The Dissertation Committee for Meghana Venkata Palukuri Certifies that this is the approved version of the following Dissertation:

Machine learning methods for community detection in networks using known community information

Committee:

Edward M. Marcotte, Supervisor

Rachel Ward

Karl Schulz

Ron Elber

Claus Wilke

Machine learning methods for community detection in networks using known community information

by

Meghana Venkata Palukuri

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2022

Dedication

To Amma, Naanna and Kalyani.

Acknowledgments

I am extremely grateful to Edward, for being an amazing and compassionate guide during my Ph.D., giving me the freedom to learn and explore several research ideas of my interest, and for supporting me through this journey.

I also want to acknowledge my undergraduate advisors and alma mater, IIT Madras for encouraging me to pursue a Ph.D., and my parents for being supportive of my decision to do so, especially in moving to a different country (after 23 years of living in the same city as them).

I am grateful to my parents, sister, roommates, colleagues, friends, relatives, the Marcotte lab members, the Oden Institute members, UT, and Austin, for providing a wonderful social environment that helped me work on my Ph.D. over the last 5 years of my life.

A special mention to Claire McWhite and Kevin Drew who introduced me to computational work in the lab, Ridhi Patil for being an amazing mentee on my second project on reinforcement learning, and Eric Verbeke and Caitie McCafferty for helpful discussions on my third project on projection clustering. I am grateful to all the professors who have taught me different courses at UT and IITM; my internship mentors at Schlumberger and Amazon, and my Ph.D. committee members who have all contributed to my holistic development as a researcher.

Abstract

Machine learning methods for community detection in networks using known community information

Meghana Venkata Palukuri, Ph.D.

The University of Texas at Austin, 2022

Supervisor: Edward M. Marcotte

In a network, the problem of community detection refers to finding groups of nodes and edges that form 'communities' relevant to the field, such as groups of people with common interests in social networks and fraudulent websites linked to each other on the web. Community detection also yields downstream use-cases such as the summarization of massive networks into smaller networks of communities. We are most interested in mining protein complexes, *i.e.*, communities of interacting proteins, accelerating biological experiments by providing candidates for previously unknown protein complexes. Characterization of protein complexes is important, as they play essential roles in cellular functions and their disruption often leads to disease. Previous methods in community detection comprise a majority of unsupervised graph clustering strategies, which work on the assumption that communities are dense subgraphs in a network - which is not always true. Also, many community detection algorithms are in-memory and serial and do not scale to large networks. In this dissertation, we use knowledge from communities, including rich features from graph nodes, with supervised and reinforcement learning, improving on accuracies, with parallel algorithms ensuring high performance and

scalability. Specifically, we work on (1) learning a community fitness function using supervised machine learning methods with AutoML; (2) a distributed algorithm for finding candidate communities using multiple heuristics; (3) learning to walk trajectories on a network leading to communities with reinforcement learning and (4) feature augmentation with graph node information, such as images and additional graph node embeddings. While we optimize our algorithms on protein complexes that have characteristics such as being overlapping in nature with different topologies, our methods are generalizable to other domains since they learn and use characteristics of communities to predict new communities. Further, in domains with limited known information, the algorithms we develop can be applied by transferring learned knowledge such as dense community fitness functions from other domains. In conclusion, we build Super.Complex, RL complex detection, and DeepSLICEM - three accurate, efficient, scalable, and generalizable community detection algorithms, that effectively utilize known community information with different machine learning methods and also present 3 evaluation measures for accurate community evaluation.

Table of Contents

List of Tables

List of Figures

Chapter 1. Background

Networks are ubiquitous - from the world-wide-web connecting websites and social networks connecting people, to transportation networks connecting different locations, and biological networks such as gene, protein, and metabolic networks. A network or graph, *G* can be represented as a pair of nodes and edges *(V, E)*, where the set of nodes or vertices *V* represents the entities under consideration along with attribute information, and the set of edges *E* represents the interactions between entities. An edge connects a pair of nodes and in a weighted graph, additionally has a weight associated with it indicating the strength of the interaction.

Any group of nodes and edges that can be characterized as a unit can be referred to as a community. For example, groups of proteins interact with each other to collectively perform a specific function in a cell. In a work organization network, members of a certain hierarchy work together to run the organization. Communities can be mined from networks by optimizing the properties they exhibit. One such property exhibited by some communities (that has served as a reference guideline for defining communities in several fields [1]), is when a community has more interactions or connectivity among the community than with the rest of the network. This can be modeled by a community fitness function, mapping subgraphs - groups of nodes and edges from the full graph, to a scalar value representing a score, where a higher score indicates more community resemblance for the subgraph.

However, there exist many communities that do not follow this criterion but can be identified by different properties they exhibit, such as a star-like topology, where one central node interacts with several nodes [2]. Methods for detecting new communities typically comprise an algorithm for finding candidate communities (which many a time uses the community fitness function in this process itself), followed by evaluation and

possible fine-tuning with the community fitness function. Existing community detection methods have primarily tried to optimize for high scores of popular community fitness functions based on dense communities [3] using unsupervised [3]–[9] and semi-supervised machine learning algorithms [10]. When there is sufficient data available on known communities, rather than applying a popular community fitness function to the problem, we hypothesize that it would be more accurate to learn a community fitness function from the known communities. Then, new communities detected with the learned community fitness function can be expected to better resemble known communities in that field.

1.1. COMMUNITY EMBEDDINGS

To learn community fitness functions, first, we need a representation of a community in vector space, termed a community embedding, that is used by a machine-learning algorithm to learn the fitness function. The method to construct community embedding can be used to build graph embeddings for candidate community subgraphs that are then evaluated using the community fitness function. To build community embeddings, features from communities can be extracted and concatenated.

1.1.1. Topological features

As communities exhibit different types of topological structures on the graph, these can be learned by considering useful topological features of communities. Based on graph theory, features such as the community's number of nodes, the number of edges, and more, such as the intra-cluster density can be handcrafted. Several node-wise topological properties exist, such as the degree d_v of a vertex v which is the number of its first or immediate neighboring nodes. For a community, statistical measures of the combinations of individual node features can be used to characterize a set of nodes, such as the mean

value, standard deviation, and maximum and minimum values. In weighted networks, the definitions of several topological features change to incorporate the edge weights and give the weighted variants. For instance, the weighted degree of a vertex d_v is the sum of the weights of the edges connecting the node to its immediate neighbors.

1.1.2. Domain-specific features

Along with topological features, including domain-specific information about the nodes and edges forming a community can be helpful. For instance, in a social network, node attributes such as a person's interests and edge attributes such as the age when two people became friends can help find communities such as hobby groups.

1.1.2.1. Proteins, protein interactions, and protein complexes:

The application we are most interested in is the identification of candidate protein complexes, computationally possible with community detection by, (i), grouping interacting proteins in protein-protein interaction networks, and, (ii), determining their 3D structures by grouping their 2D projection images photographed from a cell sample. A protein complex is a group of proteins that interact with each other to perform a particular function in a cell, the basic biological unit of all living organisms. We further motivate our application with a brief discussion about proteins.

For perspective on the abundance of proteins, in one of the most common model organisms used for biological experiments, *E. coli* (**Figure 1.1**), a bacterium found in the intestines of many animals including humans, proteins constitute 50% of the dry weight of the cell, as compared to other important components of the cell-like DNA and RNA which contribute only 3% and 20% respectively [11]. While DNA codes instructions for the functions to be carried out in the cell, RNA decodes these instructions and passes them on

to proteins, which are the primary macromolecules in the cell that carry out these instructions. However, proteins rarely act alone and interact with other proteins by physically binding to them in order to perform these functions. Typically, a particular cellular function involves many proteins that bind to each other, either simultaneously or over a period of time, and can be characterized as a protein complex. An example of a human protein complex is given in **Figure 1.1**.

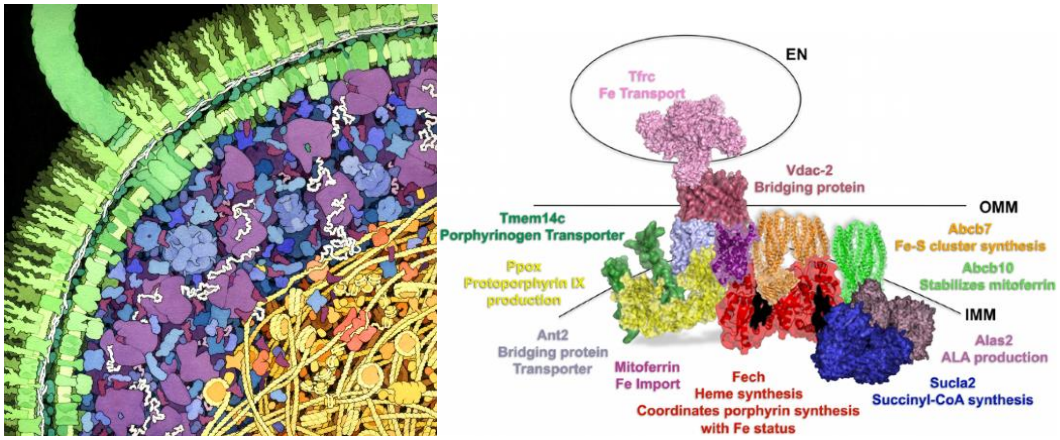


Figure 1.1. Proteins and protein complexes

Left. An artistic rendition of the cross-section of an *E. coli* cell by David Goodsell [12]. The yellow and orange parts comprise DNA and proteins that interact with the DNA. The L-shaped maroon molecules and white strands are RNA. Most other molecules are different proteins, for example, the green molecules lining the two outermost concentric membranes are transmembrane proteins and the blue molecules are protein enzymes. **Right.** a blood protein complex [13].

Extensive biological experiments have investigated the physical interactions between proteins, and these have been modeled via protein-protein interaction networks, where the strength of interactions has been incorporated as network edge weights. While several biological experiments (such as AP/MS, affinity purification with mass spectrometry, and CF/MS, co-fractionation with mass spectrometry) have been performed to identify stable complexes, they show moderate false-positive and high false-negative rates, for example, by also extracting proteins attached to the actual protein complexes and not being able to extract protein complexes deeply embedded in the cell membranes respectively. Computational analysis of protein-protein interaction networks can therefore be very useful in identifying accurate protein complexes and will help augment and direct experimental methods. Protein characteristics such as the sizes, weights, polarities, and frequencies of the constituent amino acids have been used as domain-specific features in some existing methods.

1.2. FITNESS FUNCTIONS WITH SUPERVISED METHODS

Once we have community embeddings, we can use a supervised machine learning model to learn the community fitness function. This can be posed as a binary machine learning classification task between communities (positives) and non-communities (negatives) if we have sufficient embeddings of subgraphs that are not communities. This can either come from labeled data in the field or when there is a shortage of such data, as is the case in our application, the negatives can be modeled as random walks on the network.

1.3. CANDIDATE COMMUNITY SEARCH METHODS

The community fitness function that we learned is used in the process of selecting candidate communities from the network and evaluating them. A common strategy is to select a seed node and grow it into a candidate community - a process that is repeated with different seeds to generate a set of candidate communities. Nodes of the network can be used as seeds. Some algorithms such as the greedy clique expansion algorithm [3] use maximal cliques as seeds. Cliques are fully connected subgraphs, *i.e.*, every node is connected to every other node in the subgraph, and maximal cliques are cliques that are not subgraphs of any other cliques. At each step of the growth of a seed into the final community, the new neighboring node to be added to the current subgraph is selected using one of several heuristics. For example, in a *greedy heuristic*, the new node to be added is the maximum scoring neighbor, *i.e.*, the neighbor which on addition gives the maximum community score value when compared to other nodes. Another *greedy growth heuristic* is to add a node only if it improves the community score. Another famous heuristic is *iterative simulated annealing*. In the standard formulation, the objective is to find a state that minimizes energy. In this search, states are perturbed, *i.e.*, moves are made, leading to exploration of the energy landscape. All moves are accepted except for moves that lead to a state of higher energy. These are accepted with a probability based on the difference between the energies of the previous and current state. Our objective function of maximizing the community score function (minimizing energy) can be stated as,

$$\min_C -S(C)\,. \tag{1.1}$$

Here, *C* is a candidate subgraph and *S(C)* is the community fitness function. At each stage of the growth of the current subgraph, its maximum scoring neighbor is added, except in the case when the new community score of the subgraph is lesser than the value before adding the new node (*i.e.*, $Snew < Sold$).

1.4. EXISTING SUPERVISED METHODS

Having described all the elements necessary to understand existing supervised methods, we now turn to a survey of these and provide a short discussion. Most supervised methods follow the paradigm described and have used different machine learning algorithms to learn community fitness functions and different heuristics to pick candidate communities.

The first supervised method [2] used a support vector machine (SCI-SVM) and a Bayesian network (SCI-BN) with 33 features with a greedy heuristic, followed by iterative simulated annealing. Stopping criteria for the growth of a seed node include limiting the rounds of growth, checking for score improvement over multiple iterations, and checking for overlap with learned candidate communities so far. A second approach [14] recursively trained a two-layer feed-forward neural network model, NN for the classifier using 43 features. This greedy heuristic sequentially grows seeds of the highest degree with similar stopping criteria as [2]. Supervised learning protein complex detection SLPC [15] uses a regression model (RM) with 10 topological features, solved by gradient descent. An algorithm using modified cliques finds and grows maximal cliques using a random but exhaustive neighbor selection followed by a greedy growth heuristic. The algorithm stops when no node addition can yield a higher score, after which they merge some pairs of overlapping complexes with an overlap greater than a threshold. ClusterEPs, short for cluster emerging patterns [7] uses a score function based on noise-tolerant emerging patterns (NEPs) which are minimal discriminatory feature sets using 22 features, along with an average node degree term. Like [14], the heuristic for this method also grows the highest degree seed nodes sequentially. The neighboring node that shares the maximum number of edges with the current subgraph is selected as a candidate for growth in each iteration and a greedy growth heuristic is used, stopping when the score is greater than 0.5.

