), (dm10450, hs3857), \\ (dm11070, hs12566), (dm6389, hs553)\Big\}.$$

Step 3. Matching between families of clusters.

Let

$$\mathcal{A} = \{\eta_{u,u'} \mid u \in V,\ u' \in V', B(u, u') > 0\},$$

be the set of alignments obtained in step 2. The *score* of each $\eta_{u,u'} \in \mathcal{A}$ is defined as

$$Score(\eta_{u,u'}) = \frac{\sum_{v\in Dom\,\eta_{u,u'}} B(v, \eta_{u,u'}(v))}{|Dom\,\eta_{u,u'}|} + \frac{|Dom\,\eta_{u,u'}|}{max_{\eta_{w,w'}\in\mathcal{A}}|Dom\,\eta_{w,w'}|}.$$

This score assesses simultaneously the average similarity of the sequences of the proteins matched by $\eta_{u,u'}$ and their number.

Once computed all these scores, AligNet obtains a matching between $C(G)$ and $C(G')$ by applying the maximum weighted bipartite matching algorithm to the bipartite graph whose nodes are the clusters in $C(G)$ and $C(G')$, whose edges connect pairs of clusters $C_u \in C(G)$ and $C_{u'} \in C(G')$ with $B(u, u') > 0$, and the weight of the edge connecting C_u with $C_{u'}$ is the score $Score(\eta_{u,u'})$. We shall denote by $\mathcal{C}$ the set of partial injective mappings $\eta_{u,u'}$ corresponding to pairs of clusters $(C_u, C_{u'})$ that are matched by this matching.

Figure 2.8 shows the matching obtained in this step between the families of clusters in Figure 2.3. Notice that, C_{hs399} remains unassigned because the second network has more clusters than the first one.

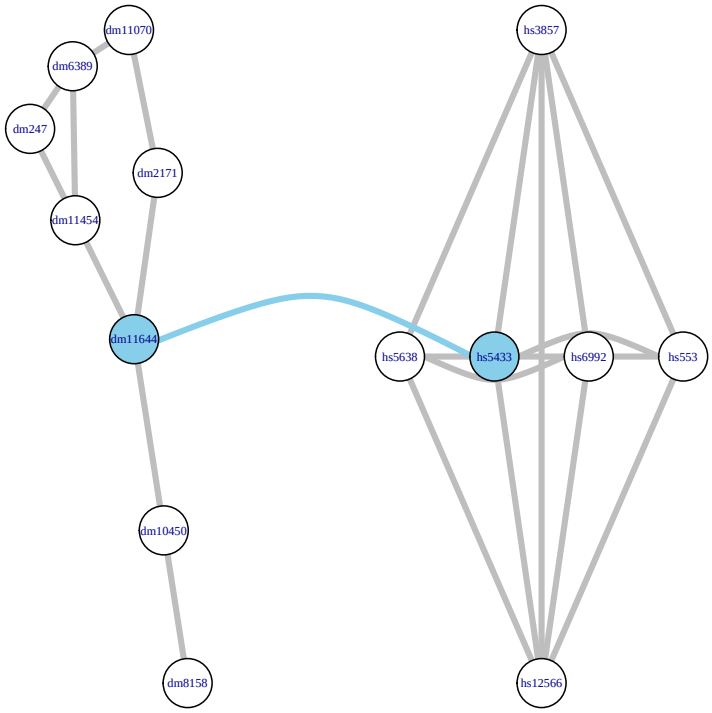


Figure 2.4: First step of the alignment of a pair of clusters.

Step 4. Local alignment of PPINs.

In this step, AligNet produces a local alignment between G and G' from the matching between $C(G)$ and $C(G')$ obtained in the previous step.

The main idea is to define this alignment by merging the partial injective mappings $\eta_{u,u'} \in \mathcal{C}$. The problem is that these mappings may be inconsistent. A first approach to overcome this problem would be to consider the weighted bipartite hypergraph with set of nodes $V \sqcup V'$ and where every mapping $\eta_{u,u'}$ defines a hyperarc with source its domain, target its image, and weight $Score(\eta_{u,u'})$, and to solve on it the weighted bipartite hypergraph assignment problem, whose solution would provide a well-defined local alignment of the input networks.

However, in order to decrease the computation time of AligNet, we do not define this hypergraph from the whole $\mathcal{C}$, but just from a subset $\mathcal{R}$ of *best-scored* alignments built recursively as follows. Starting with $\mathcal{R} = \emptyset$, AligNet adds to $\mathcal{R}$ at each step a mapping $\eta_{w_0,w'_0} \in \mathcal{C}$ with w_0 not belonging to the union of the domains of the mappings $\eta_{w,w'}$

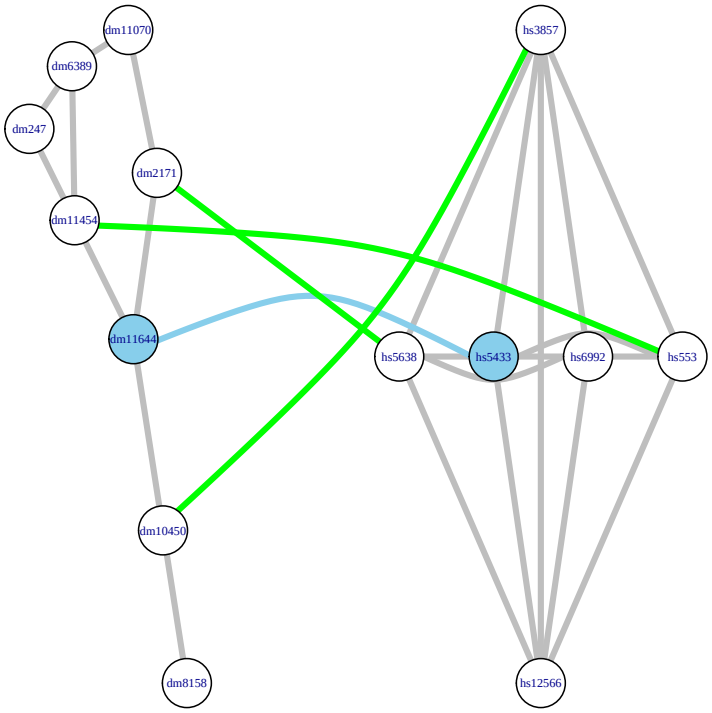


Figure 2.5: Second step of the alignment of a pair of clusters.

already in $\mathcal{R}$ and with maximum $Score(\eta_{w_0,w_0'})$ among all such mappings. AligNet iterates this procedure until every node in $\bigcup_{\eta_{u,u'} \in \mathcal{C}} Dom\, \eta_{u,u'}$ belongs to the domain of some mapping in $\mathcal{R}$. In Figure 2.12 we give the subset $\mathcal{R}$ of $\mathcal{C}$ for the networks in our running example. We also explain below how the appropriate set of alignments is selected on our running example.

1. First, the mapping with maximum score is $\eta_{dm11644,hs5433}$, (see Figure 2.9).

2. Next, the mappings such that the center of the cluster does not belong to $Dom_{\eta_{dm11644,hs5433}}$, are the mappings whose centers are $dm247$ and $dm8158$, (see Figure 2.10).

3. Since the score of the mapping $\eta_{dm247,hs12566}$ is higher than the score of the mapping $\eta_{dm8158,hs12206}$, we add $\eta_{dm247,hs12566}$ to the hypergraph, (see Figure 2.11).

4. Finally, we add $\eta_{dm8158,hs12206}$ to the hypergraph, (see Figure 2.12).

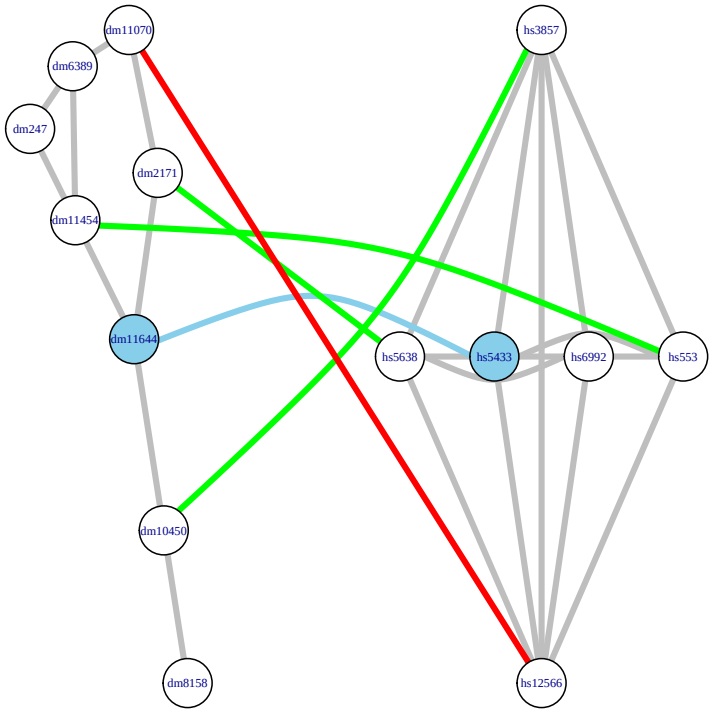


Figure 2.6: Third step of the alignment of a pair of clusters.

Notice that in the end, that is, when we consider the three alignments together, there are four nodes in the source network with inconsistent assignments.

Then, Alignet obtains from the directed hypergraph with nodes $V \sqcup V'$ and hyperarcs defined by the mappings $\eta_{u,u'} \in \mathcal{R}$ as explained above, a local well-defined alignment between G and G' as a solution of the corresponding weighted bipartite hypergraph assignment problem (see subsection 1.1.2 in the Preliminaries chapter).

Figure 2.13 shows the local alignment of the original networks obtained by AligNet in its fourth step, once the inconsistent assignments have been solved. The coherent assignment of nodes is obtained as the solution to the weighted bipartite hypergraph assignment problem, for the hypergraph associated to the appropriate set of alignments described in Figure 2.12. In this case, the hypergraph has three hyperarcs, corresponding to the three alignments considered in the appropriate set of alignments.

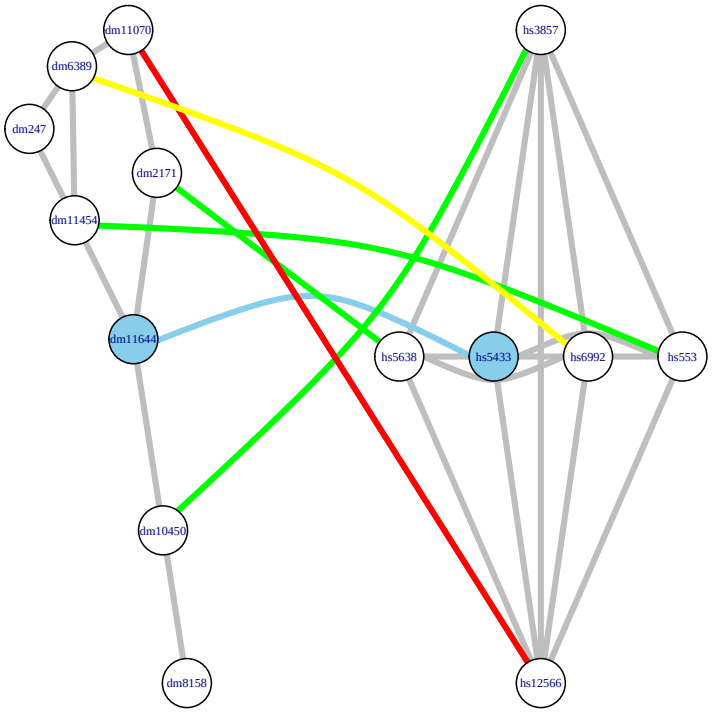


Figure 2.7: Fourth step of the alignment of a pair of clusters.

Step 5. Global meaningful alignment of PPINs.

In order to extend the local alignment produced in the previous step, AligNet iterates the following procedure:

- It removes the nodes in G and G' that have already been aligned, and it recomputes the score of each alignment $\eta_{u,u'}$ following the same definition as in step 3, but only taking into account the remaining nodes in its domain and image.
- It computes a new optimal matching $\mathcal{C}$ between $C(G)$ and $C(G')$, as in step 3, but using as edges those $\eta_{u,u'}$ whose updated score is positive, and weights these updated scores.
- It computes a new set $\mathcal{R}$ of best-scored alignments $\eta_{u,u'}$ with $Score(\eta_{u,u'}) > 0$, as in step 4.

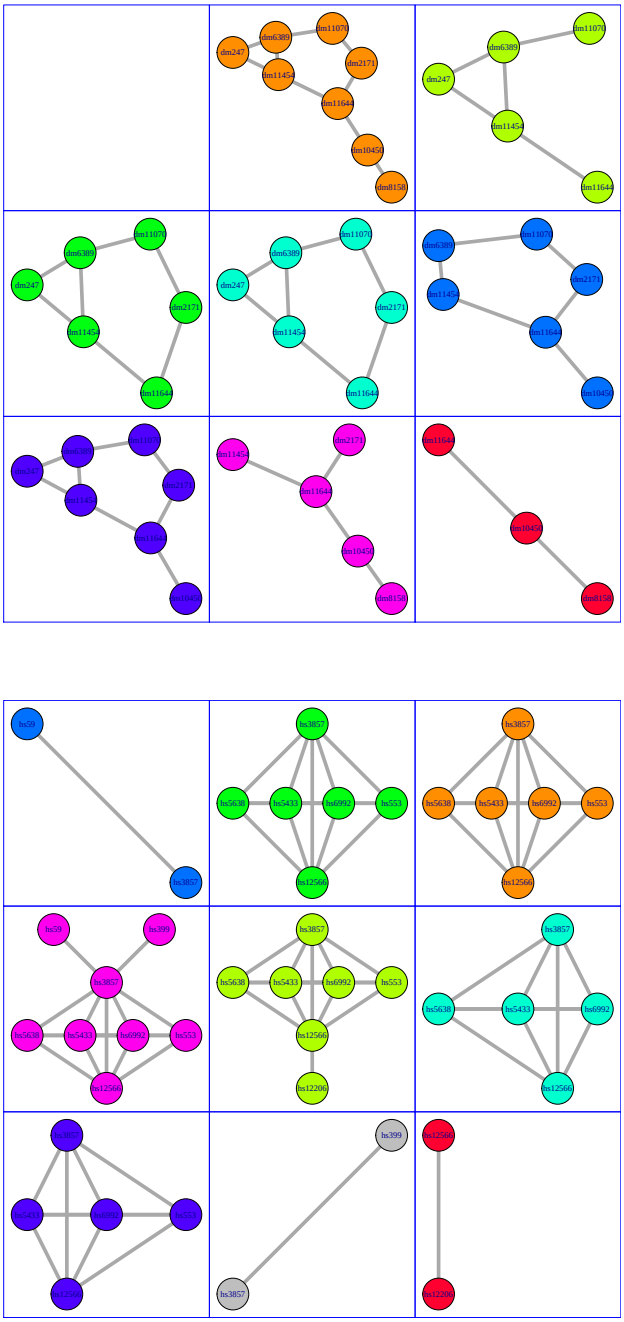


Figure 2.8: Alignment of the clusters.

- It defines a new directed hypergraph whose nodes are the nodes in $V \cup V'$ not yet aligned and hyperarcs the mappings $\eta_{u,u'}$ in the new set $\mathcal{R}$, understood as hyperarcs with source the still unaligned nodes in their domain and target the still unaligned nodes in their image.

- It computes a local alignment between unaligned nodes in V and V' by solving the weighted bipartite hypergraph assignment problem for this hypergraph, and it adds

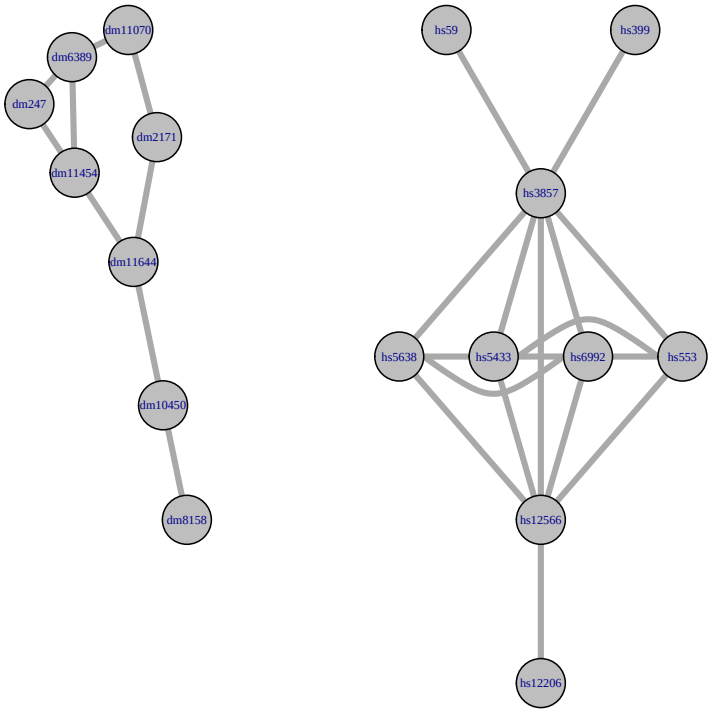


Figure 2.9: First step of the appropriate set of alignments.

this local alignment to the alignment obtained so far.

This procedure is iterated while there exist nodes not aligned belonging to the domain or the image of some alignment $\eta_{u,u'}$ with (updated) positive score. In Figure 2.14 we show the final global meaningful alignment obtained with AligNet for the networks in our running example. Notice that, in the fifth step of AligNet, the previous alignment is extended to a global one. In this case, there were two unmatched nodes in the source network in Figure 2.13 which are now assigned.

2.2 How to use the library

AligNet is implemented in R. It can be installed via GitHub, using the function available in `devtools`, `install_github("adriaalcala/AligNet")`. In order to facilitate its use, some data is available in the library. We explain below how to use it.

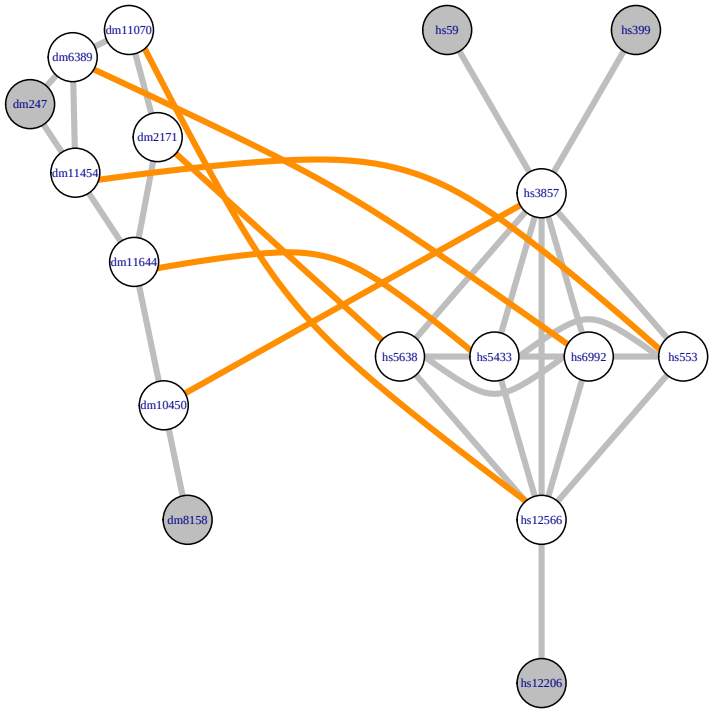


Figure 2.10: Second step of the appropriate set of alignments.