ClusterSS, short for clustering with supervised and structural information [16] uses a neural network with one hidden layer and 17 features, along with a traditional structural score function from [9]. A greedy heuristic grows seed nodes, also considering deletion of any existing subgraph nodes, with an optimization step of considering only the top k nodes by degree. The stopping criterion is when the new score is less than a factor times the old score. Both ClusterEPs and ClusterSS merge pairs of communities with overlap greater than a threshold at the end.

ClusterSS can theoretically yield disconnected communities as they perform a step where they remove nodes that yield higher complex scores. This is an issue, as it can be assumed that each community should be completely internally connected. For evaluation, many of these methods reduce the protein interaction network into a subset of the proteins found in known complexes and then perform the candidate community selection on this reduced network. This is not reflective of the real scenario, where we would want to perform candidate community selection on the entire network to determine new complexes along with known complexes. The candidate selection algorithm that performs well on a reduced network, need not perform well on the full network, therefore it is important to test the algorithm on the full network as well. Many of the unsupervised methods such as ClusterONE [9] are good in this regard, as they pick out complexes from the entire network and then compare them with known complexes.

An alternate paradigm to supervised community detection is to view it as a protein classification task, where a protein is classified into one or more communities. Decision trees were implemented by [17] using this paradigm. Here, features are the non-leaf nodes in a decision tree. The leaf nodes indicate complexes, the edge weights indicate feature values, and the tree structure is obtained by training on a known set of complexes. For complex identification, proteins of the PPIN are assigned to complexes by tree-traversal.

They use 65 features. So, based on its features, each protein gets assigned to a particular partition arrived at by making non-overlapping choices in a decision tree. This means that using a single decision tree does not support overlapping protein complexes. To circumvent this problem, an ensemble of trees could be used to get multiple memberships of complexes for proteins. In a decision tree paradigm, however, it is unclear how new protein complexes are predicted. In the paradigm of classifying proteins into communities, it is important to make sure properties such as internal connectivity are satisfied by the algorithm. This is not an issue with some of the seeding and growth strategies that inherently satisfy this property.

1.5. INTRODUCTION TO DISSERTATION

In this dissertation, we discuss three community detection algorithms we developed, Super.Complex (Chapter 3), RL complex detection (Chapter 4), and DeepSLICEM (Chapter 5). These methods are evaluated using 3 novel evaluation measures, along with existing evaluation measures discussed in Chapter 2. In the appendix, we describe a method based on network analysis, applied to determine the role of gut microbiota and diet in autism.

The goals of our work were to build methods for community detection in networks such that they are (i) accurate, by making use of known information where available and transferring knowledge where limited information is present; (ii) efficient and scalable, in terms of time complexities and coding design; and (iii) generalizable to multiple domains while allowing specific information relevant to the domain to be incorporated - as we demonstrate in the case of protein complex detection from protein interaction networks (the application the community detection methods in Chapters 3 and 4 are tested on), and utilization of structural information from 2D projection images in grouping them to their

respective protein complex (Chapter 5). We worked on these goals by splitting our work into fairly independent parts, formulating specific aims to guide our work. Our first aim was to sufficiently explore different supervised learning methods to learn a good community fitness function, for which we adopt the approach of using AutoML methods (achieved with Super.Complex in Chapter 3). The second aim was to efficiently and scalably sample candidate communities from the network with distributed algorithms (implemented both in Super.Complex from Chapter 3 and RL Complex detection in Chapter 4). Our third aim was to learn trajectories yielding good candidate communities, achieved by framing community detection as a reinforcement learning problem and solving it with value iteration (Chapter 4). The fourth aim was to aid community detection by including rich features from graph nodes (demonstrated with image and graph node embeddings extracted and used for community detection with an AutoML pipeline, DeepSLICEM in Chapter 5) and the fifth aim was to propose good evaluation measures and test them (3 evaluation measures we devised are described in Chapter 2).

We now briefly discuss the significance of our work. Existing supervised methods have explored a limited few machine learning algorithms for community fitness function, limiting achievable accuracies, and employing serial candidate community sampling, limiting their scalability. We develop Super.Complex, an end-to-end scalable and efficient community detection pipeline that explores multiple supervised learning methods with AutoML to learn the most accurate community fitness function from known communities and use it in parallel to sample candidate subgraphs by seeding nodes or maximal cliques and growing them using an epsilon-greedy heuristic with options of iterative simulated annealing, metropolis and additional heuristics built on top. To our knowledge, compared with existing supervised methods, this is the first highly scalable distributed implementation.

We propose the (to our knowledge) first reinforcement learning method for community detection using known community information, which learns candidate community growth heuristics. Novel evaluation measures such as the sensitive FMM F1 score are proposed to overcome shortcomings of existing measures such as failing to penalize the prediction of multiple overlapping communities which should instead have been detected as a single larger community. Super.Complex and RL complex detection are applied to the latest human protein interaction networks to yield high-scoring previously unknown protein complexes, potentially contributing to new biology. We make our results easily accessible via websites we built for interactive visualizations of the learned protein complexes.

Towards finding individual protein complex structures, we work on the problem of grouping 2D projection images of different protein complexes from a cellular extract. Here, we improve community detection on a similarity graph of 2D projections from different protein complexes by combining rich image features from the projections with graph node information, an approach not previously attempted for this application. This community detection algorithm, DeepSLICEM is designed as an efficient AutoML pipeline, exploring several combinations of image and graph node embedding methods, along with different clustering algorithms to find the best one for the application at hand, making it a generalizable paradigm for community detection applications on graphs with nodes having image attributes.

1.6. WORK DONE DURING PH.D. STUDIES:

1.6.1. Papers

1. **Palukuri, M. V.**, & Marcotte, E. M. (2022), "DeepSLICEM: Clustering CryoEM particles using deep image and similarity graph representations." (Paper in prep.)
2. **Palukuri, M. V.**, Patil, R. S., & Marcotte, E. M. (2022). "Molecular complex detection in protein interaction networks through reinforcement learning". *bioRxiv*. https://doi.org/10.1101/2022.06.20.496772
3. **Palukuri, M. V.**, & Marcotte, E. M. (2021). "Super. Complex: A supervised machine learning pipeline for molecular complex detection in protein-interaction networks". *PLOS One*, 16(12), e0262056. https://doi.org/10.1371/journal.pone.0262056
4. Mohammad, Faiz Khan, **Palukuri, M. V.**, et al (2022). "A Computational Framework for Studying Gut-Brain Axis in Autism Spectrum Disorder." *Frontiers in physiology* 13:760753. https://doi.org/10.3389/fphys.2022.760753
5. Kizhuveetil, U., **Palukuri, M. V.**, Karunagaran, D., Rengaswamy, R., Suraishkumar, GK (2019). "Entrainment of superoxide rhythm by menadione in HCT116 colon cancer cells", *Scientific Reports*, Nature Publishing Group 9.1: 3347. https://doi.org/10.1038/s41598-019-40017-7
6. **Palukuri, M. V.**, Shivakumar, S., Sahoo, S., Rengaswamy, R. (2018). "Computational framework for exploring the interplay of diet and gut microbiota in autism." *bioRxiv*: 422931. https://doi.org/10.1101/422931

1.6.2. Conference talks and posters

1. **Palukuri, M. V.**, Marcotte, E. M. "Super.Complex v3.0: A Supervised Machine Learning Pipeline for Molecular Complex Detection in Protein-interaction Networks", *US HUPO (Human Proteome Organization Conference)* [Poster] (2021)
2. **Palukuri, M.V.**, Marcotte, E. M. "Super.Complex: Intelligent subgraph search for communities with deep reinforcement learning", *SIAM MDS: Conference on Mathematics of Data Science*, Cincinnati [Invited Talk] (2020)
3. **Palukuri, M.V.**, Marcotte, E.M. "Super.Complex: A Computational Pipeline for Supervised Community Detection in Graphs", *TACCSTER 2019: TACC Symposium for Texas Researchers*, Austin [Invited Talk, Poster] (2019)
4. **Palukuri, M.V.**, Marcotte, E.M. "Supervised community detection in protein-interaction networks", *The 2nd Annual Meeting of the SIAM Texas Louisiana Section*, Dallas [Best Poster Award] (2019)
5. **Palukuri, M.V.**, Marcotte, E.M. "Supervised community detection", *Workshop on Recent Developments on Mathematical/ Statistical approaches in Data Science (MSDAS)*, Dallas [Poster] (2019)
6. **Palukuri, M.V.**, et al. "An integrated COBRA-PBPK model to study interactions between gut & brain in autism", *5th Conference on Constraint-Based Reconstruction & Analysis*, Seattle [Poster] (2018)
7. Kizhuveetil, U., **Palukuri, M.V.**, Rengaswamy, R., Suraishkumar, GK. "Menadione induced reset of circadian superoxide rhythms in human colon cancer cells", *Free Radical Biology and Medicine*, 112, 91-92, Baltimore [Poster] (2017)

1.6.3. Code developed

1. Super.Complex: https://github.com/marcottelab/super.complex
2. RL complex detection: https://github.com/marcottelab/RL_complex_detection
3. DeepSLICEM: https://github.com/marcottelab/2D_projection_clustering
4. Graph Classification:

   https://github.com/meghanapalukuri/graph_classification

1.6.4. Websites developed for interactive visualizations of results

1. Super.Complex 2.0: https://sites.google.com/view/supercomplex/home
2. Super.Complex 3.0:

   https://marcottelab.github.io/super.complex_website_humap
3. RL complex detection: https://marcottelab.github.io/RL_humap_prediction/

Chapter 2. Evaluation measures for community detection using known information[1]

Before we discuss different community detection methods that we developed, we first examine evaluating predicted communities, to determine both the performance of an algorithm and tuning its hyperparameters to achieve the best performance. In this chapter, we first review existing evaluation measures, note their limitations, and address some of these shortcomings with 3 new evaluation measures for community detection using known information. These methods, along with existing measures are used for evaluating existing community detection algorithms, along with the 3 new community detection methods developed in the subsequent chapters of this dissertation.

2.1. EXISTING EVALUATION MEASURES

Most methods compare nodes of predicted complexes with nodes of known complexes. For direct comparison of predicted and known community nodes, usually only predicted communities with nodes present in the known communities are considered for evaluation. In threshold-based metrics, first, a similarity measure is required to compare two communities C_1 and C_2 and they are determined to be a match if the similarity measure value for the two communities is above a threshold, *i.e.*,

$$sim(C_1, C_2) > t\,, \tag{2.1}$$

where t is a set threshold.

Some examples of similarity measures include the Jaccard similarity and neighborhood affinity.

$$Jaccard\ similarity\ (C_1, C_2) = \frac{|\ C_1\ \cap\ C_2\ |}{|\ C_1 \cup\ C_2\ |} \tag{2.2}$$

[1]We propose new evaluation measures in this chapter, reproduced from published work, Palukuri MV, Marcotte EM (2021) Super. Complex: A supervised machine learning pipeline for molecular complex detection in protein interaction networks. PLoS ONE 16(12): e0262056. https://doi.org/10.1371/journal.pone.0262056.

$$Neighborhood\ affinity(C_1, C_2) = \frac{|\ C_1\ \cap\ C_2\ |^2}{|\ C_1\ | * |C_2|} \quad (2.3)$$

Here, $|C|$ is the number of nodes in community *C*, $|\ C_1\ \cap\ C_2\ |$ is the number of nodes common to both communities, and $|\ C_1 \cup\ C_2\ |$ is their union, *i.e.*, the total number of nodes in the two communities with each node counted once.

A common metric [18] used in many existing methods is a stricter variant of neighborhood affinity, which requires a precision-like and recall-like component to both exceed a threshold, given by,

$$\text{Qi overlap measure: } \frac{|\ C_1\ \cap\ C_2\ |}{|\ C_1\ |} > t \text{ and } \frac{|\ C_1\ \cap\ C_2\ |}{|\ C_2\ |} >\ t \quad (2.4)$$

Here, *t* is the user-specified threshold, usually set to 0.5.

Once we have a way to match a predicted complex to a known complex or vice-versa as described above, standard machine learning metrics such as precision, recall, and F1 scores can be computed.

$$Qi\ et\ al\ Precision\ p\ = \frac{No.\ of\ learned\ communities\ matching\ known\ communities}{No.\ of\ learned\ communities} \quad (2.5)$$

$$Qi\ et\ al\ Recall\ r\ = \frac{No.\ of\ known\ communities\ matching\ learned\ communities}{No.\ of\ known\ communities} \quad (2.6)$$

$$Qi\ et\ al\ F1\ score\ = \frac{2pr}{(p + r)} \quad (2.7)$$

The Qi et al precision, recall, and F1 score consider one-to-many associations, *i.e.*, a learned community can be matched to multiple known communities and vice versa. One-to-one matches are made by measures such as the MMR - maximum matching ratio [9] which calculates the weighted recall (the weights here are neighborhood affinity similarity scores) after selecting a set of one-to-one matches such that the sum of their similarity scores is maximal. Implicit one-to-one associations are made by measures such as

sensitivity (Sn) and positive predictive value (PPV) which resemble a recall-like measure and a precision-like measure respectively. The implicit one-to-one associations made here correspond to picking the community that best matches another community in terms of the number of common nodes. The Sn-PPV accuracy (SPA) combines sensitivity and PPV, resembling a composite measure like the F1 score that combines Precision and Recall. Note that this measure makes one-to-many matches. Letting the set of learned communities be *L* and the set of known communities be *K*, we have,

$$Sn = \frac{\sum_{C_k \in K} \max_{C_l \in L} | C_l \cap C_k |}{\sum_{C_k \in K} | Ck |}, \tag{2.8}$$

$$PPV = \frac{\sum_{C_l \in L} \max_{C_k \in K} | C_l \cap C_k |}{\sum_{C_l \in L} \sum_{C_k \in K} | C_l \cap C_k |}, \tag{2.9}$$

$$SPA = \sqrt{(Sn^{*} PPV)}. \tag{2.10}$$

2.2. THREE NOVEL EVALUATION MEASURES TO COMPARE LEARNED COMMUNITIES WITH KNOWN COMMUNITIES

Comparing sets of learned and known communities accurately is an outstanding issue. Poor evaluation measures do not satisfactorily identify the quality of learned communities and make it difficult to evaluate a community detection algorithm. Sets of learned communities achieving high scores with existing evaluation measures have been observed to have a lot of redundancies, e.g. multiple learned communities are very similar with high overlaps [19]. Known big communities were also observed to be split into several learned communities while still achieving good scores on evaluation measures. While it is undesirable to have many false negatives, having many false positives is more hurtful, as

wet-lab experiments for biological validation tend to be quite expensive and time-consuming to perform. Therefore, we concentrate on including precision-like measures that compute false positives. Further, evaluation measures that are not sensitive to changes in the sets of learned communities limit our abilities to iterate successfully over algorithm modifications to improve algorithms. We examine the specific shortcomings of different evaluation measures and propose new measures to help overcome the issues discussed and construct robust yet sensitive measures.

2.2.1. F-similarity-based Maximal Matching F-score (FMMF)

An issue with many measures such as Qi et al F1 score (**Equation 2.7**) and SPA **(Equation 2.10)** is that they don't penalize redundancy, *i.e.*, if we learn multiple same or very similar communities which are each individually high scoring, we will get a high value of precision-like measures. This is because in many cases, many-to-one matches are being made between learned communities and known communities. To deal with such issues, it is best to make one-to-one matches. The MMR (Maximal Matching Ratio) is one such good measure, however, it only calculates a recall-like measure by dividing the sum of the weights of edges (in a maximal sum of one-to-one edge weights) by the total number of known communities. Taken alone this cannot account for precision, for instance, if we learn a series of random subgraphs, these have low weights and will be ignored, while a high MMR score can be obtained from only a small number of high-quality learned communities. Therefore, we define the precision equivalent for MMR, P_{FFM} in **Figure 2.1c**.

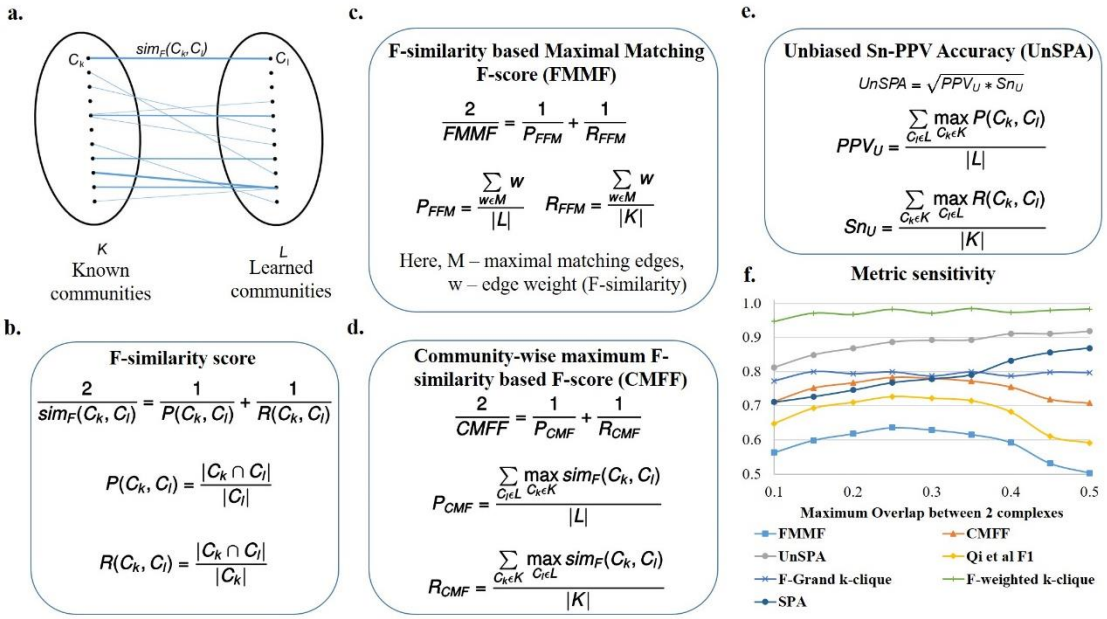


Figure 2.1. Proposed evaluation measures - FMMF, CMFF, and UnSPA are sensitive metrics.

a. Bipartite graph, where each edge weight corresponds to the F-similarity ($sim_F(C_k, C_l)$) between C_k, a known community from K, the set of known communities and C_l, a learned community from L, the set of learned communities. **b**. The F-similarity score combines precision ($P(C_k, C_l)$) and recall ($R(C_k, C_l)$) measures, computed as fractions of the number of common nodes w.r.t the number of nodes in a community. $|C|$is the number of nodes in community C and $|\ C_1 \cap C_2\ |$ is the number of nodes common to both communities. **c.** F-similarity-based Maximal Matching F-score (FMMF) combines precision (P_{FFM}) and recall (R_{FFM}) measures computed for a maximal matching, M of the bipartite graph in **Figure 2.1a d.** Community-wise Maximum F-similarity based F-score (CMFF) combines precision (P_{CMF}) and recall (R_{CMF}) measures, averaging over the maximum F-similarity score for a community in a particular set (e.g. known communities) w.r.t to a community of the other set (e.g. learned communities) **e.** UnSPA is an unbiased version of Sn-PPV accuracy (SPA), computed as the geometric mean of unbiased PPV (PPV_u) and unbiased Sensitivity (Sn_u), computed similar to precision and recall measures in CMFF, only, instead of the F-similarity score, precision and recall similarity scores are used respectively **f.** Sensitivity of different evaluation measures w.r.t. (maximum pairwise Jaccard coefficient) overlap between communities shows that FMMF, CMFF, UnSPA, and existing measures Qi et al F1 score, and SPA are sensitive metrics, with FMMF, CMFF, and Qi et al F1 score following the desired trend. Here, each data point on the plot corresponds to a measure evaluating an individual run of Super.Complex's merging algorithm applied to

hu.MAP 1.0 with a maximum Jaccard overlap threshold set to the x-axis value (**Chapter 3**).

In **Figure 2.1c**, *M* is a set of weights of a set of maximal one-to-one matches, found using Karp's algorithm [20]. The weight w that we use is the F-similarity score (**Figure 2.1b**), also described in the next section, Community-wise Maximum F-similarity based F-score (CMFF), unlike the neighborhood affinity used in the original MMR. Correspondingly we can define an F-score, FMMF, as the harmonic mean of the precision P_{FFM} and recall R_{FFM}, also shown in **Figure 2.1c**.

By doing a one-to-one match, we are also indirectly penalizing cases where the benchmark community is split into multiple smaller communities in the learned set of communities, since the measure considers the weight of only one of the smaller learned communities that comprise the known community, ignoring the rest. Thus only the small weight of the matched community is considered, penalizing this case, unlike one-to-many measures that aggregate the contributions from each of the smaller communities to finally achieve a high score.

2.2.2. Community-wise Maximum F-similarity-based F-score (CMFF)

F1 scores at the individual community match level between known and learned communities have been computed in previous work [3], along with histograms of these scores for all known communities. While their work [3] does not state the exact formulation of their F1 score, we are inspired by them to define an F1 score at the match level, *i.e.*, an F-similarity score, by comparing the nodes of a learned and a known community. Our F-similarity score is a combination of the recall (of the nodes of the known community) and the precision (of the nodes of the learned community), as shown in **Figure 2.1b**.

Our F-similarity score can be compared with a threshold to determine a match, and then the overall precision, recall, and F1 scores for the set of predictions can be computed. Alternatively, our F-similarity score can be used to determine the best matches for communities and overall measures can be defined that can be investigated to reveal the contributions at the individual match level as well. For interpretability at the match level, similar to the unbiased sensitivity and PPV metrics (as discussed in the next section, Unbiased Sn-PPV Accuracy (UnSPA)), we can define precision and recall measures that evaluate, for each community, the closest matching community in the other set using a similarity metric. Using the F-similarity score as the similarity metric here, we define precision and recall-like measures, and combine them into the F1-like measure, CMFF - Community-wise Maximum F-similarity based F-score, as shown in **Figure 2.1d**. We detail a general framework to construct similar measures in the next paragraph, drawing inspiration from modifications to the Qi et al F1 score. This framework also gives another method of constructing the CMFF.

In the Qi et al measures from [18], a binary indication of a possible match is used, *i.e.*, as long as there exists a possible match, it is used as a 1 or 0 count towards the aggregate precision or recall measures. Having a measure that provides matches between learned and known communities allows easy identification of previously unknown communities. One-to-many matches such as Qi et al precision-recall (PR) measures that do not use an explicit matching between learned and known communities can be modified to obtain a matching. In the modified measure, for each community, we choose the most similar community in the other set to give the matching. While measures that use a threshold such as Qi et al F1 score have the advantage of being robust, until a match crosses a threshold, the measure will not change, making it insensitive to small variations in predictions. Measures with low sensitivity make it difficult to compare algorithms and

select parameters. Weighted measures are more sensitive, giving different values based on the quality of matches, and are more precise when compared to summing binary values of match existence. Accordingly, a more sensitive and precise version of the Qi et al F1 score can be obtained by summing up weights indicating the similarity scores. For instance, instead of the Qi overlap measure, the neighborhood affinity similarity measure can be used to construct a more precise and sensitive measure.

$$Recall\ r = \frac{\sum_{Ck\,\epsilon\,K}\ \underset{C_l \epsilon\ L}{max}\ sim(C_l, C_k)}{|\ K\ |} \tag{2.11}$$

$$Precision\ p = \frac{\sum_{Cl\,\epsilon\,L}\ \underset{C_k \epsilon\ K}{max}\ sim(C_l, C_k)}{|\ L|} \tag{2.12}$$

$$F1\ = \frac{2^{*}p * r}{p + r} \tag{2.13}$$

Here, $sim(C_1, C_2)$ is a similarity measure between communities $C_1$and C_2, with $|C_1|$ is the number of nodes in C_1. C_k is a known community from K, the set of known communities and C_l is a learned community from L, the set of learned communities.

Different similarity measures such as the Jaccard coefficient can be used to construct different F1 measures. We recommend the F-similarity measure in **Figure 2.1b**, as it can be broken down into a precision-based and recall-based measure at the level of comparing a known and learned community, and use it to construct the CMFF score.

2.2.3. Unbiased Sn-PPV Accuracy (UnSPA)

Consider the precision-like positive-predictive value (PPV), recall-like Sensitivity (Sn), and their combined Sn-PPV accuracy (SPA) (**Equations 2.8-2.10**) [21]. In Sensitivity, the numerator is a sum of the maximal number of recalled nodes for each

community and the denominator is a sum of the number of nodes in each community. Measures like these do not give equal importance to each of the known communities and assign higher values for recalling larger communities when compared to recalling smaller communities. For instance, an algorithm that perfectly recalls numerous smaller communities and does not recall much of a few bigger communities can get a worse sensitivity score when compared to an algorithm that does the opposite, *i.e.*, recalls most of the big community and does not recall much of any of the smaller communities. Rather than inducing bias into a measure that decides which communities should be weighted higher, it may be a better idea to have a measure that gives equal weights to all communities. We define an unbiased sensitivity Sn_u in **Figure 2.1e**, by dividing by the total number of known communities.

In PPV, the denominator sums, for each learned community, the sum of the subset of nodes in the learned community shared by all known communities. This does not contribute accurately to a precision-like measure, as nodes that are absent in known communities are ignored. For instance, a learned community that has all the nodes in a known community, but also includes a lot of possibly spurious nodes will be scored in the same way as a learned community which is an exact match to the known community. Further, in PPV, nodes in a learned community shared by multiple known communities get counted an extra number of times in the denominator. So if we share a set of nodes with multiple known communities we get penalized more than (i) if we share the set with only a few known communities, or (ii) if nodes of our community are shared with different known communities in a disjoint manner. The reasoning for allowing such behavior is again biased and does not support the detection of overlapping known communities. For example, a learned community that has a high overlap with 2 known communities (ex: a learned community with 10 nodes that shares all of its nodes with each of the known

communities) will contribute lesser (0.5) to a PPV score than a learned community which overlaps lesser with one known community (ex: 6 nodes in a learned community with 10 nodes overlapping with only one known community, giving a 0.6 contribution to the PPV). To overcome these issues, we propose an unbiased PPV, PPV_u in **Figure 2.1e**, where we divide by the total number of learned communities. The corresponding unbiased accuracy is obtained by taking the geometric mean of the PPV_u and Sn_u as shown in **Figure 2.1e**.