Insert and load data

To build the networks from a list of protein interactions, use the function `read.network` as in the following example:

```
edges1 <- matrix(c(
  "85962.HP0109", "85962.HP0136",
  "85962.HP0109", "85962.HP0137",
  "85962.HP0136", "85962.HP0247",
  "85962.HP0136", "85962.HP0303",
  "85962.HP0137", "85962.HP0247",
  "85962.HP0137", "85962.HP0853",
  "85962.HP0247", "85962.HP1316"
 ), ncol=2, byrow=TRUE)
hpy <- read.network(edges1, mode="edges")
```

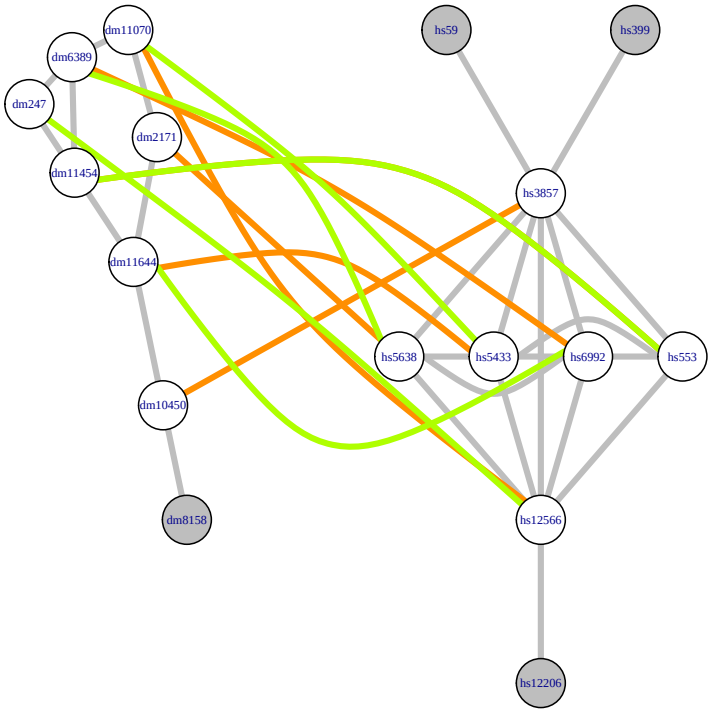


Figure 2.11: Third step of the appropriate set of alignments.

```
edges2= matrix(c(
  "DBP2_YEAST", "RL2A_YEAST",
  "HAS1_YEAST", "MAK5_YEAST",
  "NOP10_YEAST", "DBP2_YEAST",
  "NOP10_YEAST", "HAS1_YEAST",
  "NOP10_YEAST", "MAK5_YEAST",
  "NOP10_YEAST", "RL2A_YEAST",
  "TSA1_YEAST", "HSP7F_YEAST",
  "TSA1_YEAST", "TSA2_YEAST"
 ), ncol=2, byrow=TRUE)
sce <- read.network(edges2,mode="edges")
```

The networks can be visualized using `plot(hpy)` and `plot(sce)`.

To upload the similarity matrix, you can use the function `read.matrix` if you

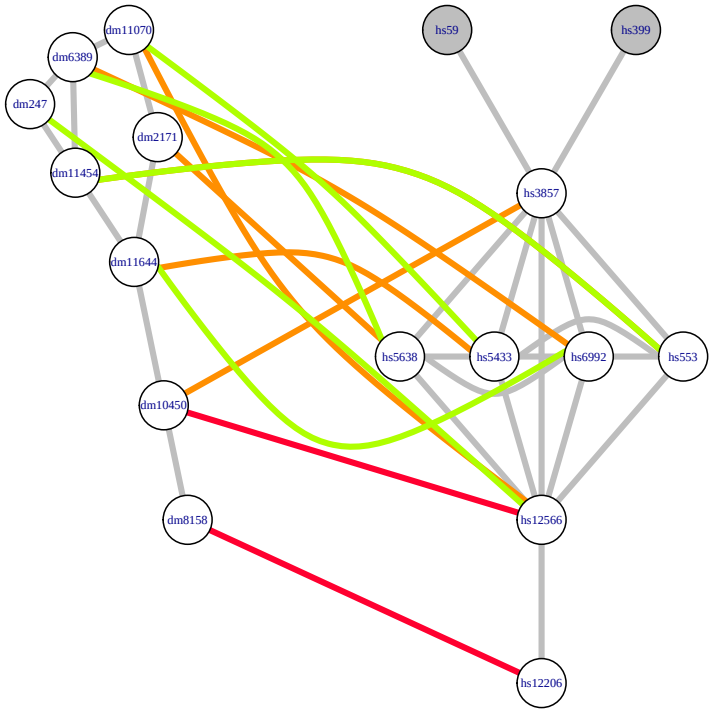


Figure 2.12: Final step of the appropriate set of alignments.

want to read the matrix from a file, or you can compute the similarity matrix. In this case, the data files `Sim1` and `Sim2` are uploaded and included in the package to compute the similarity matrix with the function `compute.matrix`.

```
data(Sim1)
data(Sim2)
Dis1 = compute.matrix(net1=hpy)
Dis2 = compute.matrix(net1=sce)

Sim1 = (Sim1+Dis1)/2
Sim2 = (Sim2+Dis2)/2
```

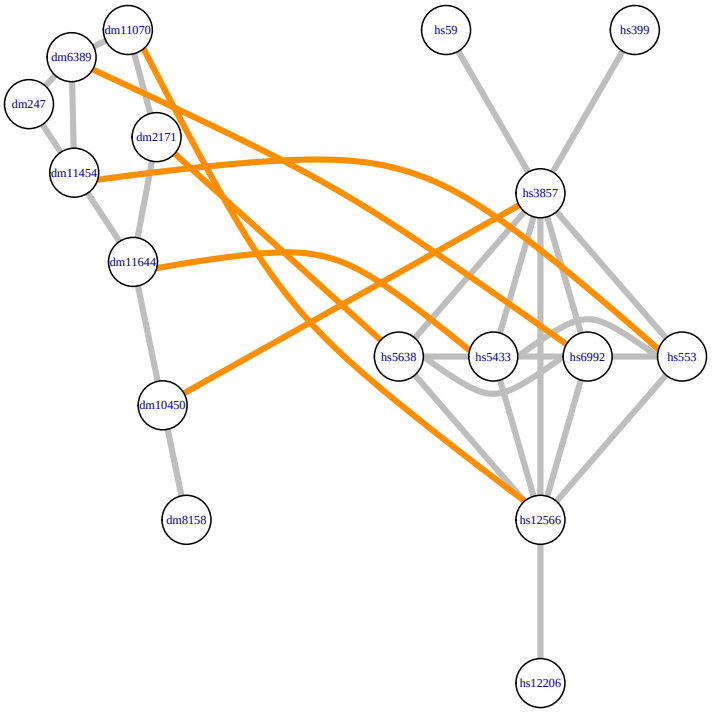


Figure 2.13: Local alignment obtained in the fourth step.

Compute clusters

Once the networks and the similarity matrices are computed, the clusters can be obtained with the functions `cluster.network` and `extract.clusters`.

```
clust1 = cluster.network(sigma=Sim1,lambda=0.2,k=5)
clusters1 = extract.clusters(Net=hpy,ClustMat=clust1)

clust2 = cluster.network(sigma = Sim2, lambda = 0.2, k = 5)

clusters2 = extract.clusters(Net = sce,ClustMat = clust2)
```

Local alignment

The local alignments of clusters are computed using `align.local.all`.

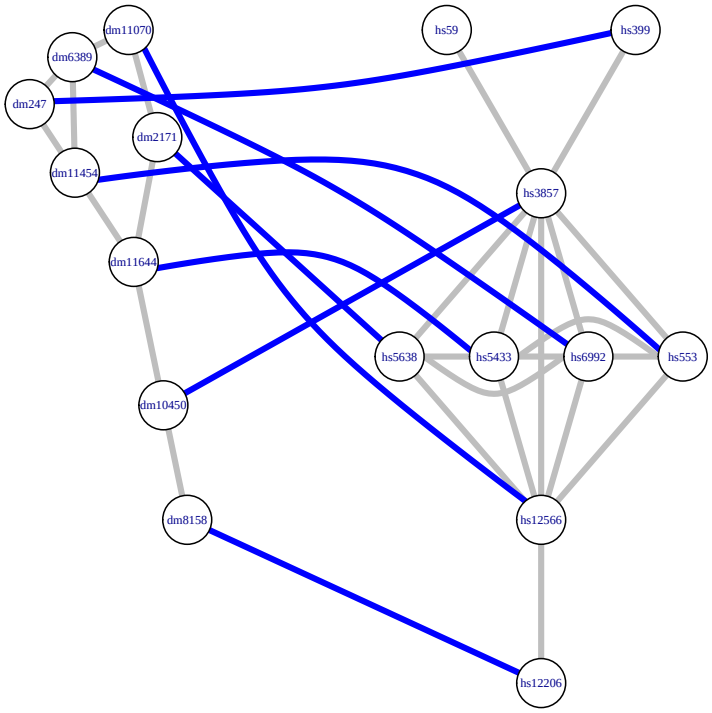


Figure 2.14: Final global alignment.

```
data(Sim)
localAligns = align.local.all(
    clust1 = clusters1,
    clust2 = clusters2,
    mat = Sim,
    threshold = 0
)
```

Global alignment

Finally, from the local alignments, the global alignment is computed using `align.global`.

```
alinGlobal = align.global(localAligns=localAligns,Sim=Sim)
```

2.3 Comparison with other aligners

In this section we report the tests performed to assess the performance of AligNet. Following the comparisons published in [16, 64], we decided to compare AligNet with SPINAL [1], HubAlign [34], L-GRAAL [62], and PINALOG [79] on the dataset used in [16], which consists of the PPINs of *M. musculus* (mus), *C. elegans* (cel), *D. melanogaster* (dme), *S. cerevisiae* (sce), and *H. sapiens* (hsa), downloaded from the IsoBase (see subsection 1.4.5 in Preliminaries chapter); see Table 2.2. We refer to subsection 1.4.4 in the Preliminaries chapter for a brief description of these aligners.

Table 2.2: Number of nodes and edges (with and without loops) of the PPINs considered as input data in our tests.

	Nodes	Edges (with loops)	Edges (without loops)
M. musculus	623	776	559
C. elegans	2,995	8,639	4,827
S. cerevisiae	5,524	164,718	82,656
D. melanogaster	7,396	49,467	24,937
H. sapiens	10,403	105,232	54,654

In a first assessment of the alignments, we used two quality measures: the *edge correctness ratio* (*EC*), which quantifies the amount of structure preserved by the alignment, and the *functional coherence value* (*FC*), which assesses the functional similarity of the aligned proteins by comparing their *Gene Ontology annotation* (see subsection 1.4.3 in the Preliminaries chapter).

Tables 2.3 and 2.4, as well as Figure 2.15 and Figure 2.16, give the EC and FC scores, respectively, of the alignments produced by the aligners under consideration (using the aligners' parameters suggested by default whenever it was needed).

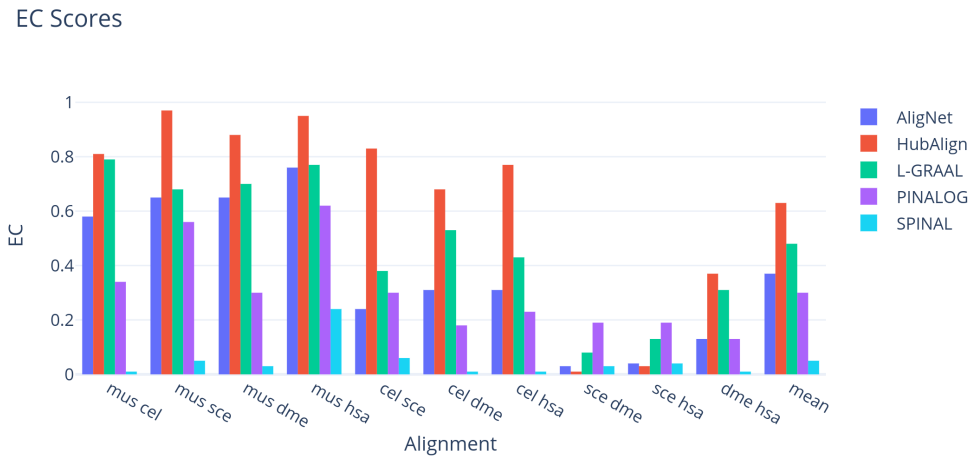


Figure 2.15: Edge Correctness Scores. This figure shows the edge correctness score obtained for each aligner in every alignment. The different aligners are presented in different colours.

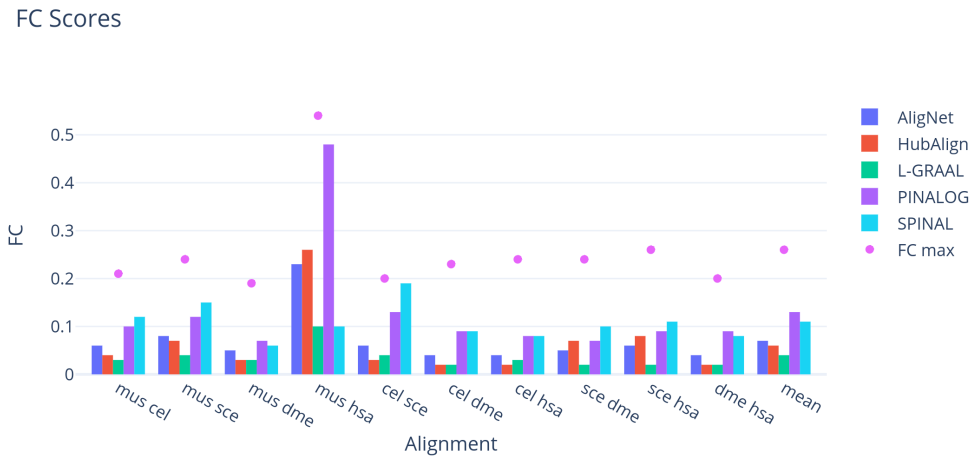


Figure 2.16: Functional Coherence Scores. This figure shows the functional coherence score obtained for each aligner in every alignment. In a purple dot we show the maximal value expected for every The different aligners are presented in different colours.

Table 2.3: Edge correctness ratio obtained in every alignment.

Net1	Net2	AligNet	HubAlign	L-GRAAL	PINALOG	SPINAL
mus	cel	0.58	0.81	0.79	0.34	0.01
mus	sce	0.65	0.97	0.68	0.56	0.05
mus	dme	0.65	0.88	0.70	0.30	0.03
mus	hsa	0.76	0.95	0.77	0.62	0.24
cel	sce	0.24	0.83	0.38	0.30	0.06
cel	dme	0.31	0.68	0.53	0.18	0.01
cel	hsa	0.31	0.77	0.43	0.23	0.01
sce	dme	0.03	0.01	0.08	0.19	0.03
sce	hsa	0.04	0.03	0.13	0.19	0.04
dme	hsa	0.13	0.37	0.31	0.13	0.01
mean		0.37	0.63	0.48	0.30	0.05

We can observe there that the alignments of small networks with a few edges, such as *M. musculus*, produced alignments with high EC scores, especially when the target network has many edges. However, we can also observe that, when the number of edges in the source network increased, the EC scores decreased dramatically even in the case of HubAlign.

As far as the functional coherence goes, we can observe in Table 2.4 and Figure 2.16 that all aligners attained a very low FC score, and the order from the highest to the lowest FC scores is almost the opposite to the order obtained when considering the EC scores. To put these low scores in perspective, we estimated the maximum value FC_{max} of the FC score for every pair of networks, by solving the maximum weighted bipartite matching problem on the complete bipartite graph where the nodes are the proteins in the two input networks and the weight of each edge connecting one protein in a network to a protein in the other network is the FC score of the corresponding pair of proteins. These maximum values are given in the first numerical column of Table 2.4.

Table 2.4: Functional coherence value obtained in every alignment.

Net1	Net2	FC_{max}	AligNet	HubAlign	L-GRAAL	PINALOG	SPINAL
mus	cel	0.21	0.06	0.04	0.03	0.10	0.12
mus	sce	0.24	0.08	0.07	0.04	0.12	0.15
mus	dme	0.19	0.05	0.03	0.03	0.07	0.06
mus	hsa	0.54	0.23	0.26	0.10	0.48	0.10
cel	sce	0.20	0.06	0.03	0.04	0.13	0.19
cel	dme	0.23	0.04	0.02	0.02	0.09	0.09
cel	hsa	0.24	0.04	0.02	0.03	0.08	0.08
sce	dme	0.24	0.05	0.07	0.02	0.07	0.10
sce	hsa	0.26	0.06	0.08	0.02	0.09	0.11
dme	hsa	0.20	0.04	0.02	0.02	0.09	0.08
mean		0.26	0.07	0.06	0.04	0.13	0.11

As it can be seen in Tables 2.3 and 2.4, AligNet and HubAlign obtained the best balance between FC and EC scores, followed by PINALOG and L-GRAAL.

In addition, in order to measure the amount of variation or dispersion of the EC and FC scores, we introduce some *noise* to the networks considered in this test by randomly adding and deleting a 5% of the edges. Thus, we compute 100 new alignments with every aligner, and we calculate the corresponding EC and FC scores. That is, we obtain a sample of the EC and FC scores, and we calculate the sample standard error and the corresponding confidence scores.

We also compared the behavior of AligNet, PINALOG, HubAlign, and L-GRAAL in relation to the alignment of protein complexes (we excluded SPINAL from this test because its results in the EC and FC tests were not convincing). Following the procedure explained in [79], we considered the database MIPS CORUM (see the subsection 1.4.5 in the Preliminaries chapter) as the gold standard for the human protein complexes and the information available in [27] as the gold standard for the yeast complexes. In addition, we considered the functional information available in MIPS CORUM for the human complexes and in MIPS FunCat (see the subsection 1.4.5 in the Preliminaries chapter) for the yeast complexes. To measure the quality of an alignment in terms of its behaviour on protein complexes, we used the *complex functional coherence value* (CFC), defined as the ratio of complexes that are aligned correctly with respect to the aligned complexes. More specifically, if we call a pair of complexes, one in each network, *coherent* when they share some biological function and *incoherent* otherwise, and if we denote by CP and NCP the numbers of coherent and incoherent pairs of aligned complexes, then $CFC = \frac{CP}{CP+NCP} \times 100$. In Table 2.5, Figure 2.17 and Figure 2.18 we show the results obtained by all the aligners. The result is that AligNet has the highest CFC value (25.34) followed by PINALOG (24.48) and HubAlign and L- GRAAL obtained a very low CFC value (5, 4.75 resp.).

In order to further compare the results obtained by AligNet on protein complexes with those of the others aligners, we counted, for each other aligner A, the complexes that were not aligned either by AligNet or by A; the coherent and incoherent pairs among those complexes that were aligned by AligNet but not by A; and the coherent and incoherent pairs among those complexes that were aligned by A but not by AligNet. The results are given in Table 2.6. For instance, in its first two numerical columns we can see that 891 complexes were not aligned neither by AligNet nor by HubAlign; 263 complexes were aligned by AligNet but not by HubAlign, of which 88 were correctly aligned

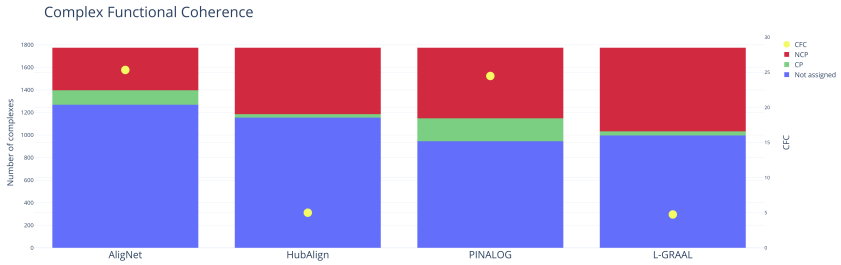


Figure 2.17: Complex Functional Coherence. This figure shows the number of non-assigned complexes (in blue), the number of coherent pairs (in green), the number of incoherent pairs (in red) and the complex functional coherence value (yellow dot). The number of complexes is shown on the left axis, while the complex functional coherence value is shown on the right axis.

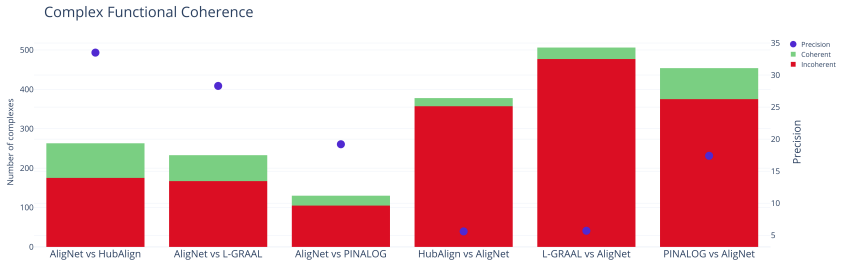


Figure 2.18: Complex Functional Coherence Precision. This figure shows the number of coherent pairs (green) and incoherent pairs (red) obtained with one aligner versus the other.

Table 2.5: Number of not assigned, correctly assigned (CP), incorrectly assigned (NCP) protein complexes and the complex functional coherence value obtained for every aligner.

	AligNet	PINALOG	HubAlign	L-GRAAL
Not Assigned	1269	945	1154	996
CP	128	203	31	37
NCP	377	626	589	741
CFC	25.34	24.48	5	4.75

(coherent pairs) and 175 were incorrectly aligned (by AligNet); and 378 complexes were aligned by HubAlign but not by AligNet, of which 21 were correctly aligned and 357 were incorrectly aligned (by HubAlign). Therefore, HubAlign aligned more complexes than AligNet, but AligNet achieved a higher precision in the alignment of complexes than HubAlign: 33.5% vs 5.6%. Similarly, AligNet also showed a higher precision than L-GRAAL and a slightly higher precision than PINALOG (19.2% vs 17.4%), although PINALOG aligned more complexes than AligNet, which tells us that AligNet is more conservative than PINALOG.

As a third test to evaluate the aligners, we consider the essential proteins, i.e. those proteins that are indispensable for the survival of an organism, again in the Human and

Table 2.6: Numbers of complexes assigned by AligNet and not assigned by the other aligners, and conversely.

	AligNet vs HubAlign	HubAlign vs AligNet	AligNet vs L-GRAAL	L-GRAAL vs AligNet	AligNet vs PINALOG	PINALOG vs AligNet
Not Assigned	891	891	763	763	815	815
Assigned	263	378	233	506	130	454
Incoherent	175	357	167	477	105	375
Coherent	88	21	66	29	25	79
Precision	33.5%	5.6%	28.3%	5.7%	19.2%	17.4%

Yeast PPINs. We evaluate the aligners' performance assuming that essential proteins must be aligned to essential proteins. Thus, for every alignment between the PPINs of *S. cerevisiae* and *H. sapiens*, a true positive (TP) is an essential protein matched to an essential protein while a false positive (FP) is an essential protein matched to a non-essential one. In the same way, a true negative (TN) is a non-essential protein matched to a non-essential one and a false negative (FN) is a non-essential protein matched to an essential one. The essential proteins information was retrieved from the DEG Database (see the subsection 1.4.5 in the Preliminaries chapter). We consider the following statistical measures defined in the Preliminars chapter to evaluate the aligners' performance: *specificity*, *precision*, F_1*-score*, *accuracy*, *MCC (Matthews Correlation Coefficient*. We also consider the *balanced accuracy*, defined by $((TP/P) + (TN/N))/2$, where P and N are the number of essential and non-essential proteins respectively in *S. cerevisiae*, and the *proficiency*, also called uncertainty coefficient or entropy coefficient. The uncertainty coefficient in this test is defined as follows: let $\{p_1, \dots p_n\}$ be the set of proteins in *S. cerevisiae* and let η be an alignment between the two PPINs *S. cerevisiae* and *H. sapiens*. Two random variables X and Y are considered such that, X is a binary vector $X = (x_i)_{i=1,\dots,n}$ such that x_i takes the value 1 if protein p_i is essential and the value 0 otherwise. Y is a binary vector $Y = (y_i)_{i=1,\dots,n}$ such that y_i takes the value 1 if protein $\eta(p_i)$ is essential and the value 0 otherwise. Then, the uncertainty coefficient is defined by,

$$UC = (H(X) - H(X|Y))/H(X)$$

where $H(X)$ is the entropy of X and $H(X|Y)$ is the conditional entropy. In this test, the uncertainty coefficient measures the capability to predict that a *S. cerevisiae* protein is essential provided that its image by η is essential.

In Figure 2.19 we show the values for each statistical measure obtained for every aligner. As we can observe there, all aligners have a similar value of accuracy and balanced accuracy. Concerning specificity, precision and F_1-score, HubAlign obtained the lowest value while the others aligners are comparable. The highest proficiency and MCC values were obtained by AligNet while the lowest one was obtained by PINALOG.

Finally, in order to study the efficiency of the considered aligners, we observed their running time and memory space needed to perform an alignment. We run our implementation of AligNet on a server with 4 processors at 2.6 GHz and 20 GB of RAM, and we also run the latest implementations of PINALOG, SPINAL, HubAlign and also L-GRAAL.

One of the weak points of PPINs aligners is their lack of efficiency. Indeed, although NATALIE [47] was suggested as a good aligner, it could not align the two smallest net-

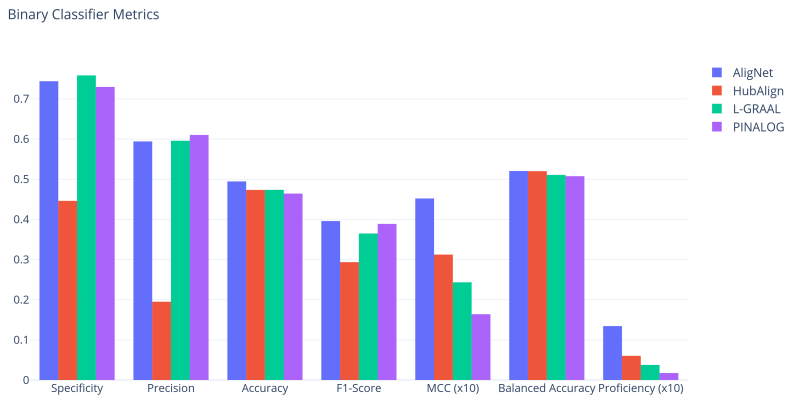


Figure 2.19: Binary Classifier Metrics. This figure shows the results obtained for each aligner in the essential proteins alignment test, for every statistical measure.

works, *C. elegans* and *D. melanogaster*, on a computer with 64 GB of RAM. With respect to PINALOG, SPINAL, HubAlign and L-GRAAL, we were able to complete all the alignments. In order to visualize their running times, we show the running time of every finished computation for each aligner in the top barplot in Figure 2.20. We can observe there that SPINAL is, with a big difference, the slowest one to compute the alignment between *H. sapiens* and *S. cerevisiae*, and also between *D. melanogaster* and *S. cerevisiae*. In addition, PINALOG is the slowest one, also with a big difference, to compute the alignment between *C. elegans* and *H. sapiens*, as well as the alignment between *H. sapiens* and *M. musculus*. We can also observe that AligNet is considerably faster than PINALOG and SPINAL, with a running time of less than 1,000 seconds in most of the alignments. Actually, AligNet is slower than PINALOG and SPINAL only in one computation, namely the alignment between *D. melanogaster* and *H. sapiens*, and the difference is less than 2,000 seconds. However, it is difficult to see the running times in some alignments because SPINAL needed more than 20,000 seconds for the alignment between *S. cerevisiae* and *H. sapiens*. Thus, in order to visualize the results in the cases where the aligners consumed less than 3,500 seconds, we describe in Figure 2.21 the running times cutting at ten minutes. We observe there that HubAlign is the fastest aligner. Thus, we can conclude that HugAlign is the fastest aligner, followed by AligNet.

In Figure 2.20 Figure 2.21 and Figure 2.22, we present the running times ordered by the network's size. It should be expected that the running time increases as so do the networks, and this is the case of AligNet. However, as We can observe in the graphic, this is not the case of L-GRAAL, SPINAL and HubAlign. Actually, L-GRAAL shows an unpredictable running time related with the network's size. We can see that HubAlign is clearly the faster aligner, but the correlation between the network's size and running times is not clear. On the other hand, PINALOG presents a correlation between networks sizes and running times, but it is the slowest aligner. Thus, AligNet is the aligner that present the strongest correlation between running time and networks size.

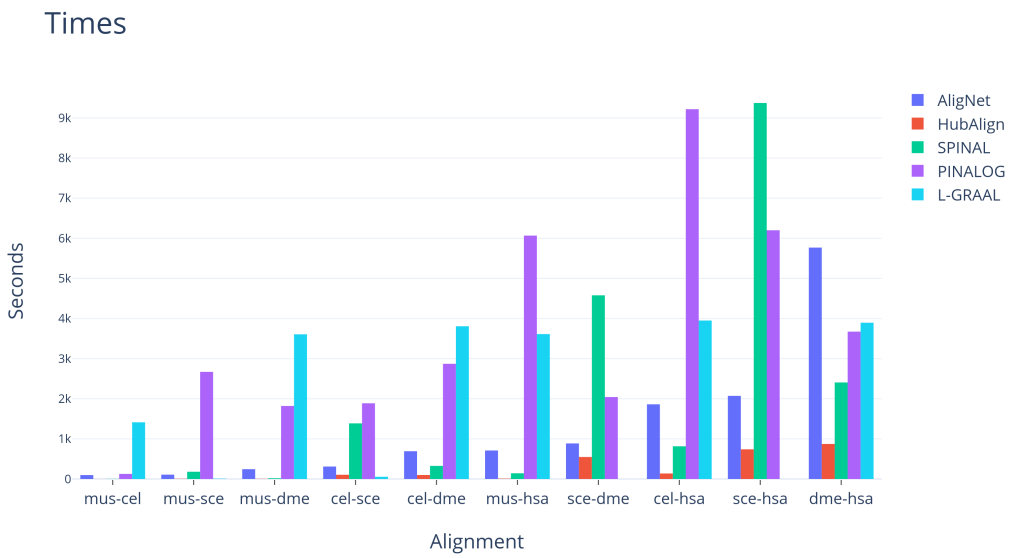


Figure 2.20: Running times. This figure shows the running times (in seconds) we obtained when we performed all the alignments for every pair of the considered networks. In this figure we present the results obtained with the aligners AligNet, PINALOG, SPINAL, HubAlign and L-GRAAL.

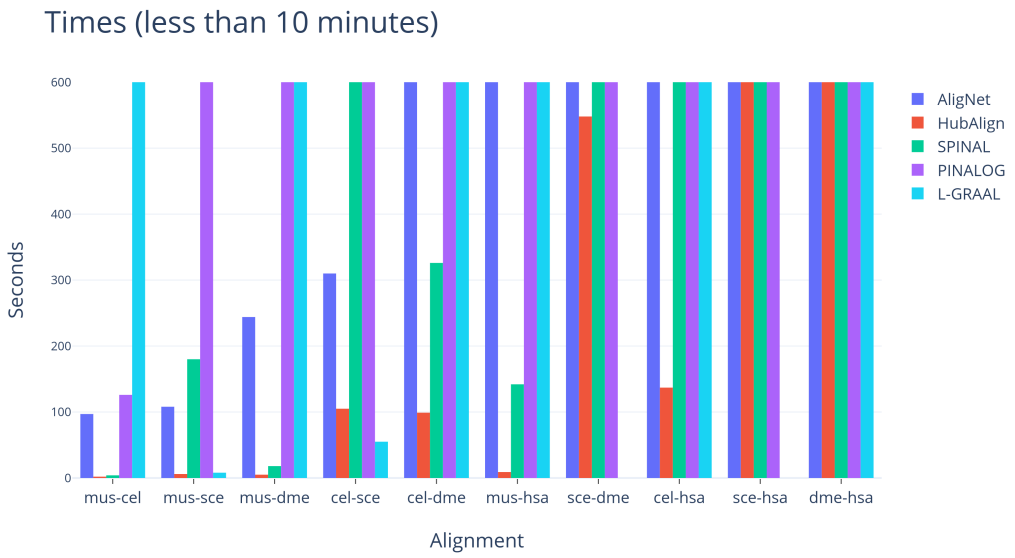


Figure 2.21: Running times cut at 10 minutes. We show in this figure the running times for those alignments that took less than 10 minutes.

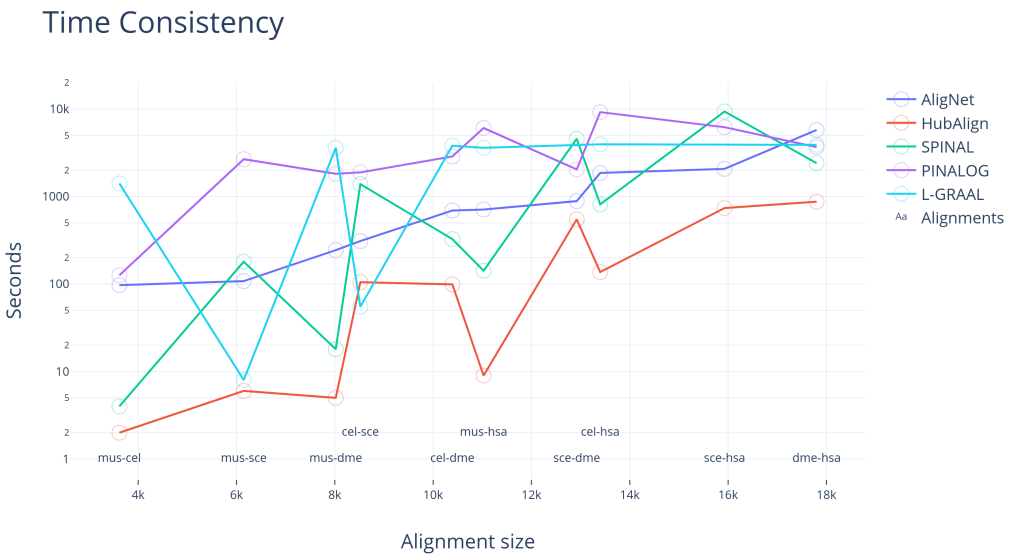


Figure 2.22: Time Consistency. This figure shows the running times in seconds obtained for every pairwise alignment and every aligner. We ordered the pairwise alignments considering the size (number of nodes) of the networks.

CHAPTER 3

PINAWeb

In this chapter, we introduce PINAWeb, a web-server tool to compute PPINs alignments. The main reason to develop this tool is the current lack of a "one size fits all" PPINs aligner, together with the computational problems that arise when installing their implementations. PINAWeb not only facilitates the task of launching the aligners, but it is useful to compare the results between them, for every aligner and pair of networks, PINAWeb returns the corresponding alignment, its topological and biological correctness scores and also the visualization of the alignments' comparison (agreement/differences) when several aligners are considered.

3.1 Architecture

PINAWeb is organized under the philosophy of microservices instead of a monolithic architecture (see subsection 1.5.1 in the Preliminaries chapter). The main benefit of such a design is that each small piece is responsible for only one type of task. In this way, each piece of the tool has no dependencies caused by other pieces, and it is easier to detect errors. As shown in Figure 3.1, PINAWeb's architecture is divided into four main layers: the front-end, the results dashboard, the REST API, and the server. Then, we use connectors to link these layers. The main information flow is the following:

1. The user selects the alignment's options: database, aligner, and networks.
2. The front-end sends a POST request to the REST API to create a job.
3. The REST API enqueues the job to the server using Celery and RabbitMQ as a broker.
4. The server computes the alignment and stores the results in a Mongo database.
5. The server calls back to the REST API with a POST request referencing the inserted result.

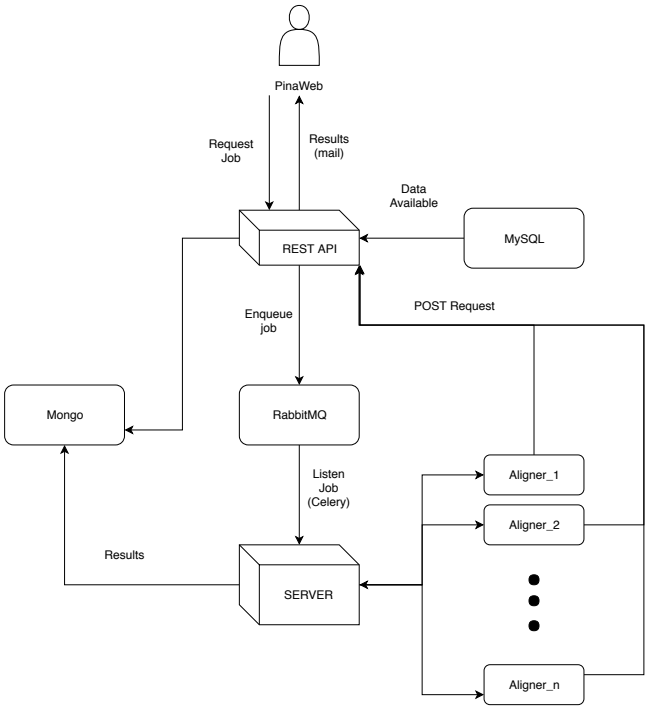


Figure 3.1: Diagram explaining the architecture of PINAWeb.

6. The REST API sends an email to the user with the results dashboard link.

7. The user can see the outcome via the results dashboard.

Front-end web server

The user interacts with PINAWeb through the front-end. This front-end has a layer of css that uses *Bootstrap* to improve the user experience (UX) and user interface (UI), and it uses JavaScript to interact with the REST API through HTTP requests. Its main flow is then:

1. The front-end sends a GET request to the REST API to know which databases and aligners are available, and it displays them at the corresponding interface selector.

2. When the user chooses a database, the front-end asks the REST API which PPINs are available using another GET request, and it displays them at the corresponding interface selector.

3. Finally, once the user has chosen the database, the aligner(s), and the networks, the front-end sends a POST request to the REST API with all this information and the user's email, and it shows a message to the user explaining that an email will be sent with the results.

Dashboard

The outcome from the alignment can be visualized in a dashboard in order to help the user to compare the different alignments. It is implemented in Python 3.9 using the dash framework and deployed using Docker (see subsection 1.5.4 in the Preliminaries chapter). The main flow is:

1. Get the job_id from the path.
2. Request the results data to the REST API.
3. Computes the consensus for each protein.
4. Creates the HTML code to show the consensus and the algorithms results.
5. Returns the HTML.

REST API

The main function of the REST API is to allow the user to create a job that the server will compute, and it is also in charge of sending the results to the user. It is implemented in Python 3.9 with all the libraries needed to create an asynchronous API (aiohttp, gunicorn, UVloop, aiomysql). The REST API has the following routes:

- */database* A GET request to ask which databases are available.
- */networks/{database}* A GET request to ask which networks are available in the chosen database.
- */aligner* A GET request to ask which aligners are available.
- */create-job* A POST request to create a job in Celery (see subsection 1.5.4 in the Preliminaries chapter) which will be executed in the server.
- */finished-job* A POST request to create a task which will send the results of the job to the user.

The REST API uses REDIS (see subsection 1.5.4 in the Preliminaries chapter) to avoid computing the same job several times. So, if a user wants to compute an alignment which has been previously computed, the REST API simply sends the results to the user. Also, if a job has been previously created but still not finished, the REST API adds the email of the user to the mailing list without creating a new job, so that once the alignment is computed, all users who have requested this alignment will receive the results.