From the sensitivity of measures plot in **Figure 2.1f**, we find that the FMMF score, the Qi et al F1 score, and CMFF score are most sensitive to the pairwise overlap between communities, giving high values at the overlap coefficient yielding the best results, determined via visual inspection of the learned results, as follows. We observed highly overlapping, repetitive, and large numbers of similar learned protein complexes in our experiment on hu.MAP, such as several resembling the ribosome complexes at the high overlap threshold of 0.5 Jaccard coefficient, whereas, at low overlaps, we obtain a total small number of learned complexes, 84 learned complexes after removing proteins absent from known complexes. As we would like a high number of good quality complexes, we find that intermediate values of overlap Jaccard coefficient yield satisfactory results, for instance, at 0.25 Jaccard coefficient, we obtain 121 complexes after removing proteins absent from known complexes, with a high recall of known complexes and good observed quality, *i.e.*, low numbers of very similar overlapping learned complexes. The clique-based measures from [22] - F-grand K-clique and F-weighted K-clique do not vary much with overlap, and the UnSPA, like the SPA, increases with increasing overlap threshold. However, the rate of increase of SPA w.r.t increasing overlap values is greater than UnSPA, yielding comparatively higher scores at undesirable high overlaps. In other words, instead of the desired decreasing trend from 0.25 to 0.5 Jaccard coefficient overlap, we have a highly increasing trend for SPA, compared to the almost constant trend for UnSPA

- an improvement over SPA that can be attributed to the unbiasing modification we have introduced. Therefore, for accurate evaluation in which redundancy (high overlap) is penalized, we recommend UnSPA over SPA, and primarily recommend the FMMF score, CMFF score, and the existing Qi et al F1 score.

2.3. FUTURE RESEARCH DIRECTIONS

2.3.1. Using community fitness functions for evaluating communities

Recall that a community fitness function maps subgraphs - groups of nodes and edges from the full graph, to a scalar value representing a score for the community, where a higher score indicates more community resemblance for the subgraph. If the community fitness function was learned from known communities with high accuracy, it can be used to evaluate communities and determine the performance of a community detection method. An idea to see the aggregate performance would be to take the difference between the average fitness score of the known communities and the average fitness score of the predicted communities, *i.e.*,

$$J(C_p, C_k) = \frac{1}{|C_p|} \sum_{C \in C_p} f(C) - \frac{1}{|C_k|} \sum_{C \in C_k} f(C) \tag{2.14}$$

Here, C_p and C_k are the sets of predicted and known complexes respectively and $f(C)$ is the community fitness function of community C. This can be an especially useful measure in scenarios with limited data, allowing us to compare a large number of predicted communities to a small number of known communities.

To see the exact performance of an algorithm, however, it would be fruitful to make a one-to-one match of the predicted and known complexes and then sum up a measure of their difference, *i.e.*,

$$J(Cp, Ck) = \sum_{(Cpi,Cki)\ \epsilon\ M} D\big(\ f(Cp_i), f(Ck_i)\big) \tag{2.15}$$

Here, M is the set of one-to-one matches between predicted and known complexes and D is a difference function such as an absolute difference or squared difference.

2.3.2. An application: Designing global community detection methods with different evaluation measures

Applications of different evaluation measures discussed in this chapter, apart from final evaluation, include evaluation at the stage of hyperparameter selection, and by extension, incorporation into a loss function that can be used in global community detection algorithms.

Similar to optimization algorithms where the objective or loss function is used to find an optimal solution and its decreasing value can be used to determine the convergence of an algorithm, a loss function can be determined for the problem of sampling subgraphs such that they correspond to communities. Any of the evaluation measures can be used to construct a loss function, for example, using a pairwise similarity measure for all the one-to-one matches between predicted and known communities, we get,

$$J(Cp, Ck) = \sum_{(Cpi,Cki)\ \epsilon\ M}\big(1 - \ sim(Cp_i, Ck_i)\big)\,. \tag{2.16}$$

Consider the greedy modularity maximization algorithm, which starts with each node of a network belonging to a community, and each step joins the pair of communities leading to the largest modularity. Modularity highly scores dense communities, and is given by,

$$J(Cp) = \sum_{(C\ \epsilon\ Cp)} \left[\frac{L_c}{m} - \ \gamma\left(\frac{k_c}{2m}\right)^2\right]. \tag{2.17}$$

Here, m is the number of edges in the network, L_c is the number of edges in community C, k_c is the sum of degrees of the nodes in community c, and γ is a resolution parameter, usually set to 1.

Instead of modularity, using a different evaluation measure (or loss function) gives a greedy community detection algorithm w.r.t to the particular evaluation measure. In this way, supervised community detection algorithms can be formulated - supervised since known training communities are used, either to construct the similarity measure or in the training of the community fitness function.

Chapter 3. Super.Complex - A supervised machine learning pipeline for molecular complex detection in protein-interaction networks[2]

In this chapter, we discuss the community detection method we developed, Super.Complex, which learns an accurate community fitness function from known communities and uses it to detect new communities in parallel on a protein-interaction network. With the help of Super.Complex, we learn accurate human protein complexes from protein interaction networks.

3.1. ABSTRACT

Characterization of protein complexes, *i.e.*, sets of proteins assembling into a single larger physical entity, is important, as such assemblies play many essential roles in cells such as gene regulation. From networks of protein-protein interactions, potential protein complexes can be identified computationally through the application of community detection methods, which flag groups of entities interacting with each other in certain patterns. Most community detection algorithms tend to be unsupervised and assume that communities are dense network subgraphs, which is not always true, as protein complexes can exhibit diverse network topologies. The few existing supervised machine learning methods are serial and can potentially be improved in terms of accuracy and scalability by using better-suited machine learning models and parallel algorithms. Here, we present Super.Complex, a distributed, supervised AutoML-based pipeline for overlapping community detection in weighted networks. We also propose three new evaluation measures for the outstanding issue of comparing sets of learned and known communities satisfactorily. Super.Complex learns a community fitness function from known

[2]This chapter is adapted from published work, Palukuri MV, Marcotte EM (2021) Super. Complex: A supervised machine learning pipeline for molecular complex detection in protein interaction networks. PLoS ONE 16(12): e0262056. https://doi.org/10.1371/journal.pone.0262056. The algorithm development and experiments were performed by me.

communities using an AutoML method and applies this fitness function to detect new communities. A heuristic local search algorithm finds maximally scoring communities, and a parallel implementation can be run on a computer cluster for scaling to large networks. On a yeast protein-interaction network, Super.Complex outperforms 6 other supervised and 4 unsupervised methods. Application of Super.Complex to a human protein-interaction network with ~8k nodes and ~60k edges yields 1,028 protein complexes, with 234 complexes linked to SARS-CoV-2, the COVID-19 virus, with 111 uncharacterized proteins present in 103 learned complexes. Super.Complex is generalizable with the ability to improve results by incorporating domain-specific features. Learned community characteristics can also be transferred from existing applications to detect communities in a new application with no known communities. Code and interactive visualizations of learned human protein complexes are freely available at: https://sites.google.com/view/supercomplex/super-complex-v3-0.

3.2. INTRODUCTION

A protein complex is a group of proteins that interact with each other to perform a particular function in a cell, the basic biological unit of all living organisms. Some examples include the elaborate multiprotein complexes of mRNA transcription and elongation helping with gene regulation and key cytoskeletal protein complexes, such as microtubules with their trafficking proteins which help establish major structural elements of cells. Extensive biological experiments have investigated the physical interactions between proteins, and these have been modeled via weighted protein-protein interaction (PPI) networks, where a protein-protein edge weight corresponds to the strength of evidence for the protein-protein interaction. Disruption of protein-protein interactions often leads to disease, therefore identifying a complete list of protein complexes allows us to

better understand the association of protein and disease. All experimental protocols for detecting complexes (such as AP/MS, affinity purification with mass spectrometry, and CF/MS, co-fractionation with mass spectrometry) have a tendency to miss interactions (false negatives) and may also predict extra interactions (false positives). Proteins may also participate in more than one complex, potentially blurring the boundaries of otherwise unrelated protein communities. Computational analysis of protein-protein interaction networks can therefore be very useful in identifying accurate protein complexes and will help augment and direct experimental methods.

The weighted PPI network or graph G can be represented as pairs of nodes and edges (*V, E)*, where the set of nodes or vertices *V* represents the proteins, and the set of weighted edges *E* represents the strengths of evidence for interactions between proteins. Any group of nodes and edges that can be characterized as a protein complex can be referred to as a community; community detection methods can be used in turn to identify protein complexes.

A standard guideline for defining communities [1] is that a community should have more interactions or connectivity among the community than with the rest of the network. This can be modeled for example by a community fitness function, mapping a subgraph, *C, i.e.*, a group of nodes and edges from the full graph, to a scalar value representing a score, where a higher score indicates more community resemblance, for example,

$$f(C) = \delta int\,(C) - \delta ext\,(C)\,. \tag{3.1}$$

The intra-cluster density $\delta int\,(C)$ and inter-cluster density $\delta ext\,(C)$ are given by,

$$\delta int\,(C) = \frac{\#\ intra-cluster\ edges}{\#\ of\ all\ possible\ edges\ in\ the\ cluster} = \frac{m_c}{\frac{n_c(n_c-1)}{2}} \tag{3.2}$$

$$\delta ext\ (C)\ = \frac{\#\ inter-cluster\ edges}{\#\ of\ all\ possible\ inter-cluster\ edges} = \frac{\#\ inter-cluster\ edges}{n_c(n-n_c)} \qquad (3.3)$$

Here, n_c and m_c are the numbers of nodes and edges in subgraph *C*, respectively, and *n* is the number of nodes in graph *G*.

However, there exist many communities that do not follow this criterion but can be identified by the different properties they exhibit. One such example is a star-like topology, where one central node interacts with several nodes in yeast protein-interaction networks [2], as, for example, in the case of a molecular chaperone that acts on a number of separate protein clients. In the case of human protein complexes, we also observe different topologies such as clique, linear, and hybrid between linear and clique, as shown in **Figure 3.1**. These human protein complexes represent proteins known to belong to experimentally characterized gold-standard protein complexes from CORUM 3.0 (the comprehensive resource of mammalian protein complexes) [3] with edge weights from hu.MAP [4], a human protein interaction network with interactions derived from over 9,000 published mass spectrometry experiments.

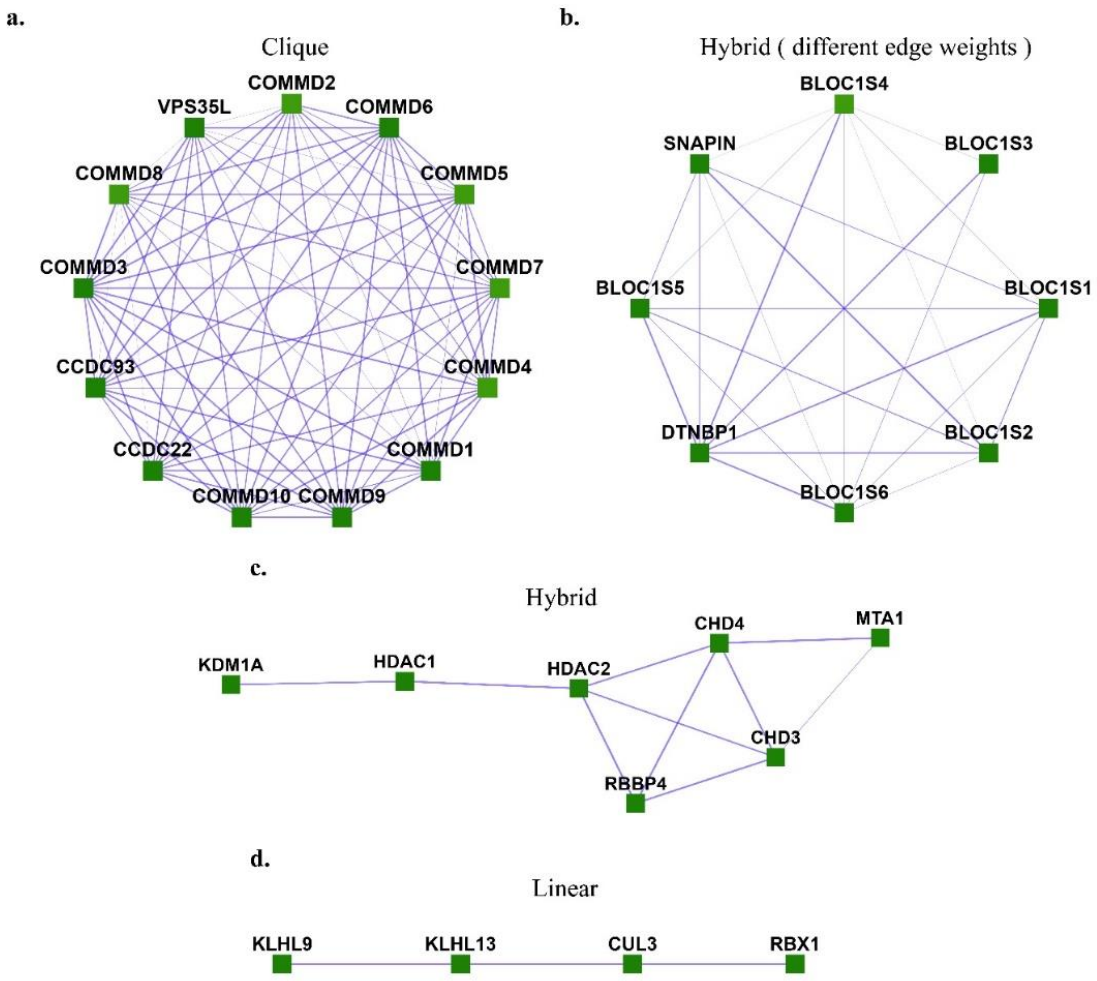


Figure 3.1. Different topologies are exhibited by human protein complexes.

a. Clique (Commander/CCC complex), b. Hybrid with different edge-weights (BLOC-1 (biogenesis of lysosome-related organelles complex 1)), c. Hybrid (NRD complex (Nucleosome remodeling and deacetylation complex), d. Linear (Ubiquitin E3 ligase (CUL3, KLHL9, KLHL13, RBX1)). These are experimentally characterized complexes from CORUM [23] with protein interaction evidence obtained from hu.MAP [22].