Server

The server is in charge of the computation of the alignments. It is implemented using essentially the same technologies as the REST API and works as follows:

1. A Celery worker is listening to a RabbitMQ (see subsection 1.5.4 in the Preliminaries chapter) queue waiting for a new job.

2. When a new job is created, the server gets all the parameters needed to compute the alignment from the job.

3. The server gets all the data needed to compute the alignment.

4. The server creates a system task to compute the alignment. In this way, is not needed to implement all aligners since we can use the binaries available. Given that all the architecture is implemented under Docker, if in the future we need any special system requirements to execute an aligner, we can easily add them to our system.

5. Once the alignment is computed, this is stored in the Mongo database (see subsection 1.5.4 in the Preliminaries chapter) and a POST request is sent to the REST API.

3.2 Results

In this section, we present the tool PINAWeb and present the results of several tests to discuss the tool's usability. In order to explain how to use PINAWeb as well as the tool's functionalities, we display here a query and its output as a running example. The query is to obtain the alignment for every aligner (AligNet, HubAlign, LGRAAL, PINALOG and SPINAL, see the subsection 1.4.4 in the Preliminaries chapter) of two PPINs. As the input network, we considered a set of proteins from the Saccaromises scerevisiae and created a network. As the output network, we selected the Homo sapiens network from the STRING database, and we considered as edges the *Experimental score* with a threshold value of 800. PINAWeb computes the alignment of the input network to the output network for every selected aligner. When the computations are finished, the user receives an email with a link to retrieve and analyze the obtained results. The link to the tool's webpage is `https://biocom.uib.es/pinaweb/`.

PINAWeb query webpage

Figure 3.2 shows PINAWeb's query webpage whith the running example introduced as a query. The query webpage consists of different boxes where the user selects the aligners and introduce the input and the output networks, as well as the email address where the results are sent. The information and usability of every box are explained below.

Database: PINAWeb retrieves from the STRING database (see the subsection 1.4.5 in the Preliminaries chapter) the networks' information as well as all the information needed to compute the requested alignments. Namely, the sequence similarity of every pair of proteins and the GO terms of every annotated protein. Therefore, the STRING database must be selected in this box (see Figure 3.3).

Aligner: The available aligners are AligNet, HubAlign, L-GRAAL, PINALOG and SPINAL. The user can select all, only one, or several of them. The selected aligners are then displayed in the toolbox (see Figure 3.4).

E-mail: The user has to provide an E-mail address where the tool sends the links to the results of every query.

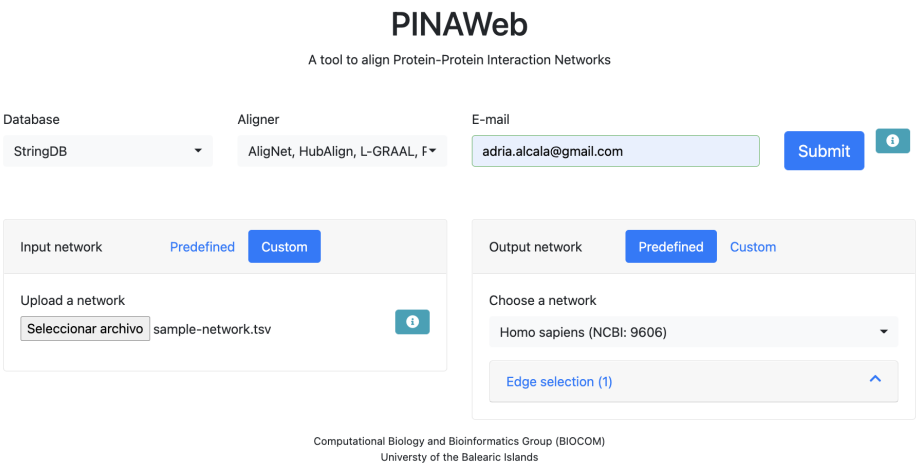


Figure 3.2: PINAWeb's query example.

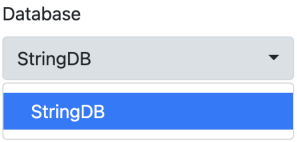


Figure 3.3: PINAWeb's database selection.

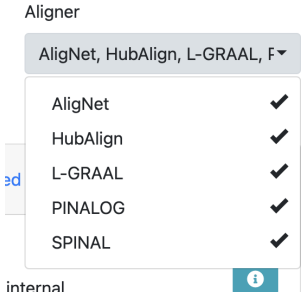


Figure 3.4: PINAWeb's aligners selection.

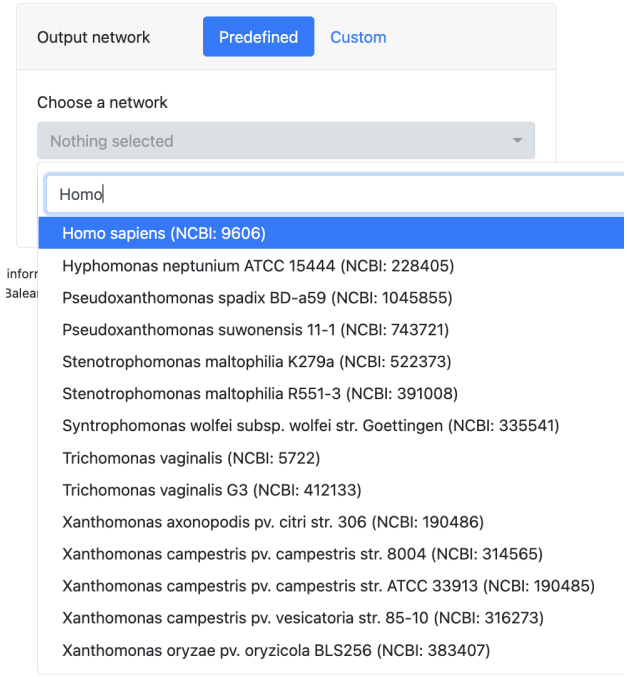


Figure 3.5: PINAWeb's species selection.

Input network: In this box, the input network of the query must be introduced or selected. In our running example, it is a personalized network. To enter a personalized network, click the button *Custom*. Then, it appears the option to upload a network from the user's computer. In the information button on the right within this box, it is explained the accepted format of the network to upload, as well as two examples of custom networks.

Output network: In this box, the output network of the query must be introduced or selected. In our running example, it is the *Homo sapiens* PPIN from the STRING database. To enter the network, the user must select the button *Predefined* and then look for the species name. To facilitate this search, the user can write the first letters of the species name to reduce the searching dataset (see Figure 3.5). Next, in the *Edge selection* button, the user can choose all the edge types (score types) from the STRING database. For every selected edge type, a threshold must be introduced, and only the edges type whose weight is above the selected threshold in the STRING database will be included in the network Figure 3.6.

Submit: Once the user submit the job clicking on the submit button, a message is show to inform the job identifier (see Figure 3.7).

PINAWeb output-results

Once the requested alignments have been completed, PINAWeb sends an email to the address introduced in the query webpage with the job identifier. In this email, the user will find a link to a webpage with the results. On top of this PINAWeb Results webpage, the

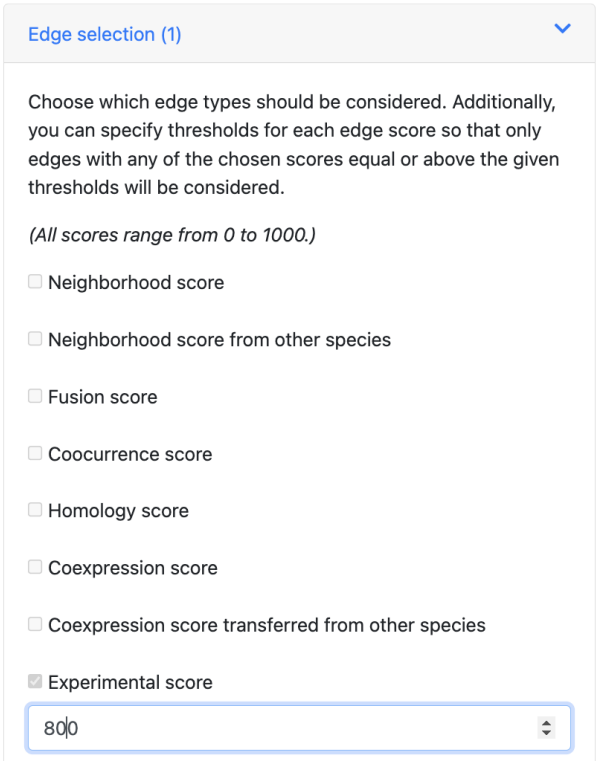


Figure 3.6: PINAWeb's scores selection.

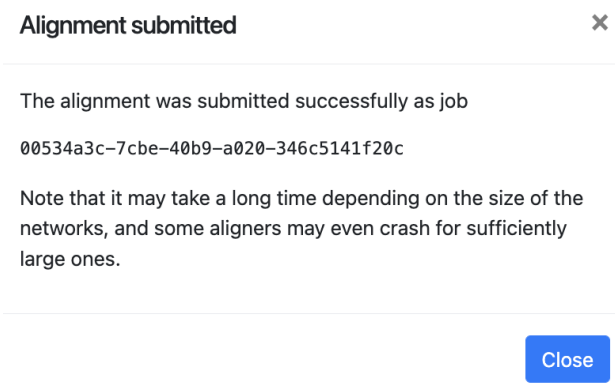
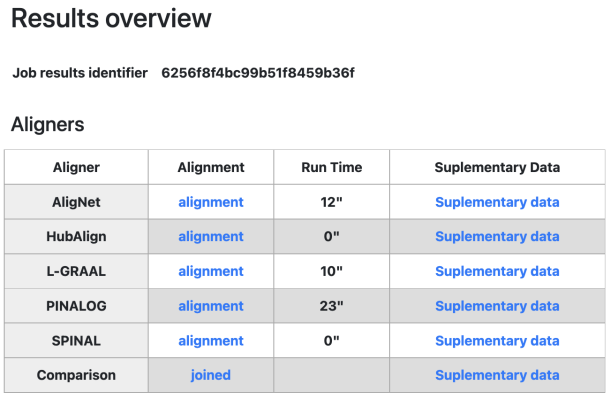


Figure 3.7: PINAWeb's job submitted.



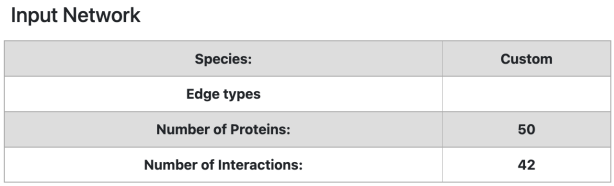
Results overview

Job results identifier 6256f8f4bc99b51f8459b36f

Aligners

Aligner	Alignment	Run Time	Suplementary Data
AligNet	alignment	12"	Suplementary data
HubAlign	alignment	0"	Suplementary data
L-GRAAL	alignment	10"	Suplementary data
PINALOG	alignment	23"	Suplementary data
SPINAL	alignment	0"	Suplementary data
Comparison	joined		Suplementary data

Figure 3.8: Aligners finished.



Input Network

Species:	Custom
Edge types	
Number of Proteins:	50
Number of Interactions:	42

Figure 3.9: Custom network in results web.

information is split in two menus. In the *Alignments info* menu, all the single information of the requested alignments is provided (see Figure 3.13). Namely, the information is classified in the following sections:

- **Aligners**: This table shows the aligner(s) for each alignment (see Figure 3.8). In the second column, there is a link to the alignment results in a tsv file format. In the third column, the run time in seconds is displayed for each aligner. In the last column, there is a link to a json file with all supplementary data of every alignment process. The json file also contains the unaligned nodes and edges, the non-preserved nodes and edges and the annotated proteins. In the last row of the table, when more than one aligner was selected, PINAWeb provides a tsv file that merges the result of all requested alignments. Also, the corresponding json with all supplementary information is provided.

- **Input Network/Output Network**: The input and output networks requested by the user are shown in these two tables, as well as their numbers of nodes and edges. In case that the network is a custom network then the species is identified as *Custom*, Figure 3.9, on the other hand, if the network is a predefined network then the species and the scores are displayed (see Figure 3.10).

- **Alignment(s) quality measures**: PINAWeb summarizes the quality measures through a barplot diagram. The measures are: the *edge correctness ratio* (see Figure 3.11), that quantifies the amount of structure preserved by every alignment, and the two functional coherence values (see Figure 3.12), using the *Jaccard score* and the

Output Network

Species:	Homo sapiens
Edge types	experimental_score:800
Number of Proteins:	1824
Number of Interactions:	4358

Figure 3.10: Predefined network in results web.

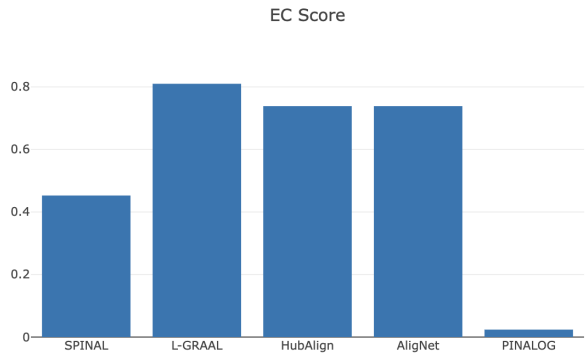


Figure 3.11: EC scores.

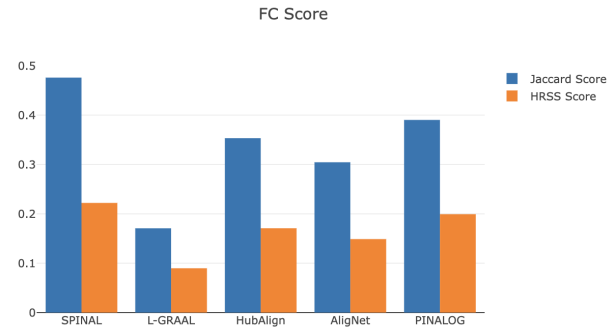


Figure 3.12: FC scores.

HRSS score which assess the functional similarity of the aligned proteins by comparing their *Gene Ontology annotation* (see subsection 1.4.3 in the Preliminaries chapter).

When more than one aligner is selected, in the *comparison* menu, PINAWeb provides a heatmap to visualize the comparison of the alignments' results (see Figure 3.14). More precisely, every selected aligner is displayed on the left in the heatmap and all proteins from the input network are listed as columns. The user can select the number of proteins (columns) to visualize in the heatmap. The color corresponding to unaligned proteins is gray. The agreement level between the aligned proteins is represented through a color palette that ranges from light green to dark green. Dark green indicates a total agreement, meaning that all aligners have matched the protein to the same protein in the output

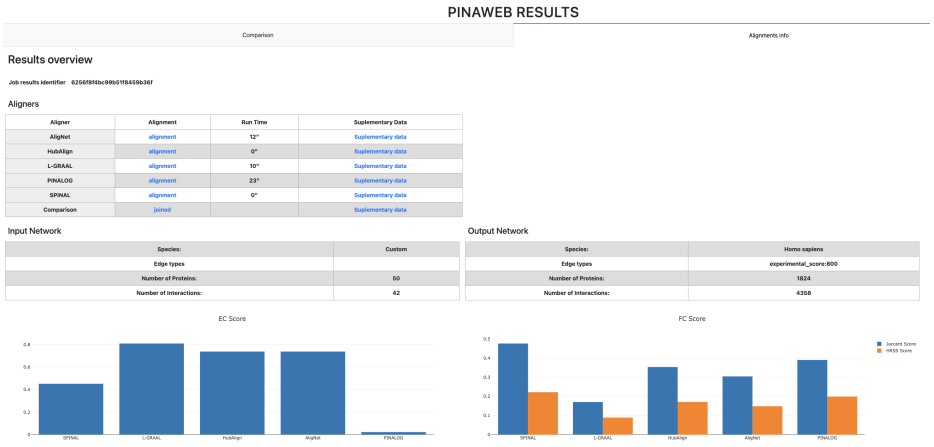


Figure 3.13: This figure displays the alignments' information provided by PINAWeb. The quality measures of every alignment are presented in a barplot.

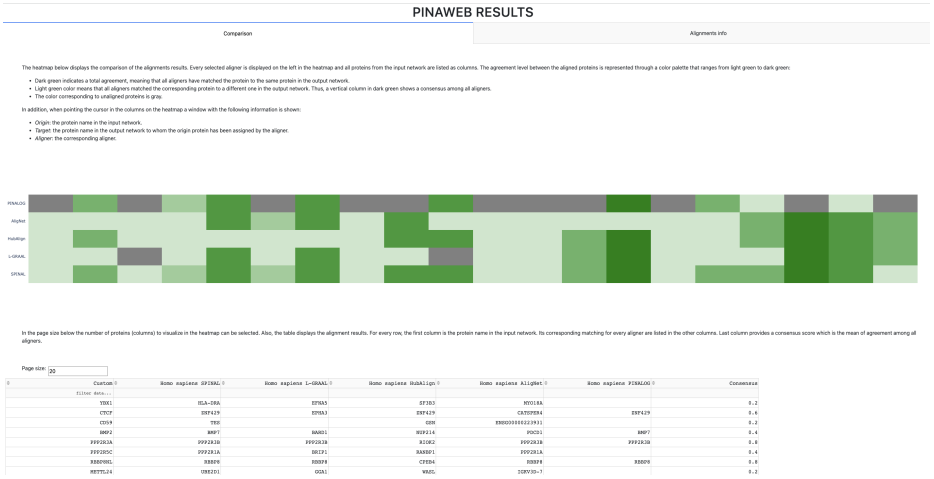


Figure 3.14: This figure shows the heatmap provided to compare the aligners results. The table displayed below the heatmap shows the matchings information.

network. Light green color means that all aligners matched the corresponding protein to a different one in the output network. Thus, a vertical column in dark green shows a consensus among all aligners. In addition, when pointing the cursor in the columns on the heatmap (see Figure 3.15) a window with the following information is shown: *Origin:* The protein name in the input network, *Target:* The protein name in the output network where the origin protein has been assigned by the aligner also displayed in the window.

Experiments

We report here the experiments designed in order to show the tools' usability. As a first experiment, we considered the PPINs of the organisms used in [16] which consists of the PPINs of *M. musculus* (mus), *C. elegans* (cel), *D. melanogaster* (dme), *S. cerevisiae*

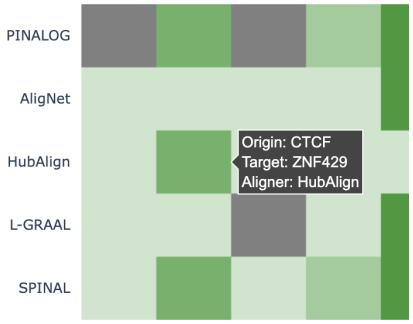


Figure 3.15: Heatmap detail.

(sce), and *H. sapiens* (hsa). The PPINs were selected from the STRING database with an experimental score of 800 as edge selection. In Table 3.1 we provide the networks' information regarding their number of nodes and edges.

	Nodes	Edges
C. elegans	509	1,149
M. musculus	751	1,443
D. melanogaster	1,224	1,933
H. sapiens	1,824	4,358
S. cerevisiae	2,775	13,269

Table 3.1: Number of nodes and edges of the PPINs considered as input data in our first experiment.

We requested the alignments for every pair of PPINs and each aligner. Thus, we ended up with 50 alignments. As a result, PINAWeb performed all alignments and provided all the alignments' information. We show the running time of every computation and each aligner in Table 3.2 below.

	AligNet	HubAlign	L-GRAAL	PINALOG	SPINAL
cel-mus	33	0	122	8	10
cel-dme	32	0	318	9	10
cel-hsa	81	1	684	27	18
cel-sce	65	1	1357	404	77
mus-dme	54	1	531	10	4
mus-hsa	171	2	964	27	13
mus-sce	92	2	2606	446	59
dme-hsa	127	6	2195	37	20
dme-sce	108	6	3602	446	35
hsa-sce	230	25	3604	519	106

Table 3.2: Time in seconds of every alignment computation.