Existing community detection methods have primarily tried to optimize for high scores of community fitness functions, maximizing density [3]. These include unsupervised methods, such as implemented by MCL- Markov Clustering [6], MCODE - Molecular COmplex Detection [5], Cfinder [4], SCAN- Structural Clustering Algorithm for Networks [8], CMC - Clustering based on Maximal Cliques [7], COACH - COre-AttaCHment based method [23], GCE - Greedy Clique Expansion [3], and ClusterONE - clustering with overlapping neighborhood expansion [9], as well as semi-supervised machine learning algorithms such as COCDM - Constrained Overlapping Complex Detection Model [10].

When there are sufficient data available on known communities, rather than applying a generic community fitness function to the problem, it can be more accurate to learn a community fitness function directly from known communities. Then, new communities detected with the learned community fitness function can be expected to better resemble known communities in the field. Supervised machine learning methods are well suited for this purpose, and a few methods have been used to learn a community fitness function from constructed community embeddings, *i.e.*, community representations in vector space, obtained by extracting topological and domain-specific features from communities. The community fitness function learned can then be used to select candidate communities from the network and evaluate them. Since finding maximally scoring communities in a network is an NP-hard (non-deterministic polynomial-time hard) problem [18], heuristic algorithms have been used to find candidate communities. A common strategy is to select a seed (such as a node or a clique) and grow it into a candidate community by iteratively selecting neighbors to add to the current subgraph using heuristics such as iterative simulated annealing until a defined stopping criterion is met for

the growth process. This process is repeated with different seeds to generate a set of candidate communities.

Existing supervised methods (SCI-SVM, SCI-BN, RM, ClusterSS, ClusterEPs, NN) use different machine learning methods to learn the community fitness function after extracting different features and use different heuristic algorithms to select candidate communities. Regarding scalability, the above methods have generally only been implemented on small yeast protein complex datasets, except for ClusterEPs, which trains on yeast data and tests on human PPIs. SLPC's [15] regression model was implemented on a human PPI network re-weighted by breast-cancer specific PPIs extracted from biomedical literature to detect disease-specific complexes [17]. However, these methods employ serial candidate community sampling, negatively impacting their scalability to large networks such as hu.MAP [22], a human protein-interaction network with ~8k nodes and ~60k edges.

In this work, we present Super.Complex (short for Supervised Complex detection algorithm), an end-to-end highly scalable (scaling to large networks that fit on a disk), distributed, and efficient community detection pipeline that explores multiple supervised learning methods with AutoML (Automated Machine Learning) to learn the most accurate community fitness function from known communities. Super.Complex then samples candidate subgraphs in parallel by seeding nodes or starting with maximal cliques and growing them with an epsilon-greedy heuristic, followed by an additional heuristic such as iterative simulated annealing or pseudo-metropolis using the learned community fitness function. On a yeast PPI network, Super.Complex outperforms all 6 existing supervised methods, as well as 4 unsupervised methods. Three novel evaluation measures are proposed to overcome certain shortcomings of existing metrics. We apply Super.Complex to hu.MAP, a human protein-protein interaction network with ~8k nodes and ~60k edges

to yield 1028 protein complexes, including high-scoring previously unknown protein complexes, potentially contributing to new biology, and make all data, code, and interactive visualizations openly and freely available at https://sites.google.com/view/supercomplex/super-complex-v3-0.

3.3. MATERIALS AND METHODS

3.3.1. Overview of Super.Complex

The pipeline Super.Complex comprises two main tasks, first, learning a community fitness function with AutoML methods, and second, using the community fitness function to intelligently sample overlapping communities from a network in parallel. As shown in **Figure 3.2**, each task is subdivided into different steps, described in brief in this section, with all details in the following sections of Materials and Methods. For the first task, we perform a pre-processing step, Data Preparation, where known communities are cleaned and split into non-overlapping training and testing sets, followed by the construction of training and testing negative community data. In (i) Topological Feature Extraction, topological characteristics for all communities are computed to construct training and testing feature matrices. AutoML (ii) then compares different ML (Machine Learning) pipelines to select the best one, followed by training and testing the best ML pipeline, thus learning the community fitness function as the binary classifier distinguishing positive communities from negatives. Having learned the community fitness function, Super.Complex then uses it in its heuristic algorithm for the second task of searching for candidate communities in the network in parallel. For (iii) intelligent sampling, the algorithm can start with either single nodes or maximal cliques as seeds. We note that all nodes of the network were used as seeds in our experiments (this is quite fast due to Super.Complex's parallel implementation), allowing us to work without any estimate of

the number of expected communities. These seeds are grown using a 2-stage heuristic, *e.g.* ϵ-greedy + iterative simulated annealing. This is followed by (iv) a post-processing step of merging highly overlapping communities. Finally, in the last step, evaluation, the learned communities are compared with known communities. The steps of the pipeline are fairly independent and can be improved on their own with methods to test the accuracy/performance of each of the steps.

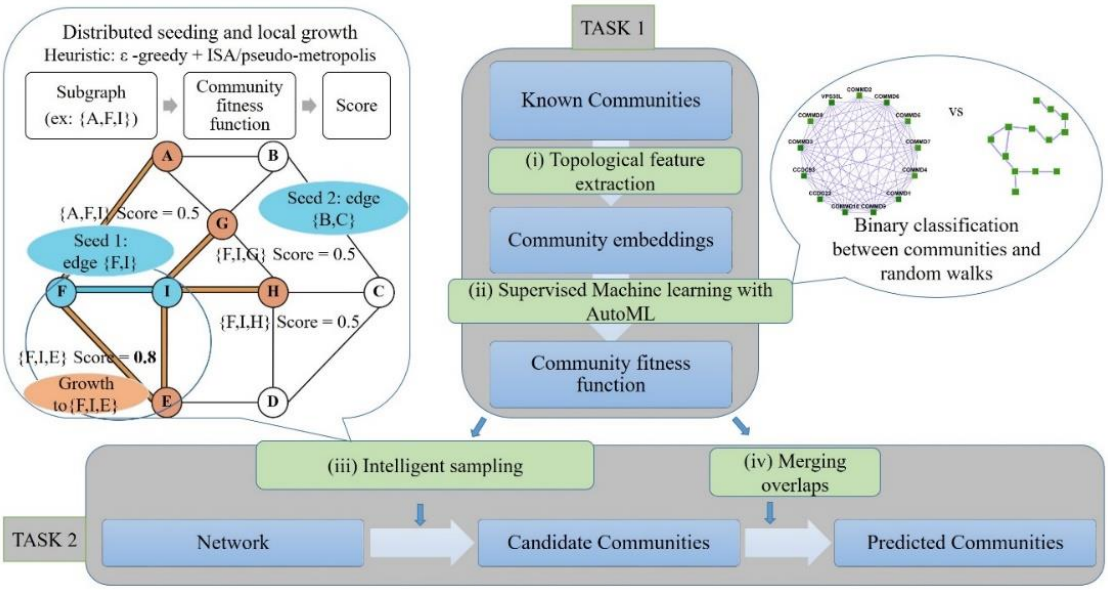


Figure 3.2. Super.Complex identifies likely protein complexes within a PPI network using a distributed supervised AutoML method.

Task 1: Learning a community fitness function: (i) Topological feature extraction: Topological features are extracted from known communities to build community embeddings (feature vectors, which are representations of communities in vector space) (ii) Supervised learning with AutoML: A score function for communities, the community fitness function, is learned from the community embeddings as the decision function for binary classification of a network subgraph as a community or a random walk (illustration on the right). The best score function is selected after training multiple machine learning models with TPOT [24], an AutoML pipeline. Task 2: Searching for candidate communities in the network: (iii) Intelligent sampling: Multiple communities are sampled in parallel from the network. To build each candidate community, a seed edge is selected and grown using a 2-stage heuristic. First, we use an epsilon-greedy heuristic to select a candidate neighbor, and then we use a pseudo-metropolis (constant probability) or iterative simulated annealing heuristic to accept or reject the candidate neighbor for growing the current community. An iteration of neighbor selection using a greedy heuristic is shown (illustration on the left), starting from a seed edge {F, I}. The edge is grown to the subgraph {F, I, E} as adding node E yields a higher community fitness function than adding any other neighbor of F and I. The seed edge {B, C} is grown in parallel (not shown) (iv) Merging overlaps: The candidate communities are merged such that the maximum overlap between any 2 communities is not greater than a specified threshold.

3.3.2. Data preparation

First, the weighted network under consideration is cleaned by removing self-loops, as we do not consider interactions with oneself as a feature of communities. For scalability, the graph is stored on disk as a set of files, each corresponding to a node and containing a list of the node's neighbors via weighted edges.

3.3.2.1. Positive communities

Super.Complex takes sets of nodes comprising known communities and obtains their edge information from the induced subgraph of these nodes on the weighted network. Nodes in communities that are absent from the network are removed. Communities with fewer than 3 nodes, communities that are internally disconnected, and duplicate communities are also removed. Constructing the final set of positive communities involves 2 main steps: (i) merging similar communities, and (ii) splitting them into non-overlapping train and test sets. Note that if independent train and test sets of communities are known in advance, these steps can be skipped.

In the first step, using a merging algorithm we devised, we merge highly similar communities to yield a final list of communities, where no pair of communities have a Jaccard score greater than or equal to *j*. We recommend users set this value based on domain knowledge of observed redundancy in the set of known communities.

Multiple solutions exist that achieve this goal, however, we want a solution with a large number of communities, *i.e.*, with only a small number of merges performed on the original set of known communities. This is especially important in applications with limited data, such as the human and yeast protein complex experiments in this work. Our algorithm was designed with this objective in mind and works as follows. The iterative algorithm makes multiple passes through the list of communities performing the merging operation

until the specified criterion is achieved. In a single pass of the list of communities, each community is considered in order and merged with the community with which it has the highest overlap (if greater than or equal to *j*) and the list is updated immediately by removing the original 2 communities and adding the merged community to the end of the list so that the updated list is available for the next community in consideration. This merging algorithm achieves a lesser number of merges than a trivial merging solution which would merge random pairs of communities that do not satisfy the required criteria until convergence. In practice, the proposed algorithm quickly converges to a solution (*i.e.*, the final set has no communities that overlap more than the specified value *j*).

In the second step, the communities are split into non-overlapping training and testing datasets, to emphasize their independence. We obtain sets with equal size distributions and a 70-30 train-test split, as recommended for machine learning algorithms with a small amount of data. Previous algorithms such as [22] and Super.Complex v2.0 [25] discard test communities with sizes greater than a threshold, thus losing out on information from some known communities and which also, in practice, do not yield train-test splits that are close to the recommended 70-30 split. Therefore, we propose the following algorithm. Here, we first make the recommended 70-30 random split into train and test communities. Then we perform iterations of transfers between the two sets until they become independent. In each iteration, we perform two directions of transfers, from train to test and vice-versa, and if the 70-30 split is disturbed, we remove the communities at the end of the list which have extra communities and add them to the other list. In each direction of transfer, for instance, from train to test, we go through the training communities in one pass and if a training community has an overlap (at least one edge) with any of the test communities, it is immediately transferred to the test set, making the updated test set available for comparison with subsequent training set communities. In practice, for many

random splits, the algorithm converges fast enough to a solution that is non-overlapping. If for an initial random split, convergence is not achieved after a few iterations, we recommend restarting the algorithm with a different random split.

3.3.2.2. Negative communities

Negative communities, *i.e.,* non-communities are represented by random walks sampled from the network by growing random seeds, adding a random neighbor at each step. The number of steps ranges from the minimum size to the maximum size of positive communities, with a total number of random walks equal to the number of positive communities multiplied by a scale factor > 1. The random walks are split almost equally across all the sizes, by splitting equally across the different number of steps to be taken for a random walk, to yield an almost uniform size distribution for negative communities. We say almost uniform size distribution, as random walks with the same number of steps need not yield the same sizes, given that the random walk as defined here can revisit edges it has already visited. To achieve random walks of the same size, the algorithm attempts an extra number of random walks and an extra number of steps to achieve the desired random walk size.

The size distribution of positive communities is taken into consideration while training the machine learning model when using a uniform distribution for negatives. We also explore using almost the same size distribution as the positive communities to construct the negative communities. For this, for each size of the positive communities, we construct the negatives by sampling a number of random walks equal to the scale factor times the number of positive communities of this size. However, in this case, we find that there are quite a few missing sizes due to limited positives which may affect the scoring of subgraphs of the missing sizes. Using a uniform distribution would provide more

information to learn a more accurate community fitness function that can recognize negatives at sizes missing for positives. In the following feature extraction step, random walks resembling communities are removed. The final number of negative communities is close to the number of positive communities, as we have sampled a slightly higher number of random walks via the scale factor.

3.3.3. Topological feature extraction

As communities exhibit different topological structures on the graph, these can be learned by considering useful topological features of communities. Based on graph theory, we extract 18 topological features, detailed in **Supplementary Methods** (Topological features) for each of the positive/negative communities to construct the final train and test data feature matrices, *i.e.*, the positive and negative community embeddings.

3.3.4. Learning the community fitness function with AutoML

A community fitness function is learned as the decision function of a binary machine learning classifier trained to distinguish the community and non-community embeddings constructed in the previous feature extraction step. For this, we use an AutoML algorithm, TPOT [24], a genetic algorithm that yields the best model and parameters. It evaluates several preprocessors along with ML models and yields cross-validation scores on the training dataset for each pipeline, which itself is usually a combination of several preprocessors followed by the machine learning model. We configure the algorithm to run in a distributed setting, exploring several combinations of several preprocessors and ML models.