As we can observe there, HubAlign is the fastest aligner followed by SPINAL, while L-GRAAL is the slowest. AligNet performed each computation in less than 4 minutes and PINALOG was faster than AligNet in most computations except those with S. cerevisiae, probably due to its large number of edges. Nevertheless, in one hour PINAWeb

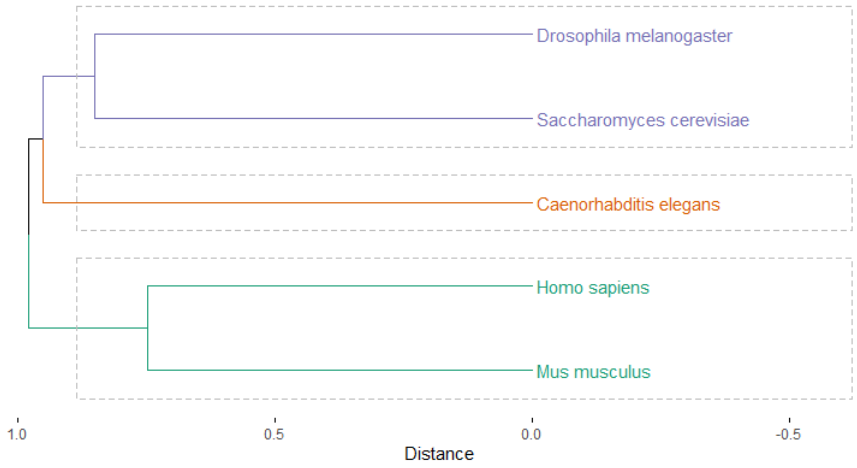


Figure 3.16: Hierarchical clustering of the consensus alignment score in the first experiment.

reported all the requested alignments, which reinforces the correct election of the aligners.

Regarding the alignments themselves, the highest quality measures were obtained in the alignment of M. musculus and H. sapiens. This may reveal that, as expected, when the networks to be aligned are similar (closer species), the aligners perform better. Indeed, Figure 3.16 shows the hierarchical clustering considering the euclidean distance and complete linkage of the consensus alignment scores. We can observe that the cluster of M. musculus and H. sapiens is clearly separated from the other species.

As a second experiment, in order to evaluate the consensus information provided by PINAWeb, we designed the following test: we considered the same PPIN of H. sapiens as in the previous test and we randomly deleted some nodes and edges. Thus, we synthetically created two PPINs as subnetworks of the H. sapiens PPIN such that they have 50 and 55 proteins respectively, and they share 37 of them. The networks have 42 and 48 edges, respectively. Then, we requested the alignment of this pair of subnetworks for every aligner. As a result, we obtained that, except PINALOG, all aligners properly matched the shared proteins. That is, all aligners mapped every protein present in both networks to itself. The consensus alignment in this case was 0.8. Also, the edge correctness ratio and the Jaccard functional coherence value were above 0.8. Therefore, we conclude that the aligners obtained a considerable agreement in this test.

In the table below we provide the links to all the alignments and comparison results obtained in the experiments reported here.

cel-mus	https://biocom.uib.es/util-aligner-results/6255773fbc99b51f8459b22c
cel-dme	https://biocom.uib.es/util-aligner-results/62557873bc99b51f8459b243
cel-hsa	https://biocom.uib.es/util-aligner-results/62557a47bc99b51f8459b25a
cel-sce	https://biocom.uib.es/util-aligner-results/62557dcdbc99b51f8459b271
mus-dme	https://biocom.uib.es/util-aligner-results/625589babc99b51f8459b2cd
mus-hsa	https://biocom.uib.es/util-aligner-results/62558339bc99b51f8459b288
mus-sce	https://biocom.uib.es/util-aligner-results/625584f1bc99b51f8459b29f
dme-hsa	https://biocom.uib.es/util-aligner-results/6255894cbc99b51f8459b2b6
dme-sce	https://biocom.uib.es/util-aligner-results/62559268bc99b51f8459b2fb
hsa-sce	https://biocom.uib.es/util-aligner-results/62558b0ebc99b51f8459b2e4
Test 2	https://biocom.uib.es/util-aligner-results/6256e011bc99b51f8459b341

Table 3.3: Link to the results obtained in the tests.

CHAPTER 4

Prots2Net: a PPIN predictor of a proteome or a metaproteome sample

In this chapter, we present *Prots2Net*, a tool to predict a PPIN from a metaproteome or a proteome sample. Given a set of protein sequences, Prots2Net predicts its PPIN (see the subsection 1.4.2 in the Preliminaries chapter.) by considering only the sequence similarity between the input proteins and the proteins in a selected PPINs from the STRING database (see the subsection 1.4.5 in the Preliminaries chapter).

4.1 The structure of Prots2Net

The workflow of Prots2Net consists of the following steps detailed below, as shown in Figure 4.1:

1. Input data.

2. Network prediction.

3. Visualize data.

In the first step of Prots2Net, the user inserts the proteins sequences whose PPIN is required. In the second step, Prots2Net predicts the requested PPIN. And finally, the user can visualize and analyze the network in the third step. Besides this workflow, the code is organized in 2 main components: A GUI (see subsection 1.5.2 in the Preliminaries chapter), where the user interacts, and a backend server, where all the computations are performed.

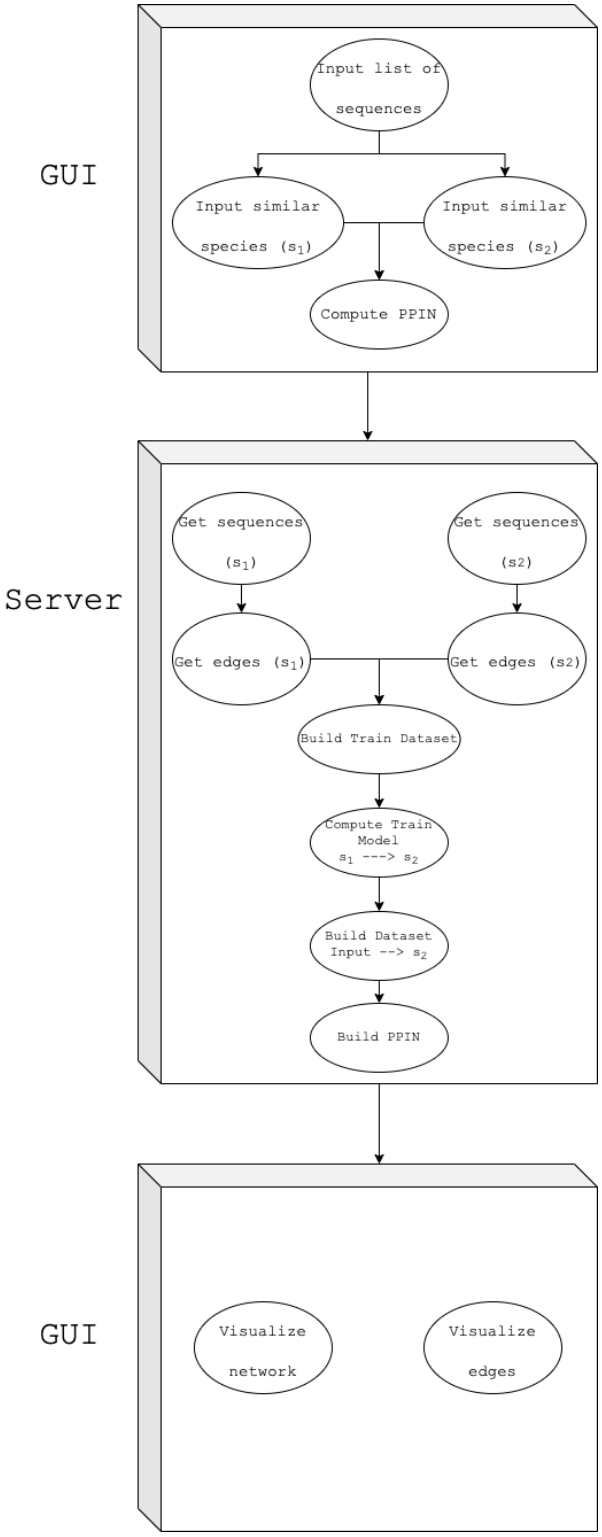


Figure 4.1: Diagram explaining the workflow of Prots2Net.

4.1.1 Graphical User Interface (GUI)

This GUI (see Figure 4.2) is implemented in Python and use tkinter (see subsection 1.5.3 in the Preliminaries chapter) libraries to allow the users to interact with the application. This interface has 4 different tabs:

1. A tab to insert the proteins sequences.
2. A tab to visualize the predicted network.
3. A tab to obtain the list of nodes (proteins).
4. A tab to obtain the list of edges (interactions).

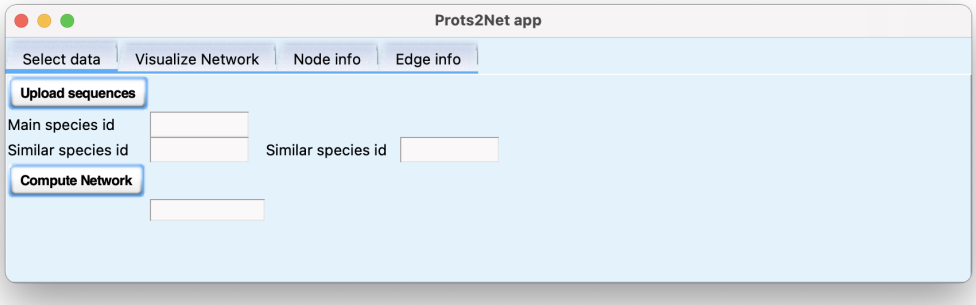


Figure 4.2: GUI of Prots2Net.

Insert data tab

In this tab, the protein sequences are inserted to predict their PPIN. This tab is composed by the following boxes:

- A button to select a file with the input data.
- Three text boxes to introduce the species identifiers.
- A button to compute the network.

The user must click the *Upload sequences* button (see Figure 4.3) to select a file with the protein sequences in FASTA format (see subsection 1.4.1 in the Preliminaries chapter), using a file selector (see Figure 4.4).

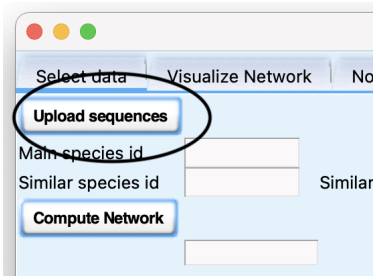


Figure 4.3: Upload sequences button.

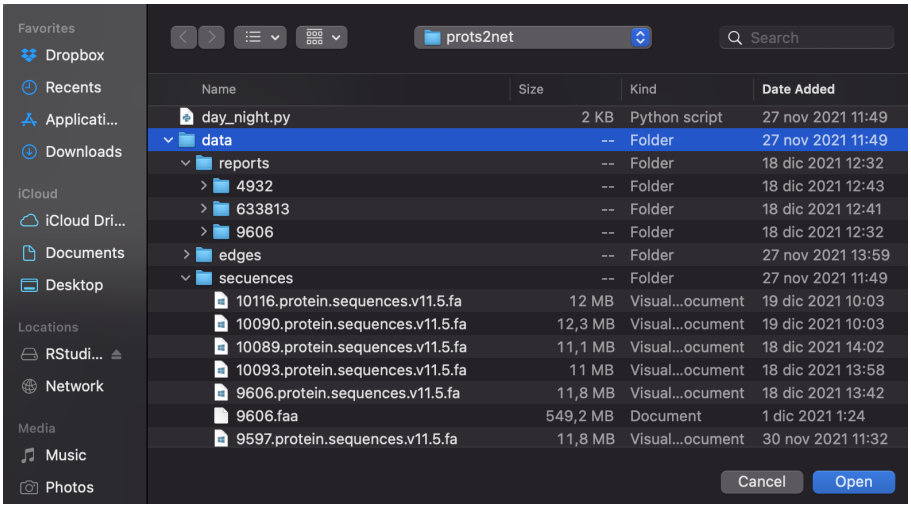


Figure 4.4: File selector.

Once the user has uploaded the protein sequences, an identifier for the main species must be introduced (see Figure 4.5). Next, the user must select 2 similar species as shown in Figure 4.6 and finally click the COMPUTE NETWORK button to compute the PPIN (see Figure 4.7).

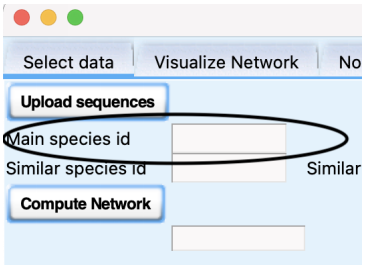


Figure 4.5: Main species text box.

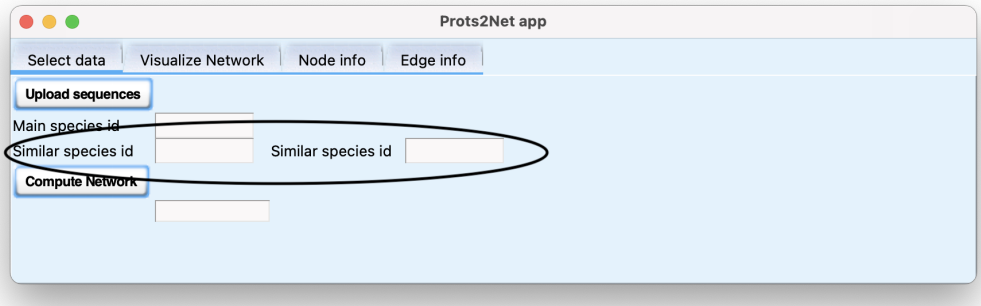


Figure 4.6: Similar species text box.

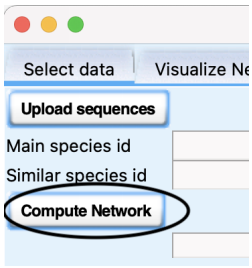


Figure 4.7: Compute network button.

Visualize network tab

The user can visualize the predicted network by clicking on the *Plot graph* button (Figure 4.8) and selecting the predicted network file. The image file is stored in the relative path $data/graphs$. This visualization is useful when the network is not so huge, as one can see in Figure 4.9 and Figure 4.10.

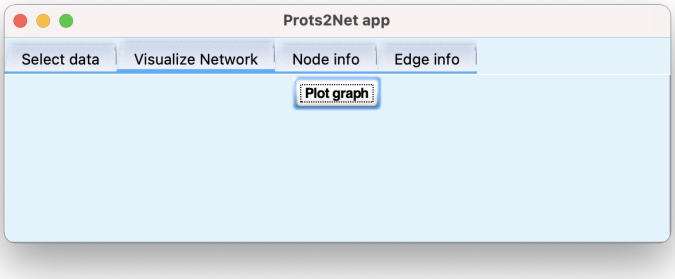


Figure 4.8: Plot Graph button.

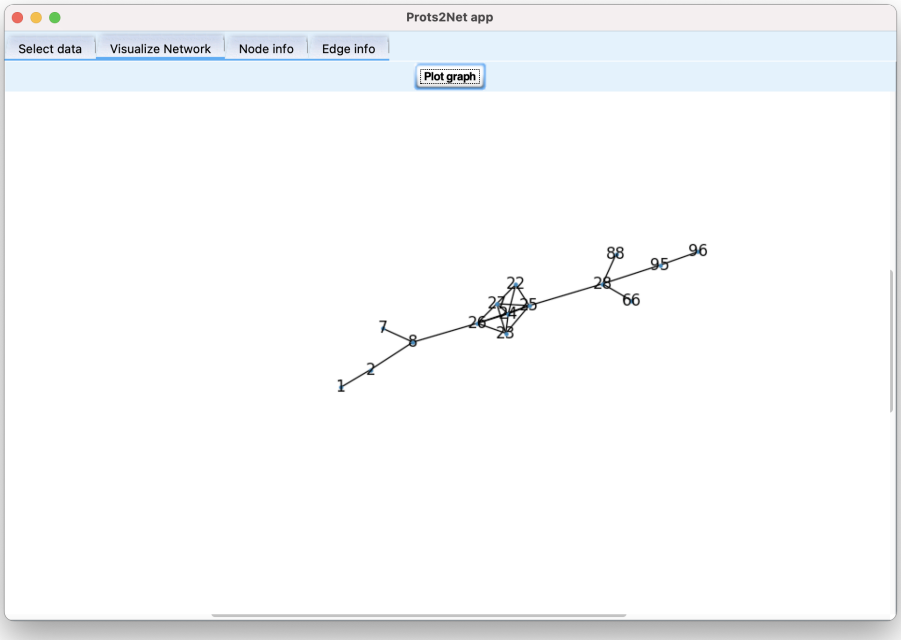


Figure 4.9: Small network visualization in Prots2Net app.

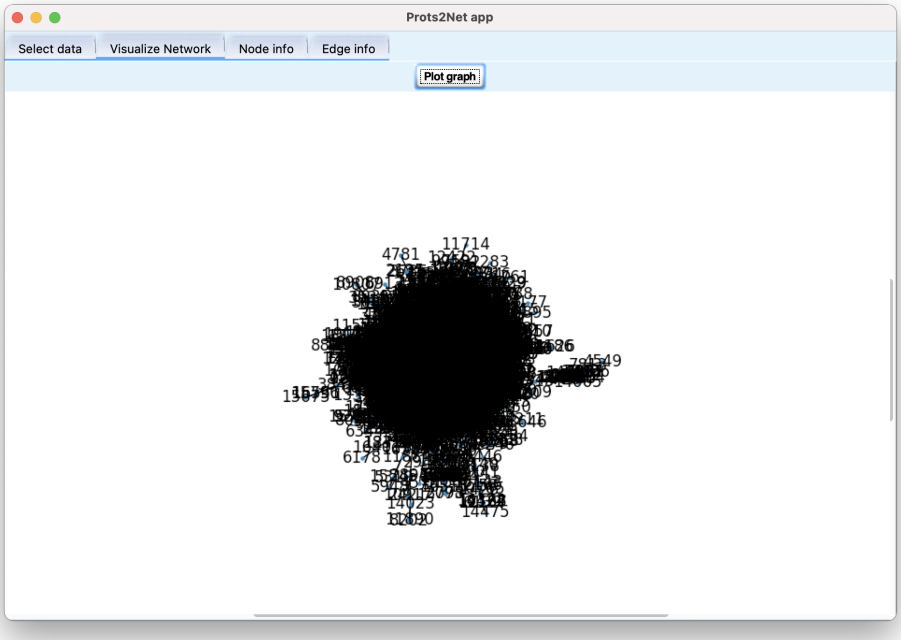


Figure 4.10: Huge network visualization in Prots2Net app.

Node and edge info tabs

The list of nodes and edges of the predicted network are shown in the following tabs (see Figure 4.11 and Figure 4.12).

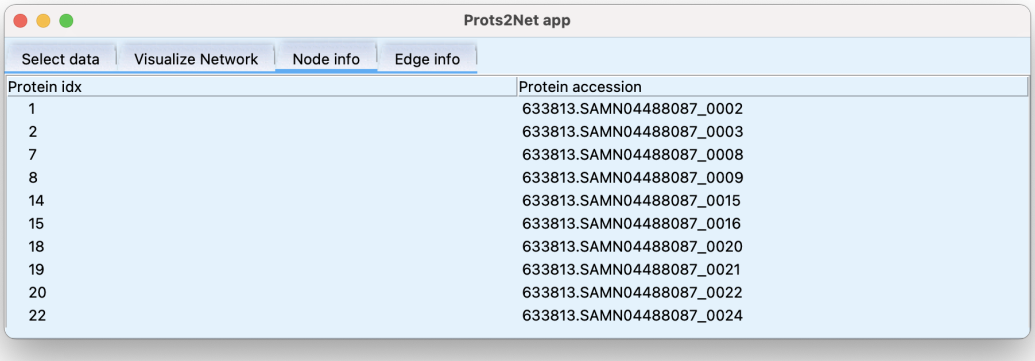


Figure 4.11: List of nodes.