We specify 6 pre-processors that scale the feature matrix. These are - (i) Binarizer, which sets a feature to 0 or 1 based on a threshold, (ii) MaxAbsScaler, which divides the

feature by the maximum absolute value of the feature, (iii) MinMaxScaler, which subtracts the minimum of the feature from the feature vector and divides by the range of the feature, (iv) Normalizer, which divides the feature vector by its norm to get a unit norm, (v) RobustScaler, which makes a feature robust to outliers by scaling using the interquartile range and (vi) StandardScaler, which standardizes to the Z-score by subtracting the mean and dividing by the standard deviation of the feature.

We include four feature selecting pre-processors, which are additionally important as we incorporate 6 additional preprocessors that construct combined features. The additional preprocessors include - (i) Decomposition: PCA (Principal Component Analysis), FastICA (Independent Component Analysis), (ii) Feature Agglomeration, (iii) Kernel Approximation methods: Nystroem, Radial Basis Function RBFSampler, (iv) Adding Polynomial Features, (v) Zero counts: Adds the count of zeros and non-zeros per sample as features and (vi) OneHotEncoder for numeric categorical variables. The feature selecting preprocessors include - (i) SelectPercentile, which selects the highest-scoring percentage of features based on 3 univariate statistical tests, FPR - False Positive Rate, FDR - False Discovery Rate, and FWE - Family-wise error rate; (ii) VarianceThreshold which removes low variance features, (iii) RFE (recursive feature elimination) using ExtraTrees and (iv) SelectFromModel using ExtraTrees based on importance weights. The ML models included are - (i) Naive Bayes methods using Gaussian, Bernoulli, and Multinomial distributions (ii) Decision Trees, (iii) Ensemble methods of ExtraTrees, Random Forest, Gradient Boosting, and XGB (XGBoost), (iv) K-nearest neighbors, (v) Linear SVMs and (vi) Linear models for Logistic Regression.

The population size and number of generations are provided as parameters for the genetic algorithm of the AutoML pipeline. In practice for our application, giving a value of 50 for each yielded good results. There is an option for a warm start, where you can run

additional generations and with additional population sizes starting from the latest results if the results are unsatisfactory. Additionally, several other machine learning models and preprocessors can also be incorporated into this pipeline, including neural networks. Note that in our experiments, we also obtained pipelines that stack different ML models. We run the pipeline in a distributed manner, setting the number of jobs as the number of processes that run in parallel on a single computer. All the processes on the computer can be used for maximum utilization, however, the documentation notes that memory issues may arise for large datasets. In practice, we set the number of jobs as 20 on a Skylake compute node (Intel Xeon Platinum 8160 with 48 cores @2GHz clock rate).

3.3.5. Evaluation

By default, 5-fold cross-validation is performed, although this can be modified by a parameter. The pipelines with high cross-validation average precision scores (area under the PR curve) are evaluated on the test dataset to find the best pipeline for our data, to use this for the community fitness function. A one hidden-layer perceptron is also available for training, and comparison with the AutoML output to select the best model. We evaluate the performance of the ML binary classifier using accuracies, precision-recall-f1 score measures, average precision score, and PR curves for the test sets while also evaluating these measures for the training set to compare with the test measures and check the bias and variance of the algorithm to make sure it is not underfitting or overfitting the data. We also plot the size-wise accuracies of the model to understand how a model performs w.r.t to the size of the subgraph it is evaluating.

3.3.6. Candidate community search

Finding a set of maximally scoring candidate communities in a network is an NP-hard problem, as can be proved by reducing it to the problem of finding maximal cliques [18]. Since this is an NP-hard problem, algorithms based on heuristics are required to solve it. We explore seeding and growth strategies.

3.3.6.1. Design and distributed architecture

First, we need to select seeds. Options for seeds include specifying all the nodes of the graph (recommended for best accuracy), all the nodes of the graph present in known communities, a specified number of nodes that will be selected randomly from the graph, or maximal cliques. In the distributed setting using multiple compute nodes, the specified seeds are partitioned equally across compute nodes, and each compute node deals only with the task of growing the seeds assigned to it. In practice, the partitioning is done by a main compute node that partitions the list of seeds and stores the partitioned lists as separate files on the file server. Then it launches one task per compute node (including itself) using the launcher module [26], where a task instructs a compute node to read its respective file containing the seed nodes and run the sampling algorithm starting with each of the seed nodes. On each compute node, we take advantage of all the cores by employing multiprocessing with the *joblib* python library.

Each process intelligently grows a single seed node into a candidate community and writes it to the compute node's temporary storage. For this, we need the graph and parameters of the community fitness function, which we store on the temporary disk space of each compute node to optimize RAM as it is impractical to store large networks and machine learning models in memory. Each process reads the model into its memory and uses it to evaluate the neighbors, to pick the neighbor to add to the current subgraph in the

growth process from the seed node. The neighbors of the subgraph under consideration at each step of the growth are read from disk on-demand and stored in memory only until they have been evaluated by the fitness function. In this way, we ensure that the processes have a low memory footprint, which can otherwise quickly become a bottleneck for large graphs. We also minimize disk storage by storing each resulting candidate community compactly using only its nodes, as its edges can be inferred if/when necessary by inducing the nodes on the graph. After the completion of execution of all the child processes of growing seeds on a compute node, the compute node reads the set of learned community files it had stored on its disk and compiles them into a list of candidate communities before writing the list to the file server. The same code also runs in a distributed setting with only one multi-core compute node. There also exists a serial option to run the code without invoking parallel constructs, useful for running on a single core.

3.3.6.2. Intelligent sampling heuristics

Only for the first step of growth, we add the neighbor connected with the highest edge-weight. We provide 2 options for growing the subgraph at each step- an exhaustive neighbor search that is suitable for graphs that are not very large, and an option that optimizes performance by evaluating only a subset of neighbors. In the latter, using a large user-defined threshold t_1, if the number of neighbors of the current subgraph is greater than the threshold, a random sample of the neighbors equal to the provided threshold is chosen for evaluation. Now, of the neighbors, first, an ϵ-greedy heuristic is used to select the neighbor to add to the subgraph. In an ϵ- greedy heuristic, with ϵ probability, a random neighbor is added instead of the maximum scoring neighbor.

In the non-exhaustive search case, in the event of 1-ϵ probability, if the number of neighbors is greater than a 2nd user-defined threshold t_2, a 2nd optimization of cutting

down the number of neighbors is applied before evaluating each of the neighbors for choosing the greedy neighbor, as follows. Here, the t_2 highest neighbors are chosen for evaluation, where the order is decided by sorting the neighbors in descending order based on their maximum edge weight (*i.e.*, the highest edge weight among all the edges connecting a neighbor to the subgraph). Note how the first threshold $t_1$ensures that the sorting complexity $O(t_1 log(t_1))$does not blow up.

Note that for efficient constant-time $O(1)$ lookup of the maximum edge weight of a neighbor, we store the neighbors of the subgraph as a hash map, where looking up a neighbor yields its maximum edge weight. This hash map also stores, for each neighbor, a list of edges connecting it to the subgraph and was constructed efficiently when the neighbors of each of the subgraph nodes were read from the corresponding file. After selecting the neighbor to add to the subgraph in the current iteration, this hash map is also used to efficiently add the neighbor to the subgraph by providing a constant-time lookup to the edges that need to be added.

Instead of the base ϵ-greedy heuristic, we also have a simple base heuristic option, termed greedy edge weight, where we add the neighbor with the highest maximum weight edge at each step of the iteration. Note that since the ML model is not used at each stage of the growth, this is fast enough and does not require the optimization steps used in the ϵ-greedy approach where subsets of neighbors were selected for evaluation by the community fitness function.

For both base heuristics, in any iteration, if no neighbors for the subgraph exist, the growth process terminates. If the community score of the subgraph in any iteration is less than 0.5, the node last added is removed and the growth process terminates. We provide additional heuristics that can be applied on top of the base ϵ-greedy heuristic. Based on the scores of the current and previous iterations of the subgraph, we accept or reject the latest

node addition using the user-defined heuristic - iterative simulated annealing (ISA), or a variant of ISA, termed pseudo-metropolis in which the acceptance probability is a constant, *i.e.*, $P(S_{new}, S_{old}) = k$. In ISA, at each stage of growth of the current subgraph, its maximum scoring neighbor is added, except in the case when the new community score of the subgraph $S_{new}$is lesser than S_{old}, the value before adding the new node (*i.e.*, $S_{new} < S_{old}$). In this case, the new node addition is accepted with a probability of,

$$P\left(S_{old,} S_{new,} T\right) = e^{\frac{(S_{new} - S_{old})}{\mathrm{T}}}, \tag{3.4}$$

here, starting with hyperparameters T_0 and α, we update the temperature as $T \leftarrow \alpha T$ after every iteration.

When ISA or pseudo-metropolis heuristics are applied, we also evaluate an additional heuristic where the algorithm terminates if it has been 10 iterations (this can be a user-defined number) since the score of the subgraph has increased.

In the implementation, we provide four options to the user - greedy edge weight, ϵ-greedy, ϵ-greedy + ISA, and ϵ-greedy + pseudo-metropolis. In all options, the algorithm terminates after a number of steps equal to a user-specified threshold. The default threshold provided is the maximum size of the known communities, and we also provide a smart option for when a few communities have a large number of nodes, where it is set to choose the maximum size after ignoring outliers. This number can also be improved by visual inspection of a generated boxplot of community sizes. Future work can also explore greedy edge weight + ISA and greedy edge weight + pseudo-metropolis heuristic algorithms and observe their performance. Note how there are 2 possibilities for exploration in the 3 algorithms other than the greedy edge weight heuristic algorithm. In the 1st stage, we pick a neighbor at random with low probability. In the 2nd stage, we accept the neighbor we picked in the 1st stage with low probability, if it yields a lower score than the original subgraph.

3.3.7. Post-processing (merging overlaps) and cross-validation

Communities with only 2 nodes are removed. Note that communities with 2 nodes are rarely found, and while dimers are biologically valid, since they do not have topological variation, we do not consider them in this work focused on higher-order assemblies with different topologies as a key feature. We then merge communities that have a Jaccard similarity greater than a specified overlap threshold employing the merging algorithm discussed in the data cleaning section. The only difference is while merging, for two overlapping communities, the final community retained out of the 2 communities or the merged variant is the one that obtains the highest score with the community fitness function. In another variant of the merging algorithm, instead of the Jaccard similarity threshold, we use Qi's overlap measure.

The parameters ϵ in the ϵ-greedy heuristic, k in pseudo-metropolis, T_0 and α in iterative simulated annealing and the overlapping threshold in the post-processing step, are varied in parameter sweeps to select the best ones using the Qi et al F1 score (**Equation 2.7**).

After parameter sweeps, the results of different heuristics are examined and the one that yields the best F1 score is chosen. Additional details regarding evaluation are outlined in **Supplementary Methods** (Evaluation with existing measures).

3.4. RESULTS AND DISCUSSION

3.4.1. Contributions of Super.Complex - a scalable, distributed supervised AutoML-based community detection method

Super.Complex implements an original distributed architecture and an efficient pipeline, scaling to large networks such as hu.MAP with ~8k nodes and ~60k edges. With an AutoML method, which also includes automated feature selection, and four 2-stage

heuristic options for candidate community search, the pipeline finds accurate community fitness functions and high-quality communities. Unlike some existing methods that remove nodes in the process of growth (e.g. such as Louvain [27] and ClusterSS), we note in **Supplementary Results** (Algorithm guarantees) that our method guarantees properties such as internal connectivity of communities. Further, the merging algorithm we employ guarantees that no two communities overlap more than a specified threshold. In the case of non-overlapping communities (obtained by specifying a merging threshold of 0 overlap), there is an additional guarantee that no two communities can be merged to yield a higher scoring community. To our knowledge, epsilon-greedy heuristics in conjunction with other heuristics such as iterative simulated annealing have not been applied in the past for community detection. This allows the pipeline to leverage the advantages of both heuristics by adding stochasticity, allowing better exploration in the candidate community search stage. Super.Complex has a cross-validation pipeline to select the heuristic and parameters that work best for the application at hand. Minimal hyper-parameter selection is required in our algorithm with default parameters provided when smart hyperparameters cannot be inferred.

Since the number of known communities can be limited, we emphasize the preservation of known communities when splitting them into train-test sets while also ensuring (i) independence - *i.e.*, no edge overlap between a train and test community on the network, (ii) similar size distributions for both sets, and (iii) 70-30 ratios in train-test sets. Similarly, a minimal number of merges is attempted in the merging algorithms devised to maintain a high number of learned protein complexes. Further, unlike existing supervised methods, which evaluate the performance of their algorithms on a reduced network with only nodes present in known communities, we evaluate our algorithm on the full network for a more accurate evaluation. Finally, we note that Super.Complex uses only

topological features of networks, and can be applied to community detection on networks from various fields, with the possibility of including domain-specific features to learn more accurate domain-specific community fitness functions. Our methods are also applicable in domains with limited or no knowledge by transferring community fitness functions from other domains, such as the defaults we provide for human protein complex detection.

3.4.2. Super.Complex applied to a human protein interaction network to detect protein complexes

3.4.2.1. Experiment details

We first test and ensure that the pipeline achieves perfect results on a toy dataset we construct comprising disconnected cliques of varying sizes, each corresponding to a known community, where we use all nodes as seeds for growth during the prediction step.

To learn potentially new human protein complexes, we apply Super.Complex on the human PPI network hu.MAP [22] using a community fitness function that is learned from known complexes in CORUM [28]. The network available on the website (http://hu1.proteincomplexes.org/static/downloads/pairsWprob.txt) has 7778 nodes and 56,712 edges, after an edge weight cutoff of 0.0025 was applied to the original 64,048 edges. There are 188 complexes after data cleaning, a set we term as 'refined CORUM', out of the original 2916 human CORUM complexes, which underscores the importance of minimizing any losses in the merging and splitting steps of the pipeline. In the data cleaning process, overlapping complexes with a Jaccard coefficient j greater than 0.6 are removed, as this value was used in the experiments of hu.MAP 2-stage clustering. Note that of the complexes from CORUM that were removed, there were over 1000 complexes that had fewer than 3 members, and the remaining removed complexes consisted of duplicates and disconnected complexes with edges from hu.MAP. Note, however, that hu.MAP was the

highest confidence human protein interaction network integrating 3 large previous human protein-interaction networks, all built using high confidence data from large-scale (~9000) laboratory experiments. The edge weights of hu.MAP were trained using an SVM based on features obtained from experiments.

3.4.2.2. Experiment results

The best results, following different parameter sweeps from the experiment on hu.MAP are given in **Figure 3.3** with the best parameter values given in **Table 3.1**. From **Figure 3.3e**, we verify that the size distributions of the train and test sets are similar. In **Figure 3.3a**, we can see that we get a good precision-recall curve on the test set for the subgraph classification task as a positive or negative community, achieving an average precision score of 0.88 with a logistic regression model (which is the final ML model stacked on a set of other ML models and processors, output as the best model trained on the training set with 5-fold cross-validation and achieving a cross-validation score of 0.978).

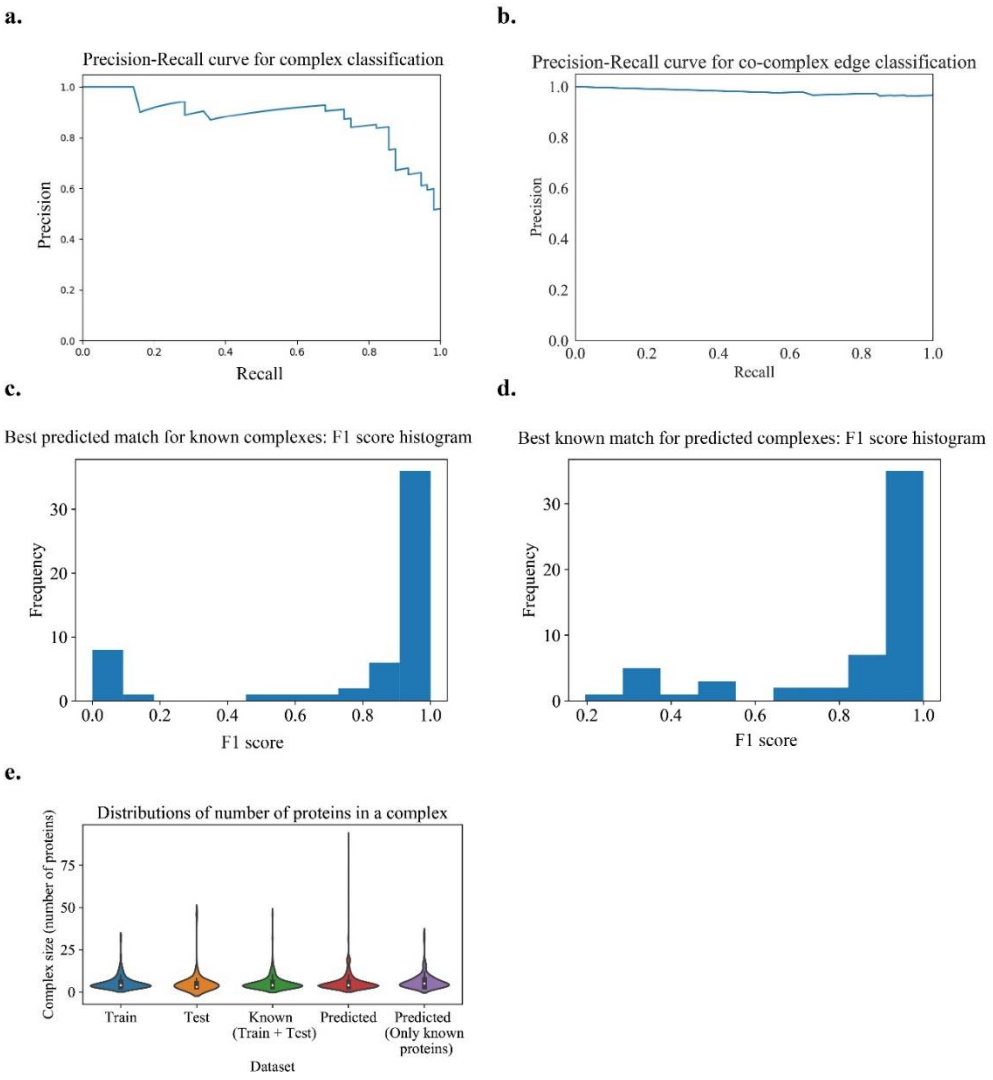


Figure 3.3. Learned human protein complexes with Super.Complex achieve good PR curves and follow similar size distributions as known complexes.

a. PR curve for the best model (community fitness function) from the AutoML pipeline on the test dataset, for the task of classifying a subgraph as a community or not. **b.** Co-complex edge classification PR curve for final learned complexes. **c & d.** Best F-similarity score distributions per known complex and per learned complex. **e.** The size distributions of train, test, and all known complexes, learned complexes and learned complexes after removing known complex proteins.

Table 3.1. Best parameters found and used in each of the experiments.

PPI Network:	**hu.MAP**	**Yeast**	**Yeast**	**Yeast**
Experiment	train: CORUM, test: CORUM (independent)	1. train: TAP, test: MIPS	2. train: MIPS, test: TAP	3. train: MIPS, test: MIPS
Seeds	All nodes	All nodes	All nodes	All nodes
No. of negatives sampled	10x positives	1.1x positives	1.1x positives	1.1x positives
Size (no. of nodes) distribution for negatives	Uniform	Uniform	Uniform	Uniform
Candidate sampling method	ϵ- greedy + iterative simulated annealing	ϵ- greedy + iterative simulated annealing	ϵ- greedy + pseudo-metropolis	ϵ- greedy + iterative simulated annealing
ϵ	0.01	0.01	0.01	0.01
Sampling method parameters	T0 = 1.75 and α = 0.005	T0 = 0.88 and α = 1.8	Probability p = 0.1	T0 = 0.88 and α = 1.8
No. of steps (specified or inferred from known complexes)	20	4	9	10
Neighbors considered for growth	All neighbors	All neighbors	All neighbors	All neighbors
Merging method and parameter	Qi overlap measure = 0.375	Qi overlap measure = 0.1	Qi overlap measure = 0.3	Qi overlap measure = 0.9

We use **Figure 3.3e** to set the maximum number of steps taken in the candidate complex growth stage as 20 and learn a total of 1028 complexes. On removal of non-gold standard proteins from these complexes for evaluation purposes, we obtain 131 complexes. Comparing learned co-complex edges with known co-complex edges, we obtain a good PR curve (**Figure 3.3b**), as we observe that the curve goes from the top left corner (precision = 1 at recall = 0) to close to the top right corner (precision = 0.97 at recall = 1), with an area under the curve (PR-AUC) of 0.98, close to the ideal value of 1. From **Figure 3.3c**, we can see that the best learned complex matches for known complexes have high F-similarity scores. Also, from **Figure 3.3d**, we can see that the best known complex matches for learned complexes have high F-similarity scores. Note that there may be unknown but true complexes that are learned by the algorithm that contribute to false positives.

In **Figure 3.3e**, we can see that learned complexes have a similar size distribution as known complexes. We also observe a small peak in the size distribution of learned complexes at size 20. The peak occurs due to some complexes growing to the specified maximum number of steps (here, 20) without encountering other stopping criteria. Due to the stopping criteria on the score (value of the community fitness function) of the complexes, all learned complexes (i) have a score greater than or equal to 0.5, and (ii) have an observed score improvement during their growth process, at least once every 10 steps.

Evaluation measures comparing learned complexes on hu.MAP by Super.Complex w.r.t known complexes from CORUM are given in **Table 3.2 and (Supplementary) Table 3.4**, along with the measures computed on the protein complexes comprising hu.MAP obtained from a 2 stage clustering method with the unsupervised ClusterONE algorithm applied first, followed by the unsupervised MCL algorithm. We observe that Super.Complex does better in terms of precision, as can be seen with the higher FMM precision value, while ClusterONE+MCL does better in terms of recall. This can be

attributed to more number of complexes learned by ClusterONE+MCL (~4000 compared to ~1000 by Super.Complex) including a few highly overlapping complexes (the maximum pairwise overlap observed was 0.97 Jaccard coefficient), compared to the strict low overlap among complexes learned by Super.Complex (the maximum pairwise overlap observed was 0.36 Jaccard coefficient). We observe 4152 pairs of complexes learned by ClusterONE + MCL having an overlap greater than 0.36 Jaccard coefficient, the maximum pairwise overlap observed in learned complexes from Super.Complex. Note that while the values of F1 evaluation measures are similar, the results from ClusterONE+MCL were achieved by the authors after significant cross-validation, while Super.Complex was faster as detailed in **Supplementary Results** (Performance).

Table 3.2. Evaluating learned complexes on hu.MAP w.r.t 'refined CORUM' complexes.

Method	FMM			CMF F1 score	Unbiased Sn-PPV accuracy	Qi et al F1 Score (t=0.5)	F-weighted k-Clique
	Precision	Recall	F1 score				
Super.Complex	**0.767**	0.534	**0.63**	0.783	0.888	0.739	**0.972**
hu.MAP (ClusterONE + MCL)	0.471	**0.686**	0.559	**0.797**	**0.911**	**0.764**	0.967

Refined CORUM comprises 188 complexes after cleaning the original CORUM complexes. The novel metrics FMM, CMF, and UnSPA are from **Figure 2.1**. F-weighted K-clique [22] and Qi et al F1 score (**Equation 2.7**) are existing metrics.

3.4.3. State of the art comparison: Super.Complex achieves good evaluation measures and performance

To compare our method with published results from existing methods, we perform experiments on the data used by these methods in their experiments - a yeast PPI network, DIP -Database of Interacting Proteins [29] with known protein complexes from MIPS - Munich Information Center for Protein Sequence [30] and TAP- Tandem Affinity Purification [31]. Specifically, for an accurate comparison, we use the same PPI network (projection of DIP yeast PPI network on MIPS + TAP proteins) and known protein complexes, available from the ClusterEPs software website. The results from **Table 3.3** show that our method outperforms all 6 supervised as well as 4 unsupervised methods (by achieving the highest F1 score and precision values) in the yeast experiments. Specifically, Super.Complex achieves the highest F1 score value (87% higher on average, 63% higher by median) when compared to the 10 other methods, the highest precision value (110% higher on average, 72% higher by median) when compared to the 10 other methods, higher recall (92 % higher on average, 45% higher by median) when compared to 8 other algorithms with lower recall values (30% lower on average and by median) when compared to only 2 methods (ClusterSS and ClusterEPs, considered next best as per the F1 score, a metric which gives a better notion of the performance of an algorithm than just the recall or precision measure taken alone). When comparing with the 2 algorithms where Super.Complex has a lower recall, it makes up for this by significantly outperforming the precision measure (55% higher on average and by median) to achieve higher F-1 scores (12% higher on average and 14% higher by median). Also, as we have noted earlier, for this application of detecting protein complexes, validation of results usually involves time-taking and expensive biological experiments, therefore, an algorithm like Super.Complex yielding a low number of false positives (translating to high precision) is more desirable

(even with lower recall) than an algorithm that is able to identify many existing communities but with high false-positive rates (translating to higher recall but low precision). From **Table 3.3**, similar to observations of metrics from the experiments on hu.MAP in **Table 3.2** and (**Supplementary) Table 3.4**, we obtain high precision values with Super.Complex, suggesting that many of the learned protein complexes are of high quality. On performance, we discuss the time complexity of Super.Complex in the **Supplementary Methods** (Time complexity). The whole pipeline was completed in an order of minutes with Super.Complex (including the AutoML step executed on a single Skylake compute node, along with parameter-sweeps for the candidate community sampling step executed on 4 Skylake compute nodes - each with 48 cores @2GHz clock rate). We attempted to run other algorithms on hu.MAP as well, but were unsuccessful due to unavailability of code or limited scalability, as detailed in **Supplementary Results** (SOTA availability).

Table 3.3. Comparing Super.Complex with 6 supervised and 4 unsupervised methods.

Method	**Train**	**Test**	**Precision**	**Recall**	**F-measure**
Super.Complex	TAP	MIPS	**0.841**	0.629	**0.72**
ClusterSS	TAP	MIPS	0.526	0.807	0.636
ClusterEPs	TAP	MIPS	0.606	0.664	0.633
RM	TAP	MIPS	0.489	0.525	0.506
SCI-BN	TAP	MIPS	0.219	0.537	0.312
SCI-SVM	TAP	MIPS	0.176	0.379	0.240
ClusterONE		MIPS	0.428	0.435	0.431
COACH		MIPS	0.364	0.495	0.419
CMC		MIPS	0.46	0.38	0.416
MCODE		MIPS	0.4	0.1	0.16
Super.Complex	MIPS	TAP	**0.718**	0.581	**0.642**
ClusterSS	MIPS	TAP	0.477	0.864	0.614
ClusterEPs	MIPS	TAP	0.424	0.782	0.548
RM	MIPS	TAP	0.424	0.433	0.429
SCI-BN	MIPS	TAP	0.312	0.489	0.381
SCI-SVM	MIPS	TAP	0.247	0.377	0.298
ClusterONE		TAP	0.480	0.46	0.47
COACH		TAP	0.387	0.533	0.449
CMC		TAP	0.447	0.353	0.395
MCODE		TAP	0.422	0.127	0.195
Super.Complex	MIPS	MIPS	**0.552**	**0.733**	**0.63**
NN	MIPS	MIPS	0.333	0.491	0.397

Precision, recall, and F-measures are from Qi et al (**Equations 2.5-2.7**). Parameters for each of the Super.Complex experiments are given in **Table 3.1**.