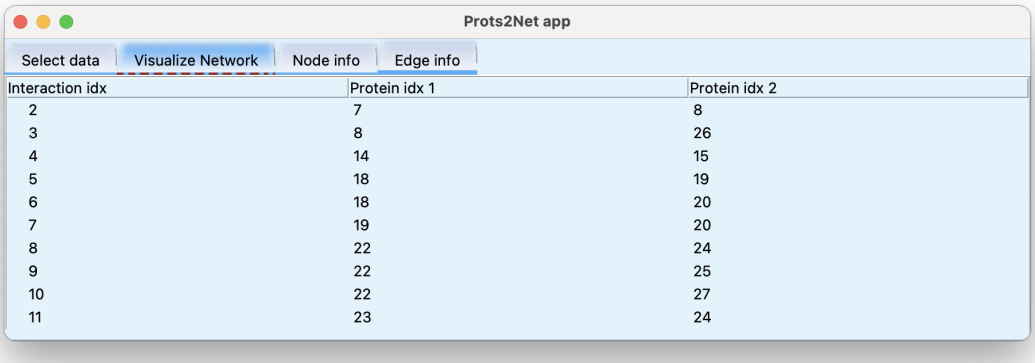


Figure 4.12: List of edges.

4.1.2 Server

The backend server carries on all the computations needed to obtain the PPIs predictions. The workflow of the entire job is as follows: first compute a training model from the two selected species. Then, read the proteins sequences in the input file. And finally, predict the output PPIN using the computed training model.

The overall idea behind this model and its novelty is that, two proteins interact if their orthologous proteins, in another species whose PPIN is known, interact. Thus, to train the model, the user must select two species from the STRING database. Next,

the PPIs information from the first species is transferred to the second species, through orthologous proteins, to predict the interactions. Finally, the PPIs information from the second species is used as the true output during training. We explain below the details of the whole process.

Training model dataset

The training model is constructed using the library sklearn (see subsection 1.5.3 in the Preliminaries chapter) as follows: let s_1 and s_2 be the two species from the STRING database, and let P_1 and P_2 be the corresponding set of proteins of s_1 and s_2, respectively. For every pair of proteins, p_{21} and p_{22} in P_2, Prots2Net takes into account the following information as input to build the training dataset:

- The bitscore b_1 of the sequence alignment between the protein p_{21} and its most similar protein p_{11} in P_1.
- The bitscore b_2 of the sequence alignment between the protein p_{22} and its most similar protein p_{12} in P_1.
- The combined interaction score in s_1 between p_{11} and p_{12} when they interact. If there is no interaction between p_{11} and p_{12}, it is set to 0.

The true output during training, is 1 i.e., p_{21} and p_{22} are considered to interact, if the combined interaction score in s_2 between p_{21} and p_{22} is greater than 999. Otherwise, the output is 0 and it is considered that p_{21} and p_{22} do not interact.

Neural Network Model

The neural network model used in this tool is an MLP (multilayer perceptron neural network) with 2 hidden layers (see subsection 1.2.2 in the Preliminaries chapter). As we can see in Figure 4.13, the input layer has 3 input values which are the values of b_1 and b_2 defined in the training dataset, but considering in this step the pair of proteins p_{21} and p_{22} from the user's input set, and the combined interaction score in s_1 when p_{11} and p_{12} interact, or 0 otherwise. The output layer has only one value, since we are only considering one class, that indicates if p_{21} and p_{22} interact. As a set of default parameters, we considered those listed below, though all of them can be easily changed in the code:

- the activation function used is $ReLU$.
- the loss function is binary cross-entropy.
- the cost function is the binary cross-entropy cost.
- the optimization algorithm used is $ADAM$.
- the regularization method is the L2 penalty, and it is set to 10^{-6}.
- the tolerance is set to 10^{-8}, that is, the minimal improvement between steps to continue the training.
- the maximum number of iterations is 30000.

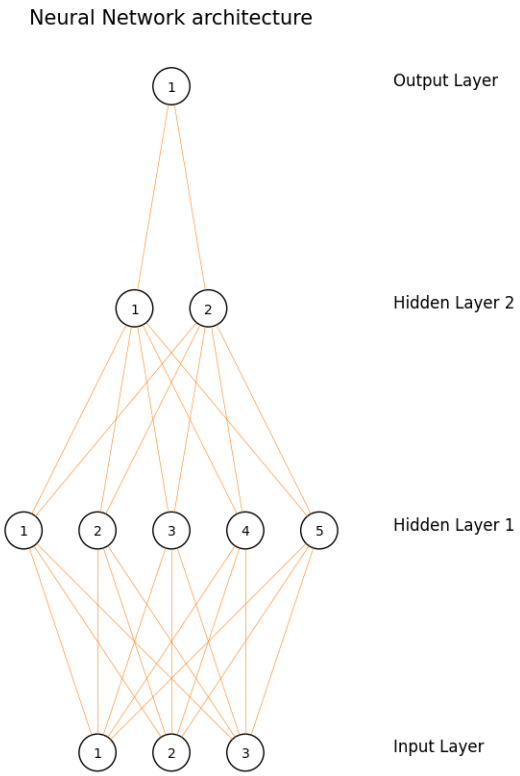


Figure 4.13: Neural network architecture.

We refer to section 1.2 in the Preliminaries chapter for a brief description of each parameter.

Reused data

Generated data for every process is stored in the server under the *data* directory considering the species identifiers. Thus, the following relative paths are created:

- *sequences*, a folder that contains all the sequences files.
- *blast*, a folder that contains the results of the blast computations. As an example, if the user's main species identifier is *main-species* and *species1* and *species2* are the selected two similar species. Then, the stored files are *blast/result_species1_species2.csv* and *blast/result_main-species_species2.csv*.
- *edges*, a folder that contains the edge list of every network used in the training step. Notice that these files are the list of edges downloaded from the STRING database under the edge score threshold.

- *nets*, a folder that contains all predicted networks. For instance, in the previous example, the stored file is *nets/Net_main-species_species2.csv*.
- *graphs*, a folder that contains the images to visualize all predicted networks. Following the previous example, the stored is *data/graphs/Net_main-species_species2.png*.
- *models*, a folder that contains the trained models. Following the previous example, the stored file is *models/Model_species1_species2.pickle*.

Since all stored files have their names referred to the species identifiers, all generated models are reused when the selected species have been already considered. Hence, we recommend the user to take this into account when choosing the identifiers names.

4.2 Prots2Net evaluation

In order to evaluate Prots2Net, and compare our tool with other existing tools, we performed three tests to predict the PPIs in species from different kingdoms. The species that we considered are the following:

1. to predict the PPIs in the Rhodothermus profundi (NCBI 633813), we selected Rhodothermus marinus (NCBI 518766) and Salinibacter ruber (NCBI 309807) as similar species.
2. to predict the PPIs in Saccharomyces cerevisiae (NCBI 4932), we selected Millerozyma farinosa (NCBI 4920) and Naumovozyma dairenensis (NCBI 27289) as similar species.
3. to predict the PPIs in Homo sapiens (NCBI 9606), we selected Pan paniscus (NCBI 9597) and Gorilla gorilla (NCBI 9593) as similar species.

Notice that, in every test we selected, as similar species, two species that are taxonomically close to the species whose PPIs are predicted. More precisely, in the first test, we consider as similar species another two bacteroidetes. In the second test, we selected as similar species another two fungi and, in the third test, we considered another two hominidae.

As explained in the previous section, we considered as parameters, a multilayer perceptron classifier (MLP) model with two hidden layers with 5 and 2 nodes respectively and the configuration explained there (see Neural Network Model subsection). Then, for every test, we applied Prots2Net to the input protein sequences considering the mentioned similar species.

4.2.1 Evaluation measures

To validate the proposed model, we considered the following error measures defined in the Preliminaries section (1.3): accuracy, precision, MCC, sensitivity, specificity, fall out, F1 score. Where now TN is the number of true negatives, i.e., the non-interacting proteins that are predicted correctly; TP is the amount of true positives, i.e., the interacting proteins that are predicted correctly; FN is the number of false negatives, i.e.,

the interacting proteins that are predicted to be non-interacting; and FP is the amount of false positives, i.e., the non-interacting proteins that are predicted to interact. Additionally, the receiver operating characteristic (ROC) curves and the area under the ROC curve (AUC) were also calculated to further evaluate the discriminatory accuracy of the proposed model. In order to obtain a highly reliable dataset as true PPINs, we considered the corresponding PPINs from the STRING database with a combined interaction score threshold of 999. That is, there is a true interaction between two proteins when their combined interaction score is equal or greater to 999.

4.2.2 Test 1. Rhodothermus profundi predictions

To predict the interactions of Rhodothermus profundi, we considered as similar species to train the model, Rhodothermus marinus and Salinibacter ruber. Prots2Net obtained the results displayed in Figure 4.14. We can observe that, the *precision* and *accuracy* values are very high, almost 1, and the *Fall out* values are nearly 0. This is a consequence of the fact that the number of false positives is almost 0, as we can observe in Figure 4.15. Indeed, we can observe in the confusion matrix that there is only 1 false positive, when the threshold is 0.5, i.e., there is only one interaction that Prots2Net predicts, and it is wrong. Notice that, even with this high precision, there is no loss of accuracy, which is higher than 0.91. Figure 4.16 also shows this excellent performance. We can observe the ROC curve with a high AUC value of 0.9261.

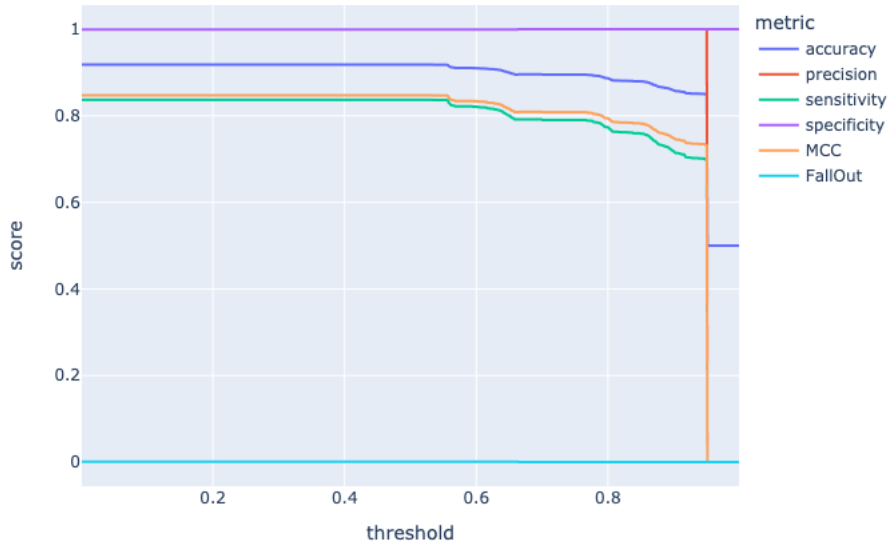


Figure 4.14: Error measures for Rhodothermus profundi.

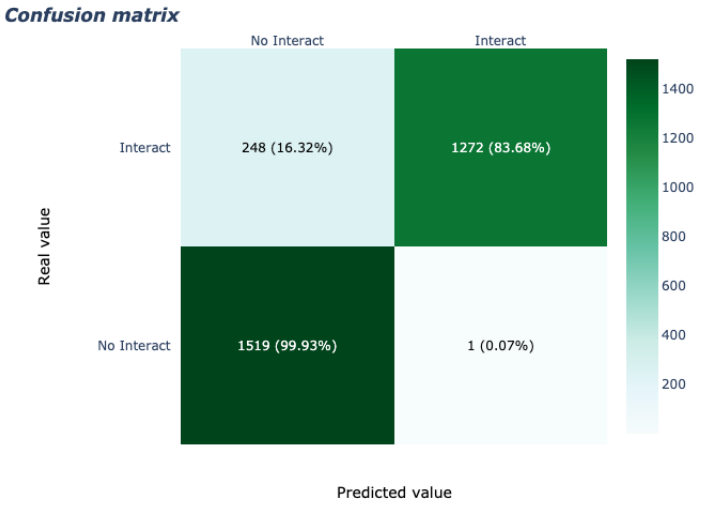


Figure 4.15: Confusion matrix for Rhodothermus profundi with a threshold of 0.5.

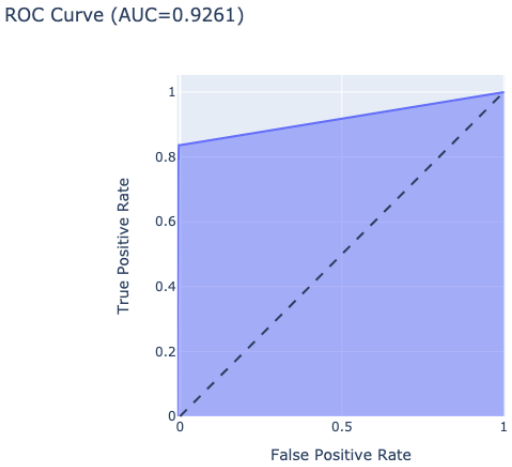


Figure 4.16: ROC for Rhodothermus profundi.

4.2.3 Test 2. Saccharomyces cerevisiae predictions

To predict the interactions of Saccharomyces cerevisiae, we considered as similar species to train the model, Millerozyma farinosa and Naumovozyma dairenensis. Prots2Net obtained the results displayed in Figure 4.17. We can observe that in this test, Prots2Net performed even better than in the previous test. Indeed, all the positive metrics (accuracy, precision, sensitivity, specificity, MCC) reached values higher than 0.9, except in the extreme values of the threshold. Such a good behavior is also reflected in Figure 4.18 where we show the confusion matrix obtained with a threshold of 0.5. We can see that

only 6 interactions were wrongly predicted by Prots2Net. Finally, we observe again the good performance of Prots2Net in Figure 4.19 where we show the ROC curve with a AUC value of 0.9811.

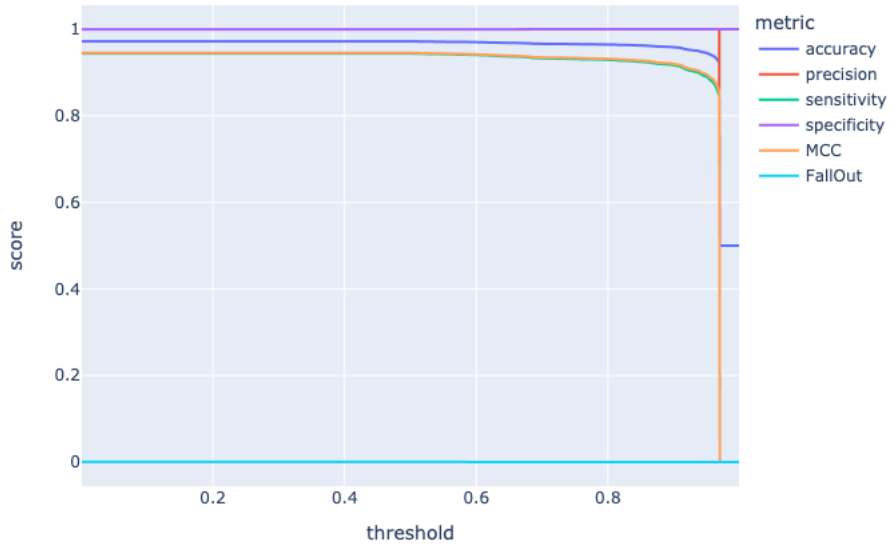


Figure 4.17: Metrics for Saccharomyces cerevisiae.

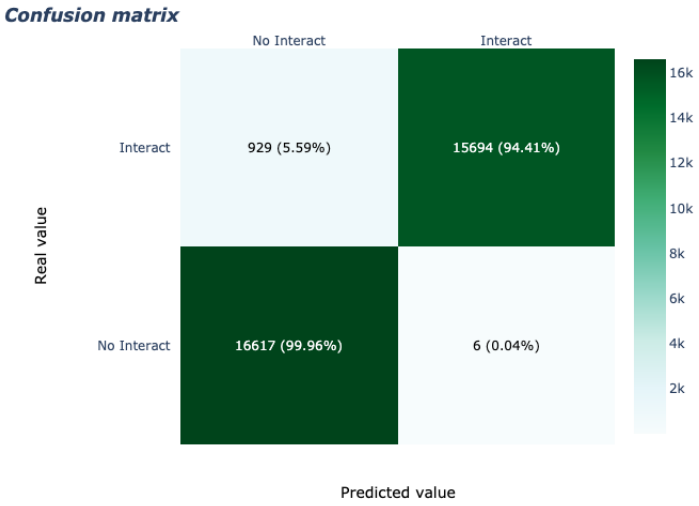


Figure 4.18: Confusion matrix for Saccharomyces cerevisiae with a threshold of 0.5.

Method	Acc	Sen	Pre	MCC
Prots2Net	97.18	94.41	99.96	94.52
DeepPPI [18]	94.43	92.06	96.65	88.97
RF + LPQ [97]	93.92	91.10	96.45	88.56
Bio2Vec [96]	93.30	92.70	93.55	87.49
MCD + SVM [102]	91.36	90.67	91.94	84.21
LCPSSMMF [6]	90.48	90.26	90.58	82.84
3-mers [96]	90.26	88.14	91.65	82.38
ACC [32]	89.33	89.93	88.87	N/A
LD [104]	88.56	87.37	89.5	77.15
AC [32]	87.36	87.30	87.82	N/A
PCA-EELM [103]	87.00	86.15	87.59	77.36
LD+KNN [101]	86.15	81.03	90.24	N/A
OLPP+RF [53]	90.07	89.83	90.24	82.10

Table 4.1: Performance comparisons of 14 methods on the Yeast dataset

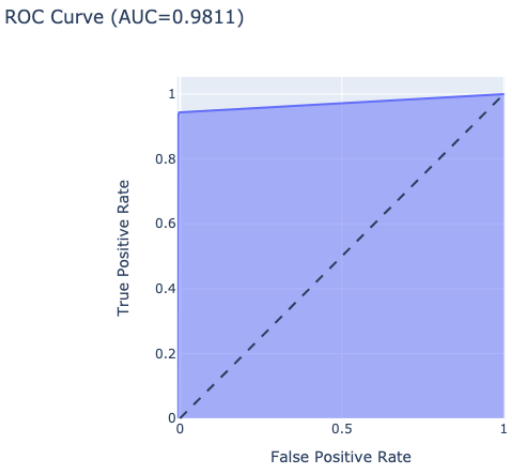


Figure 4.19: ROC for Saccharomyces cerevisiae.

In order to contextualize the performance of Prots2Net in this test, in Table 4.1 we present the results of Prots2Net and other 13 methods introduced in 1.4.6. The evaluation measures of the other methods were obtained from [53]. We observe that Prots2Net has the best error measures. Hence, Prots2Net outperform all the other methods.

4.2.4 Test 3. Homo sapiens predictions

To predict the interactions of Homo sapiens, we considered as similar species to train the model, Pan paniscus and Gorilla gorilla. The performance of Prots2Net in this test is extraordinary, as we can observe in Figure 4.20. All the positive metrics are almost 1, except for the extreme values of the threshold. As remarkable, we can see in Figure 4.21,

that with a threshold value of 0.5, Prots2Net miss 214 interaction but all interaction are correctly predicted, i.e., there are no false positives. All these results are resumed in Figure 4.22 where the ROC curve is displayed with AUC value 0.9984.

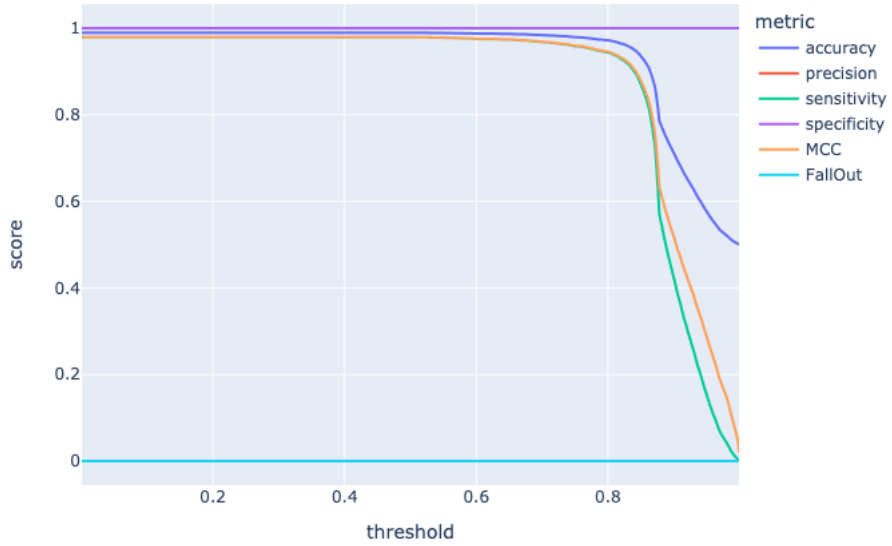


Figure 4.20: Metrics for Homo sapiens.

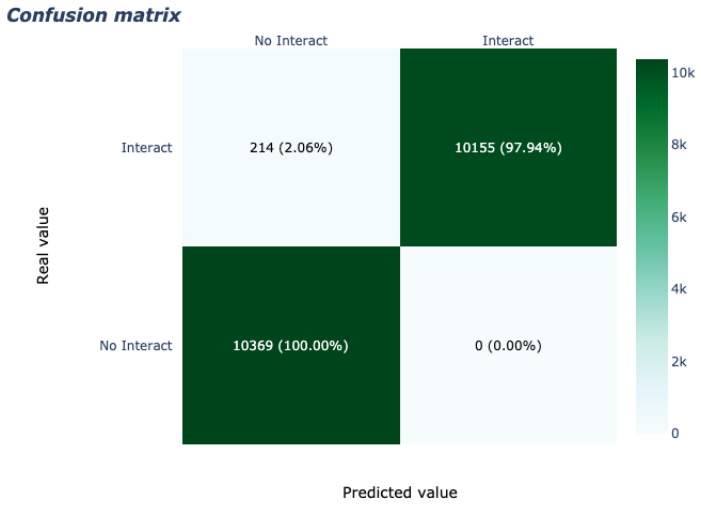


Figure 4.21: Confusion matrix for Homo sapiens with a threshold of 0.5.

Method	Acc	Sen	Pre	MCC
Prots2Net	98.97	97.94	100	97.96
DeepPPI [18]	98.14	96.95	99.13	96.29
MMI + NMBAC [17]	97.57	96.57	98.30	95.13
LDA + RF [74]	96.4	94.2	N/A	92.8
DCT + SMR + WSRC [39]	96.30	92.63	99.59	92.82
MMI [17]	96.08	95.05	96.97	92.17
LDA + RoF [74]	95.7	97.6	N/A	91.8
NMBAC [17]	95.59	94.06	96.94	91.21
AC + RF [74]	95.5	94.0	N/A	91.4
AC + RoF [74]	95.1	93.3	N/A	91.0
LDA + SVM [74]	90.7	89.7	N/A	81.3
AC + SVM [74]	89.3	94.0	N/A	79.2
PseAAC + RF [74]	95.6	94.1	N/A	91.2
PseAAC + RoF [74]	95.3	93.6	N/A	90.7
PseAAC + SVM [74]	91.2	89.9	N/A	92.0
OLPP+RF [53]	96.09	95.20	96.56	92.47

Table 4.2: Performance comparisons of 16 methods on the Human dataset.

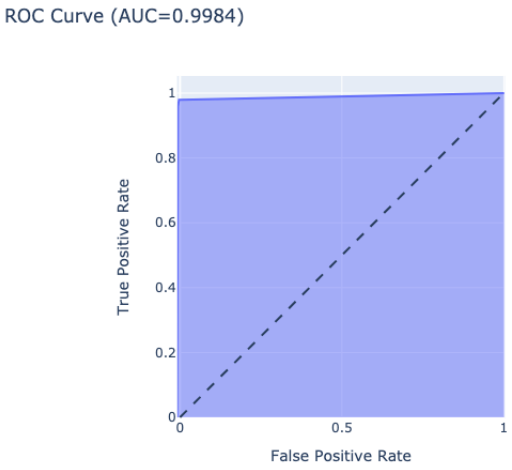


Figure 4.22: ROC for Homo sapiens.

As in the Yeast database, in order to compare the results obtained by Prots2Net with those obtained by other currently known methods, in Table 4.2 we present their evaluation measures. The evaluation measures of the other methods were obtained from [53]. We again can observe that Prots2Net obtained the best error measures. Hence, we conclude that Prots2Net outperform all the other methods.

4.2.5 Impact of species selection

In order to test whether the selection of the two similar species affects the prediction results, we again considered predicting the PPIs in Homo sapiens but, with different pairs of similar species to train the model. Thus, we selected the following pairs of species:

1. Pan paniscus and Gorilla gorilla.
2. Arabidopsis thaliana and Arabidopsis lyrata.
3. Millerozyma farinosa and Saccharomyces cereviciae
4. Salinibacter ruber and Rhodothermus marinus.
5. Archaeglobus fulgidus and Methanopyrus kandlery AV19.

Notice that, we considered pairs of species at different taxonomy distance with Homo sapiens. As a first pair we consider two primates, as a second one two plants, followed by two fungi and then two bacteria and two archaea. As one would expect, the best results were obtained with the two primates, followed by the two plants and the two fungi. The worst results were obtained with the two bacteria and archaea. Figure 4.23 shows the error measures obtained in every test when considering different pairs of similar species.

We can observe there that, indeed, the best results were in the selection of two primates. We obtained a Specificity and Precision values of 1 in every pair of similar species. Recall that the Specificity is defined by $\frac{TN}{N}$ and the Precision is defined by $\frac{TP}{TP+FP}$. Also, we trained the model considering that two proteins interact when their combined interaction score in the STRING database is greater than 999. Hence, this high threshold ensures that the probability to truly interact when the model predicts an interaction is very high. Or, equivalently, there are very few false positives. On the other side, the other error measures clearly show that the model performed better with those pairs of PPINs that are taxonomically close to Homo sapiens. Hence, we conclude that the results improve when the selected species are taxonomically close to the species whose interactions are requested.

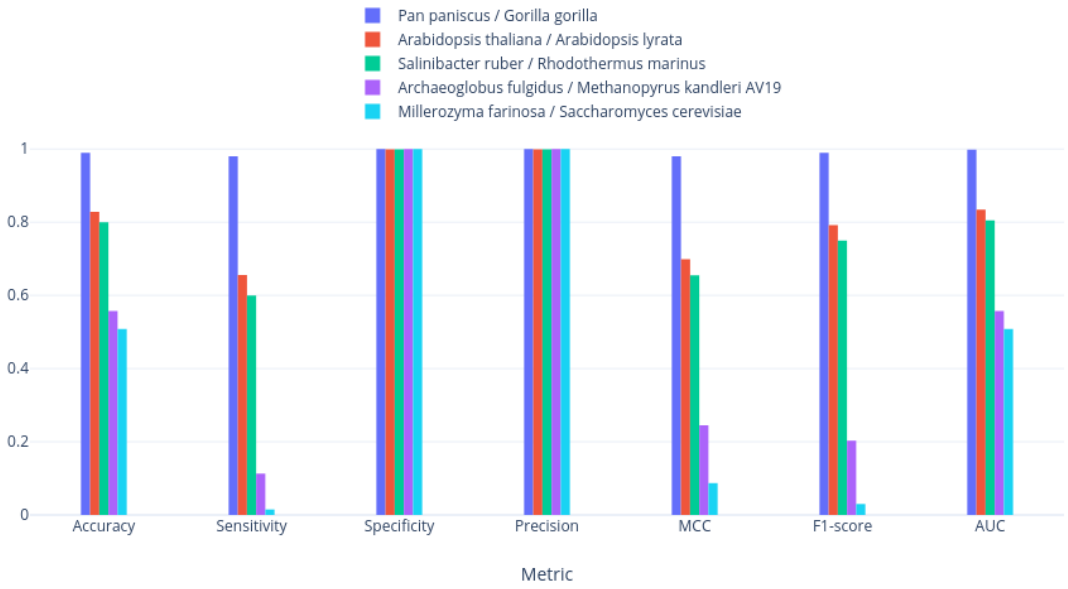


Figure 4.23: Metrics comparison to study species selection.

CHAPTER 5

Conclusion and further work

In this thesis, we have designed and developed new tools for protein-protein interaction networks (PPINs) analysis and prediction. Namely, we have developed a new aligner of PPINs, AligNet, a user-friendly web-based tool to obtain and compare the results produced by different aligners, PINAWeb, and a PPIs predictor, Prots2Net, based only on the protein sequences and two PPINs.

More precisely, AligNet is a new method and software tool for the pairwise global alignment of PPINs aimed to produce biologically meaningful alignments by achieving a good balance between structural matching and protein function conservation. AligNet is a parameter-free algorithm that, given two PPINs, produces a consistent alignment from the smaller network, in terms of number of nodes, to the larger network. Its implementation in R is freely available from `http://bioinfo.uib.es/~recerca/AligNet/`.

In order to assess the correctness of AligNet, we have evaluated the quality of the alignments obtained with it and with the 4 best aligners established in [16, 63], namely: PINALOG, SPINAL, HubAlign, and L-GRAAL. As a result of the comparison between the aligners, we came across again, as it was the case in [16, 63], that the agreement of the alignments obtained with different aligners is very low. Most global aligners achieved a high node coverage, meaning that the average number of assigned nodes in the source network is high, but all of them obtained a very low biological coherence value. With respect to the topological coherence value, some aligners were able to obtain a high score, but it was associated with a low biological coherence score. Overall, we can conclude that AligNet is the aligner that obtained a better balance between topological coherence (it preserves 60% of the edges) and functional coherence (relative function coherence values between 20% and 40% and the highest complex functional coherence score, 25.34), followed by PINALOG, which obtained similar functional coherence scores than AligNet and a bit lower topological coherence scores, but the lowest proficiency value. HubAlign and L-GRAAL obtained high topological coherence scores, but very low CFC values. On the other hand, SPINAL surprisingly obtained a very low topological coherence value.

Thus, if the purpose of the alignment is to correctly align in the biological function sense, our aligner of choice would be AligNet since it is the most precise. However, if the purpose of the alignment is to find some topological similarity, then we would propose to use HubAlign.

Besides AligNet, we have developed PINAWeb, a user-friendly web-based tool aimed at facilitating the obtention of alignments produced by the aligners AligNet, HubAlign, L-GRAAL, PINALOG and SPINAL by avoiding the tiresome work of data processing, and the separate installation of the relevant programs and computations required for each aligner. Alignments are performed either on networks retrieved from the STRING database or from the user's own data. In addition, PINAWeb provides a heatmap to visually compare the performance among aligners, and a table with the results of the matches as well as a consensus score. Moreover, the output includes a json file with information about the networks of interest (nodes and edges) as well as the GO terms of the proteins of interest, the unaligned nodes and edges, the non-preserved nodes and edges, and the annotated proteins.

Two tests were carried out to evaluate the usability of the tool. For the first test, five networks were selected and "all against all" pairwise network alignments were performed using each aligner, giving a total of 50 alignments. The tool was able to produce all the requested alignments in an average time of 5 minutes. The comparison of the results revealed that all the aligners assessed showed a high degree of agreement when used to align similar networks. To reinforce this conclusion, a second alignment test for a pair of subnetworks from H. Sapiens was performed, resulting in a very high degree of agreement between the aligners. These results show that PINAWeb could greatly facilitate the processes involved with the use of aligner software, and we encourage researchers to use this tool to ensure optimum performance when analyzing their experimental data.

Finally, Prots2Net is a multilayer perceptron neural network designed to predict the PPIs of an input set of proteins. The model takes into account sequence information only plus the PPI information available in the STRING database. The training data consists of two selected PPINs from the STRING database, which avoids the lack of *a priori* information needed to train the model. The tests performed and reported in Chapter 4 to evaluate this tool, show that Prots2Net obtains very good results in the six error measures considered here. Furthermore, in the test presented here to evaluate the impact of the species selection, we conclude that the results improve when the selected species are taxonomically close to the species whose interactions are requested.

Regarding the comparison of Prots2Net with other existing methods, Prots2Net outperform the other existing methods, in the Yeast and Human datasets, since it obtains the highest values of accuracy, sensitivity, precision, and Mathews correlation coefficient (MCC). This is probably due to the PPI data considered in the training model. Therefore, Prots2Net is a user-friendly and efficient tool to correctly predict PPIs from a metaproteome or a proteome sample.

To finish the conclusions, we want to point out the interest in the particular case of PPIs, where the interactions are between a viral protein and a protein from its host. This particular case of PPINs are called virus-host PPINs and have been proved to be successful, for instance, in the present outbreak of a coronavirus-associated acute respiratory disease, the COVID-19 pandemic, which forced the scientific community to rapidly analyze the virus-host relationships of the new coronavirus (SARS-CoV-2) human infection [78, 30]. The *modus operandi* of every viral infection is through the interaction between

viral proteins and host proteins, in order to use the host cells to replicate. In this line of research, virus-host protein-protein interaction networks, have become appropriate to analyze virus-host relationships, and information on well-known and studied virus-host protein-protein interaction networks can be carried over to new ones by way of protein-protein interaction network comparison and alignment. Therefore, as a further work of this thesis, we plan to revise our PPINs aligner AligNet to be of use for virus-host protein-protein interaction networks. And, we also consider to adapt Prots2Net to the particular case of virus-host protein-protein interactions.

Bibliography

[1] Ahmet E. Aladağ and Cesim Erten. SPINAL: Scalable protein interaction network alignment. *Bioinformatics*, 29(7):917–924, 2013.

[2] A. Alcalá, G. Riera, I. García, R. Alberich, and M. Llabrés. Pinaweb: A web-based tool for the comparison of protein-protein interaction networks aligners. *bioRxiv*, 2021.

[3] Adrià Alcalá, Ricardo Alberich, Mercè Llabrés, Francesc Rosselló, and Gabriel Valiente. Alignet: alignment of protein-protein interaction networks. *BMC bioinformatics*, 21(6):1–22, 2020.

[4] Adria Alcala Mena and Merce Llabres Segura. Prots2net: a ppin predictor of a proteome or a metaproteome sample. *bioRxiv*, 2022.

[5] A Ali, R Viswanath, SS Patil, and KR Venugopal. A Review of Aligners for Protein Protein Interaction Networks. In *Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017 2nd IEEE International Conference on*, pages 1651–1655. IEEE, 2017.

[6] Ji-Yong An, Zhou Yong, Yu-Jun Zhao, and Zi-Ji Yan. An efficient feature extraction technique based on local coding pssm and multifeatures fusion for predicting protein-protein interactions. *Evolutionary Bioinformatics*, 15:117693431987992, 10 2019.

[7] Johnathon D Anderson, Henrik J Johansson, Calvin S Graham, Mattias Vesterlund, Missy T Pham, Charles S Bramlett, Elizabeth N Montgomery, Matt S Mellema, Renee L Bardini, Zelenia Contreras, et al. Comprehensive proteomic analysis of mesenchymal stem cell exosomes reveals modulation of angiogenesis via nuclear factor-kappab signaling. *Stem cells*, 34(3):601–613, 2016.

[8] N Leigh Anderson and Norman G Anderson. Proteome and proteomics: new technologies, new concepts, and new words. *Electrophoresis*, 19(11):1853–1861, 1998.

[9] M. Ashburner, C.A. Ball, Judith Blake, David Botstein, Heather Butler, and J. Cherry. Gene ontology: Tool for the unification of biology. *The Gene Ontology Consortium. Nat Genet*, 25:25–29, 01 2000.

[10] Alex Bateman, Maria-Jesus Martin, Sandra Orchard, Michele Magrane, Rahat Agivetova, Shadab Ahmad, Emanuele Alpi, Emily Bowler-Barnett, Ramona

Britto, Borisas Bursteinas, Hema Bye-A-Jee, Ray Coetzee, Austra Cukura, Alan Silva, Paul Denny, Tunca Dogan, Thankgod Ebenezer, Jun Fan, Leyla Castro, and Douglas Teodoro. Uniprot: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49, 11 2020.

[11] Walter P Blackstock and Malcolm P Weir. Proteomics: quantitative and physical mapping of cellular proteins. *Trends in biotechnology*, 17(3):121–127, 1999.

[12] Ralf Borndörfer and Olga Heismann. The hypergraph assignment problem. *Discrete Optimization*, 15:15–25, 2015.

[13] Stephen Bustin. Molecular biology of the cell, ; isbn: 9780815344643; and molecular biology of the cell, the problems book; isbn 9780815344537, 2015.

[14] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L. Madden. Blast+: architecture and applications. *BMC Bioinformatics*, 10(1):421, Dec 2009.

[15] Jung-Hsien Chiang and Hsu-Chun Yu. Literature extraction of protein functions using sentence pattern mining. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1088–1098, 2005.

[16] Connor Clark and Jugal Kalita. A comparison of algorithms for the pairwise alignment of biological networks. *Bioinformatics*, 30(16):2351–2359, 2014.

[17] Yijie Ding, Jijun Tang, and Fei Guo. Predicting protein-protein interactions via multivariate mutual information of protein sequences. *BMC bioinformatics*, 17:398, 09 2016.

[18] Xiuquan Du, Shiwei Sun, Changlin Hu, Yu Yao, Yuanting Yan, and Yanping Zhang. Deepppi: Boosting prediction of protein–protein interactions with deep neural networks. *Journal of Chemical Information and Modeling*, 57(6):1499–1510, 2017. PMID: 28514151.

[19] Ahed Elmsallati, Connor Clark, and Jugal Kalita. Global alignment of protein-protein interaction networks: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 13(4):689–705, 2016.

[20] Anton J Enright, Ioannis Iliopoulos, Nikos C Kyrpides, and Christos A Ouzounis. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402(6757):86–90, 1999.

[21] Cesim Erten. *Global Alignment of PPI Networks*, pages 3–25. Springer International Publishing, Cham, 2021.

[22] Ask Solem et contributors. Celery - distributed task queue.

[23] Center for Computational Molecular Biology. About the ccmb.

[24] Python Software Foundation. What is python? executive summary.

[25] R Foundation. R: What is r?

[26] Martin Fowler and James Lewis. Microservices, 3 2014.

[27] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dümpelfeld, Angela Edelmann, Marie-Anne Heurtier, Verena Hoffman, Christian Hoefert, Karin Klein, Manuela Hudak, Anne-Marie Michon, Malgorzata Schelder, Markus Schirle, Marita Remor, Tatjana Rudi, Sean Hooper, Andreas Bauer, Tewis Bouwmeester, Georg Casari, Gerard Drewes, Gitte Neubauer, Jens M. Rick, Bernhard Kuster, Peer Bork, Robert B. Russell, and Giulio Superti-Furga. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, Mar 2006.

[28] Yunus Emre Göktepe and Halife Kodaz. Prediction of protein-protein interactions using an effective sequence based combined method. *Neurocomputing*, 303:68–74, 2018.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[30] David E Gordon, Joseph Hiatt, Mehdi Bouhaddou, Veronica V Rezelj, Svenja Ulferts, Hannes Braberg, Alexander S Jureka, Kirsten Obernier, Jeffrey Z Guo, Jyoti Batra, et al. Comparative host-coronavirus protein interaction networks reveal pan-viral disease mechanisms. *Science*, 370(6521):eabe9403, 2020.

[31] David E. Gordon, Gwendolyn M. Jang, Mehdi Bouhaddou, Jiewei Xu, Kirsten Obernier, Kris M. White, Matthew J. O'Meara, Veronica V. Rezelj, Jeffrey Z. Guo, Danielle L. Swaney, Tia A. Tummino, Ruth Hüttenhain, Robyn M. Kaake, Alicia L. Richards, Beril Tutuncuoglu, Helene Foussard, Jyoti Batra, Kelsey Haas, Maya Modak, Minkyu Kim, Paige Haas, Benjamin J. Polacco, Hannes Braberg, Jacqueline M. Fabius, Manon Eckhardt, Margaret Soucheray, Melanie J. Bennett, Merve Cakir, Michael J. McGregor, Qiongyu Li, Bjoern Meyer, Ferdinand Roesch, Thomas Vallet, Alice Mac Kain, Lisa Miorin, Elena Moreno, Zun Zar Chi Naing, Yuan Zhou, Shiming Peng, Ying Shi, Ziyang Zhang, Wenqi Shen, Ilsa T. Kirby, James E. Melnyk, John S. Chorba, Kevin Lou, Shizhong A. Dai, Inigo Barrio-Hernandez, Danish Memon, Claudia Hernandez-Armenta, Jiankun Lyu, Christopher J. P. Mathy, Tina Perica, Kala Bharath Pilla, Sai J. Ganesan, Daniel J. Saltzberg, Ramachandran Rakesh, Xi Liu, Sara B. Rosenthal, Lorenzo Calviello, Srivats Venkataramanan, Jose Liboy-Lugo, Yizhu Lin, Xi-Ping Huang, YongFeng Liu, Stephanie A. Wankowicz, Markus Bohn, Maliheh Safari, Fatima S. Ugur, Cassandra Koh, Nastaran Sadat Savar, Quang Dinh Tran, Djoshkun Shengjuler, Sabrina J. Fletcher, Michael C. O'Neal, Yiming Cai, Jason C. J. Chang, David J. Broadhurst, Saker Klippsten, Phillip P. Sharp, Nicole A. Wenzell, Duygu Kuzuoglu-Ozturk, Hao-Yuan Wang, Raphael Trenker, Janet M. Young, Devin A. Cavero, Joseph Hiatt, Theodore L. Roth, Ujjwal Rathore, Advait Subramanian, Julia Noack, Mathieu Hubert, Robert M. Stroud, Alan D. Frankel, Oren S. Rosenberg, Kliment A. Verba, David A. Agard, Melanie Ott, Michael Emerman, Natalia Jura, Mark von Zastrow, Eric Verdin, Alan Ashworth, Olivier Schwartz, Christophe d'Enfert, Shaeri Mukherjee, Matt Jacobson, Harmit S. Malik, Danica G. Fujimori, Trey Ideker, Charles S. Craik, Stephen N. Floor, James S.

Fraser, John D. Gross, Andrej Sali, Bryan L. Roth, Davide Ruggero, Jack Taunton, Tanja Kortemme, Pedro Beltrao, Marco Vignuzzi, Adolfo García-Sastre, Kevan M. Shokat, Brian K. Shoichet, and Nevan J. Krogan. A sars-cov-2 protein interaction map reveals targets for drug repurposing. *Nature*, 583(7816):459–468, 2020.

[32] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic acids research*, 36:3025–30, 06 2008.

[33] Katherine Harmon. Genome sequencing for the rest of us, 2010.

[34] S. Hashemifar and J. Xu. HubAlign: an accurate and efficient method for global alignment of protein-protein interaction networks. *Bioinformatics*, 30(17):i438–i444, 2014.

[35] Henry D Herce, Wen Deng, Jonas Helma, Heinrich Leonhardt, and M Cristina Cardoso. Visualization and targeted disruption of protein interactions in living cells. *Nature communications*, 4(1):1–8, 2013.

[36] et al. Ho y. Sistematic identificatioin of proein complexes in saccharomyces cerevisiae by mass specrometry. *Nature*, 415(6868):180–183, 2002.

[37] Leroy Hood and Lee Rowen. The human genome project: big science transforms biology and medicine. *Genome medicine*, 5(9):1–8, 2013.

[38] Yu-An Huang, Zhu-Hong You, Xin Gao, Leon Wong, and Lirong Wang. Using weighted sparse representation model combined with discrete cosine transformation to predict protein-protein interactions from protein sequence. *BioMed research international*, 2015, 2015.

[39] Yu-An Huang, Zhu-Hong You, Xin Gao, Leon Wong, and Lirong Wang. Using weighted sparse representation model combined with discrete cosine transformation to predict protein-protein interactions from protein sequence. *BioMed Research International*, 2015:1–10, 10 2015.

[40] Michael Huerta, Gregory Downing, Florence Haseltine, Belinda Seto, and Yuan Liu. Nih working definition of bioinformatics and computational biology. *US National Institute of Health*, pages 1–1, 2000.

[41] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574, 2001.

[42] Yuxiang Jiang, Tal Ronnen Oron, Wyatt T. Clark, Asma R. Bankapur, Daniel D'Andrea, Rosalba Lepore, Christopher S. Funk, Indika Kahanda, Karin M. Verspoor, Asa Ben-Hur, Da Chen Emily Koo, Duncan Penfold-Brown, Dennis Shasha, Noah Youngs, Richard Bonneau, Alexandra Lin, Sayed M. E. Sahraeian, Pier Luigi Martelli, Giuseppe Profiti, Rita Casadio, Renzhi Cao, Zhaolong Zhong, Jianlin Cheng, Adrian Altenhoff, Nives Skunca, Christophe Dessimoz, Tunca Dogan, Kai Hakala, Suwisa Kaewphan, Farrokh Mehryary, Tapio Salakoski, Filip

Ginter, Hai Fang, Ben Smithers, Matt Oates, Julian Gough, Petri Törönen, Patrik Koskinen, Liisa Holm, Ching-Tai Chen, Wen-Lian Hsu, Kevin Bryson, Domenico Cozzetto, Federico Minneci, David T. Jones, Samuel Chapman, Dukka BKC, Ishita K. Khan, Daisuke Kihara, Dan Ofer, Nadav Rappoport, Amos Stern, Elena Cibrian-Uhalte, Paul Denny, Rebecca E. Foulger, Reija Hieta, Duncan Legge, Ruth C. Lovering, Michele Magrane, Anna N. Melidoni, Prudence Mutowo-Meullenet, Klemens Pichler, Aleksandra Shypitsyna, Biao Li, Pooya Zakeri, Sarah ElShal, Léon-Charles Tranchevent, Sayoni Das, Natalie L. Dawson, David Lee, Jonathan G. Lees, Ian Sillitoe, Prajwal Bhat, Tamás Nepusz, Alfonso E. Romero, Rajkumar Sasidharan, Haixuan Yang, Alberto Paccanaro, Jesse Gillis, Adriana E. Sedeño-Cortés, Paul Pavlidis, Shou Feng, Juan M. Cejuela, Tatyana Goldberg, Tobias Hamp, Lothar Richter, Asaf Salamov, Toni Gabaldon, Marina Marcet-Houben, Fran Supek, Qingtian Gong, Wei Ning, Yuanpeng Zhou, Weidong Tian, Marco Falda, Paolo Fontana, Enrico Lavezzo, Stefano Toppo, Carlo Ferrari, Manuel Giollo, Damiano Piovesan, Silvio C. E. Tosatto, Angela del Pozo, JoséM. Fernández, Paolo Maietta, Alfonso Valencia, Michael L. Tress, Alfredo Benso, Stefano Di Carlo, Gianfranco Politano, Alessandro Savino, Hafeez Ur Rehman, Matteo Re, Marco Mesiti, Giorgio Valentini, Joachim W. Bargsten, Aalt D. J. van Dijk, Branislava Gemovic, Sanja Glisic, Vladmir Perovic, Veljko Veljkovic, Nevena Veljkovic, Danillo C. Almeida-e Silva, Ricardo Z. N. Vencio, Malvika Sharan, Jörg Vogel, Lakesh Kansakar, Shanshan Zhang, Slobodan Vucetic, Zheng Wang, Michael J. E. Sternberg, Mark N. Wass, Rachael P. Huntley, Maria J. Martin, Claire O'Donovan, Peter N. Robinson, Yves Moreau, Anna Tramontano, Patricia C. Babbitt, Steven E. Brenner, Michal Linial, Christine A. Orengo, Burkhard Rost, Casey S. Greene, Sean D. Mooney, Iddo Friedberg, and Predrag Radivojac. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(1):184, 2016.

[43] Lovisa Johansson. Part 1: Rabbitmq for beginners - what is rabbitmq?

[44] Raja Jothi, Praveen F Cherukuri, Asba Tasneem, and Teresa M Przytycka. Co-evolutionary analysis of domains in interacting proteins reveals insights into domain–domain interactions mediating protein–protein interactions. *Journal of molecular biology*, 362(4):861–875, 2006.

[45] Brian P. Kelley, Bingbing Yuan, Fran Lewitter, Roded Sharan, Brent R. Stockwell, and Trey Ideker. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Res.*, 32(Web Server issue):W83–W88, 2014.

[46] Babak Khorsand, Abdorreza Savadi, and Mahmoud Naghibzadeh. Sars-cov-2-human protein-protein interaction network. *Informatics in medicine unlocked*, 20:100413–100413, 2020.

[47] Gunnar W. Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(1):S59, Jan 2009.

[48] Manfred Koegl and Peter Uetz. Improving yeast two-hybrid screening systems. *Briefings in Functional Genomics and Proteomics*, 6(4):302–312, 2007.

[49] Kim Y et al. Koyutürk M. Pairwise alignment of protein interaction networks. *J. Comput. Biol.*, 13(2):182–199, 2006.

[50] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[51] Maxat Kulmanov and Robert Hoehndorf. DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics*, 36(2):422–429, 07 2019.

[52] Yang Li, Li-Ping Li, Lei Wang, Chang-Qing Yu, Zheng Wang, and Zhu-Hong You. An ensemble classifier to predict protein–protein interactions by combining pssm-based evolutionary information with local binary pattern model. *International journal of molecular sciences*, 20(14):3511, 2019.

[53] Yang Li, Zheng Wang, Liping Li, Zhu-Hong You, Wen-Zhun Huang, Xin-Ke Zhan, and Yan-Bin Wang. Robust and accurate prediction of protein–protein interactions by exploiting evolutionary information. *Scientific Reports*, 11, 08 2021.

[54] Zhenping Li, Yong Wang, Shihua Zhang, Xiang-Sun Zhang, and Luonan Chen. Alignment of protein interaction networks by integer quadratic programming. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5527–5530, 2006.

[55] Zhi Liang, Meng Xu, Maikun Teng, and Liwen Niu. Netalign: A web-based tool for comparison of protein interaction networks. *Bioinformatics (Oxford, England)*, 22:2175–7, 10 2006.

[56] Redis Ltd. Introduction to redis.

[57] Hao Luo, Yan Lin, Feng Gao, Chun-Ting Zhang, and Ren Zhang. Deg 10, an update of the database of essential genes that includes both protein-coding genes and noncoding genomic elements. *Nucleic acids research*, 42, 11 2013.

[58] Llabrés M and Valiente G. Sars-cov-2-human protein-protein interaction network alignment of virus-host protein-protein interaction networks by integer linear programming: Sars-cov-2. *PLoS One.*, 15(12), 2020.

[59] Cheng-Yu Ma and Chung-Shou Liao. A review of protein–protein interaction network alignment: From pathway comparison to global alignment. *Computational and Structural Biotechnology Journal*, 18:2647–2656, 2020.

[60] Liao CS Ma CY. A review of protein-protein interaction network alignment: From pathway comparison to global alignment. *Comput Struct Biotechnol J*, 18:2647–2656, 2020.

[61] Mahmoud A Mahdavi and Yen-Han Lin. False positive reduction in protein-protein interaction predictions using gene ontology annotations. *BMC bioinformatics*, 8(1):1–10, 2007.

[62] N. Malod-Dognin and N. Pržulj. L-graal: Lagrangian graphlet-based network aligner. *Bioinformatics*, 31(13):2182–9, 2015.

[63] Noël Malod-Dognin, Kristina Ban, and Nataša Pržulj. Unified alignment of protein-protein interaction networks. *Scientific Reports*, 7(1):953, Apr 2017.

[64] Noël Malod-Dognin, Kristina Ban, and Nataša Pržulj. Unified alignment of protein-protein interaction networks. *Scientific reports*, 7(1):1–11, 2017.

[65] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[66] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[67] Inc. MongoDB. What is mongodb?

[68] Loris Nanni, Sheryl Brahnam, and Alessandra Lumini. Wavelet images and chou's pseudo amino acid composition for protein classification. *Amino Acids*, 43, 08 2011.

[69] Manikandan Narayanan and Richard Karp. Comparing protein interaction networks via a graph match-and-split algorithm. *Journal of computational biology : a journal of computational molecular cell biology*, 14:892–907, 10 2007.

[70] Sam Newman. *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2015.

[71] Behnam Neyshabur, Ahmadreza Khadem, Somaye Hashemifar, and Seyed Shahriar Arab. NETAL: a new graph-based method for global alignment of protein–protein interaction networks. *Bioinformatics*, 29(13):1654–1662, 05 2013.

[72] Somashe Niranjanakumari, Erika Lasda, Robert Brazas, and Mariano A Garcia-Blanco. Reversible cross-linking combined with immunoprecipitation to study rna–protein interactions in vivo. *Methods*, 26(2):182–190, 2002.

[73] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814–818, 2005.

[74] Xiaoyong Pan, Ya-nan Zhang, and Hong-Bin Shen. Large-scale prediction of human protein-protein interactions from amino acid sequence based on latent topic features. *Journal of proteome research*, 9:4992–5001, 10 2010.

[75] D. Park, R. Singh, M. Baym, C. Liao, and B. Berger. Isobase: A database of functionally related proteins across ppi networks. *Nucleic Acids Research*, 39:295–300, 2011.

[76] Robert Patro and Carl Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics (Oxford, England)*, 28, 10 2012.

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[78] Laure Perrin-Cocon, Olivier Diaz, Clémence Jacquemin, Valentine Barthel, Eva Ogire, Christophe Ramière, Patrice André, Vincent Lotteau, and Pierre-Olivier Vidalain. The current landscape of coronavirus-host protein–protein interactions. *Journal of translational medicine*, 18(1):1–15, 2020.

[79] Hang T. T. Phan and Michael J. E. Sternberg. PINALOG: A novel approach to align protein interaction networks—implications for complex detection and function prediction. *Bioinformatics*, 28(9):1239–1245, 2012.

[80] Oscar Puig, Friederike Caspary, Guillaume Rigaut, Berthold Rutz, Emmanuelle Bouveret, Elisabeth Bragado-Nilsson, Matthias Wilm, and Bertrand Séraphin. The tandem affinity purification (tap) method: a general procedure of protein complex purification. *Methods*, 24(3):218–229, 2001.

[81] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[82] Burkhard Rost, Jinfeng Liu, Rajesh Nair, Kazimierz O Wrzeszczynski, and Yanay Ofran. Automatic prediction of protein function. *Cellular and Molecular Life Sciences CMLS*, 60(12):2637–2650, 2003.

[83] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[84] Andreas Ruepp, Brigitte Waegele, Martin Lechner, Barbara Brauner, Irmtraud Dunger, Gisela Fobo, Goar Frishman, Corinna Montrone, and Hans-Werner Mewes. Corum: The comprehensive resource of mammalian protein complexes-2009. *Nucleic acids research*, 38:D497–501, 11 2009.

[85] Andreas Ruepp, Alfred Zollner, Dieter Maier, Kaj Albermann, Jean Hani, Martin Mokrejs, Igor Tetko, Ulrich Güldener, Gertrud Mannhaupt, Martin Münsterkötter, and H. Werner Mewes. The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic acids research*, 32(18):5539–5545, Oct 2004. 15486203[pmid].

[86] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular systems biology*, 3(1):88, 2007.

[87] Abdel-Nasser Sharkawy. Principle of neural network and its main types. *Journal of Advances in Applied & Computational Mathematics*, 7:8–19, 2020.

[88] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[89] YI SHEN. *LOSS FUNCTIONS FOR BINARY CLASSIFICATION AND CLASS PROBABILITY ESTIMATION*. PhD thesis, University of Pennsylvania, 2005.

[90] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences of the United States of America*, 105:12763–8, 10 2008.

[91] Roy D. Sleator and Paul Walsh. An overview of in silico protein function prediction. *Archives of Microbiology*, 192(3):151–155, 2010.

[92] Jingchun Sun, Jinlin Xu, Zhen Liu, Qi Liu, Aimin Zhao, Tieliu Shi, and Yixue Li. Refined phylogenetic profiles method for predicting protein–protein interactions. *Bioinformatics*, 21(16):3409–3415, 2005.

[93] Szklarczyka and Damian et al. The string database in 2017: Quality-controlled protein–protein association networks, made broadly accessible. *Nucleic Acids Research*, 45:362–368, 2018.

[94] Madden T. The blast sequence analysis tool, 8 2003.

[95] Kevin Titeca, Irma Lemmens, Jan Tavernier, and Sven Eyckerman. Discovering cellular protein-protein interactions: Technological strategies and opportunities. *Mass spectrometry reviews*, 38(1):79–111, 2019.

[96] Yanbin Wang, Zhu-Hong You, Shan Yang, Xiao Li, Tong-Hai Jiang, and Xi Zhou. A high efficient biological language model for predicting protein–protein interactions. *Cells*, 8(2), 2019.

[97] Leon Wong, Zhu-Hong You, Zhong Ming, Jianqiang Li, Xing Chen, and Yu-An Huang. Detection of interactions between proteins through rotation forest and local phase quantization descriptors. *International Journal of Molecular Sciences*, 17(1), 2016.

[98] Cathy Wu, Lai-Su Yeh, Hongzhan Huang, Leslie Arminski, Jorge Castro-Alvear, Yongxing Chen, Zhangzhi Hu, Panagiotis Kourtesis, Robert Ledley, Baris Suzek, C. R. Vinayaka, Jian Zhang, and Winona Barker. The protein information resource. *Nucleic acids research*, 31:345–7, 02 2003.

[99] Xiaomei Wu, Erli Pang, Kui Lin, and Zhen-Ming Pei. Improving the measurement of semantic similarity between gene ontology terms and gene products: insights from an edge- and ic-based hybrid method. *PloS one*, 8(5):e66745–e66745, 05 2013.

[100] Ioannis Xenarios, Lukasz Salwínski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30(1):303–305, 01 2002.

[101] Lei Yang, Jun-Feng Xia, and Jie Gui. Prediction of protein-protein interactions from protein sequence using local descriptors. *Protein and peptide letters*, 17:1085–90, 09 2010.

[102] Zhu-Hong You, Lin Zhu, Chun-Hou Zheng, Hong-Jie Yu, Su-Ping Deng, and Zhen Ji. Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set. *BMC bioinformatics*, 15 Suppl 15:S9, 12 2014.

[103] Zhuhong You, Ying-Ke Lei, Lin Zhu, Junfeng Xia, and Bing Wang. Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. *BMC Bioinformatics*, 14:S10 – S10, 2013.

[104] Yu Zhou, Yun Gao, and Ying Zheng. Prediction of protein-protein interactions using local description of amino acid sequence. *Advances in Computer Science and Education Applications, Pt Ii*, 202, 01 2011.

[105] Heng Zhu and Michael Snyder. Protein chip technology. *Current opinion in chemical biology*, 7(1):55–63, 2003.