3.4.4. Learned human protein complexes from Super.Complex, and applications to characterizing unknown proteins and COVID-19

We provide interactive lists and visualizations of the 1028 learned human protein complexes by Super.Complex, along with refined and original CORUM complexes as a resource on https://sites.google.com/view/supercomplex/super-complex-v3-0. The high precision values obtained by Super.Complex in **Table 3.2** suggest that many of the learned complexes are of high quality since the ones with proteins from known complexes match individual known complexes closely. We provide individual community fitness function scores for each of the learned complexes and rank the list of learned complexes by this score to help identify good candidates for investigation for various applications. In this section, we analyze learned human protein complexes by Super.Complex, aiming to provide easily accessible resources for two biological applications that can be investigated further by researchers in the future. We highlight learned complexes with uncharacterized proteins to provide experimental candidates for functional characterization. In the second application, we construct an interactive map of protein interactions between SARS-CoV-2 (the virus causing COVID-19) and 234 learned human protein complexes from Super.Complex using protein-interaction information between SARS-CoV-2 proteins and human proteins [32]. We also provide a list of complexes interacting with SARS-CoV-2 proteins ranked by their possible importance, which can be used to determine potential COVID-19 drug targets (**Figure 3.5** & **Supplementary Results** (SARS-CoV-2 affected protein complexes)).

3.4.4.1. Uncharacterized proteins and their complexes

111 uncharacterized proteins (Uniprot [33] annotation score unknown or less than 3) and their corresponding learned 103 complexes are presented on the website (https://meghanapalukuri.github.io/Complexes/* where * is Protein2complex_annotated.html and Complex2proteins_annotated.html). Three examples of uncharacterized proteins (C11orf42, C18orf21, and C16orf91) along with their corresponding complexes are highlighted in **Figure 3.4**. C11orf42 could potentially be related to trafficking, as it is a part of a complex with 30% similarity to the retromer complex, (*i.e.*, with 0.3 Jaccard similarity to the known CORUM retromer complex), with additional evidence available from the Human Protein Atlas (HPA) [34] (available from http://www.proteinatlas.org) showing subcellular localization to vesicles, similar to other proteins of the complex. C18orf21 also has evidence from HPA, localized to the nucleoli and interacting with other proteins of a complex with 50% similarity to the Rnase/Mrp complex with most members in the nucleoli/nucleoplasm. Further evidence [35] also independently supports C18orf21 as a cellular component of the ribonuclease MRP complex and a participant in ribonuclease P RNA binding as it exhibits significant co-essentiality across cancer cell lines with the POP4, POP5, POP7, RPP30, RPP38, and RPP40 proteins. C16orf91 could potentially be localized to mitochondria like other proteins of the COX 20-C16orf91-UQCC1 complex, with independent experimental evidence available [36].

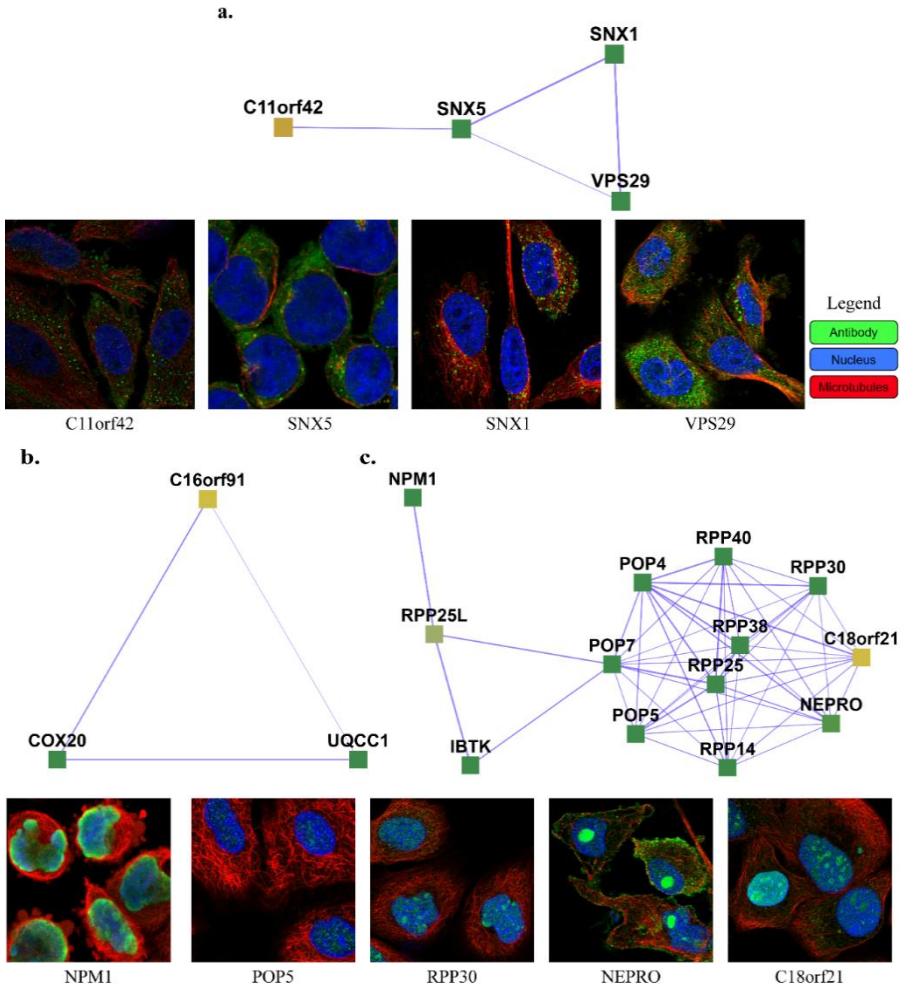


Figure 3.4. Examples of complexes with proteins having low annotation scores.

a. C11orf42 constitutes the Retromer complex (SNX1, SNX2, VPS35, VPS29, VPS26A), potentially related to trafficking, with C11orf42 localized in cells to vesicles, similar to the other proteins of the complex (SNX1, SNX5, and VPS29) **b.** C16orf91 constitutes the COX 20-C16orf91-UQCC1 complex, potentially localized to mitochondria like COX20. **c.** C18orf21 constitutes the Rnase/Mrp complex, with C18orf21, localized to nucleoli, closely interacting with nucleoplasm proteins of the complex such as RPP25, POP5, RPP14, NEPRO, RPP30, IBTK, RPP25L, and NPM1. The images of subcellular localization are available from v20.1 of proteinatlas.org, as https://v20.proteinatlas.org/ENSG00000*/cell, where * is 180878-C11orf42, 028528-SNX1, 089006-SNX5, 111237-VPS29, 167272-POP5, 163608-NEPRO, 148688-RPP30, and 181163-NPM1. Note that localizations were measured in varying cell types, including HeLa, HEL, U2OS, and U-251 MG cells, across the highlighted proteins.

3.5. ACKNOWLEDGMENTS

The authors gratefully acknowledge Benjamin Liebeskind for a TPOT code wrapper, Kevin Drew for computing some previous evaluation measures, and Claire McWhite for critical reading of the manuscript.

3.6. SUPPLEMENTARY INFORMATION

3.6.1. Supplementary tables

Table 3.4. Comparing Super.Complex with 2-stage clustering on the hu.MAP dataset shows comparable performance for both algorithms.

6 existing evaluation measures including SPA (**Equation 2.10**), Qi et al (**Equation 2.7**), K-clique measures [22], MMR [9] and precision-recall product [37], 3 new evaluation measures – FMM, CMF, and UnSPA (**Figure 2.1**) are used for comparison.

Metric			Super. Complex	hu.MAP (ClusterONE + MCL)
No. of learned protein complexes			1028	4659
No. of learned protein complexes after removing non-gold standard proteins			131	274
New metrics	FMM	Precision	**0.767**	0.471
		Recall	0.534	**0.686**
		F1 score	**0.63**	0.559
	CMF	Precision	**0.839**	0.804
		Recall	0.733	**0.789**
		F1 score	0.783	**0.797**
	UnSPA	PPV_u	0.903	**0.93**
		Sn_u	0.874	**0.893**
		UnSPA	0.888	**0.911**
Existing metrics	SPA	PPV	0.885	**0.907**
		Sn	0.675	**0.681**
		SPA	0.773	**0.786**
	Qi et al	Precision	**0.832**	0.763
		Recall	0.665	**0.766**
		F1 score	0.739	**0.764**
	K-clique	Precision	**0.992**	0.825
		Recall	0.674	**0.735**
		F1 score	**0.785**	0.772
	Weighted K-clique	Precision	**0.998**	0.973
		Recall	0.948	**0.962**
		F1 score	**0.972**	0.968
	MMR	Precision	**0.714**	0.616
		Recall	0.664	**0.747**
		F1 score	**0.688**	0.675
	PR	Precision	**0.738**	0.681
		Recall	0.692	**0.759**

		PR product	0.714	**0.718**

In summary, Super.Complex achieves better performance on 7 precision-like metrics (18% higher on average, 9% by median with 2 metrics doing better with 2-stage clustering by 3% on average and by median), while the 2-stage clustering method achieves better performance on recall measure on all 9 metrics by 10% on average and 9% by the median. Super.Complex achieves better F1-score-like measures on 4 of the metrics by 4% on average and 2% by the median, compared to 5 wins by 2-stage clustering by 2% on average and by the median.

3.6.2. Supplementary results

3.6.2.1. Algorithm guarantees

The train and test communities are guaranteed to be independent by the proposed splitting algorithm. Due to the seeding and growth heuristics we employed, the learned communities are guaranteed to be internally connected at every step of the iteration. With an overlap threshold *t*, the merging algorithm guarantees that no 2 learned communities have an overlap greater than or equal to *t*. If an overlap threshold of 0 is provided, the merging algorithm yields disjoint, non-overlapping learned communities which guarantee separation, *i.e.*, no pair of learned communities can be merged to yield a community of higher community score. The number of maximum iterations provided guarantees that the seeding and growth converge to a solution.

3.6.2.2. Robustness of the Super.Complex algorithm

In the candidate community search, we can expect that the learned complexes will differ from experiment to experiment due to 3 modes of possible stochasticity namely, (i) the ϵ-greedy heuristic, (ii) the pseudo-metropolis or ISA heuristic, and (iii) the post-processing algorithm merging pairs of highly overlapping communities. A re-run of the search with the same parameters as the best results on hu.MAP yielded 1025 complexes, compared to 1028 in the original experiment run. Of these, 999 were identical. We compare

the sets of learned complexes from both runs and observe high correlation, as observed from an FMM F1 score of 0.984, CMF F1 score of 0.988, unbiased accuracy of 0.989, and a Qi et al F1 score of 0.985. This indicates that despite the stochasticity, we can expect fairly stable and robust results.

3.6.2.3. Performance

The current problem of learning the best community fitness function and growing multiple seeds on a network into candidate communities poses an interesting design challenge. Multiple architectures are valid and can be the best based on the constraints of the computing systems. Our design performs well and is capable of giving results in an order of minutes for a network with a similar scale (~10k nodes, ~100k edges) as hu.MAP. With default parameters, a single run of the pipeline on hu.MAP gave reasonably good results in around 30 minutes, on 4 Skylake nodes (each with 48 cores @2GHz clock rate). In more detail, it averages ~10 min per parameter set of the candidate community sampling part of the pipeline using 4 Skylake nodes. For the best results, we ran multiple parameter sweeps for a few hours (a total of around 100 sets), achieving only a minor improvement in results. The AutoML model can yield good models in an order of minutes as well, however, we ran it for a few hours on a single node to obtain the best model. For accessibility to run on non-HPC (high-performance computing) systems as well, using the optimizations provided and setting low thresholds, it is possible to run the pipeline serially in a matter of hours, even on a personal laptop. A discussion on time complexity is included in Supplementary Methods (Time Complexity).

3.6.2.4. State of the art software availability

We performed experiments with the implementation of the state-of-the-art method SCI-SVM on hu.MAP whose Matlab code was available and which we were able to run on a single core of a Skylake node (2GHz clock rate). The implementation with the default parameters on a toy dataset and a human dataset resulted in the identification of a large number of small complexes and a very few big complexes, despite sufficient training with big complexes. As big complexes were only a few, the results have low precision and recall. Computed Qi et al precision, recall, and F1 scores for even a low threshold *t* of 0.2 are 0.016, 0.0075, and 0.0103 respectively for hu.MAP. A single run of SCI-SVM took around 3 hours to complete and yield these results, unlike better results yielded by Super.Complex in around 30min. Additional runs tuning different parameters in SCI-SVM did not improve results.

In our experiments, we were unable to get the UI (User Interface) for the ClusterEPs software to complete execution on a personal laptop for hu.MAP. Note that Super.Complex on the other hand finishes running on the same personal laptop and provides results after a few hours. Code was not available for ClusterEPs, or the other supervised methods in **Table 3.3** – RM, NN, and ClusterSS.

3.6.2.5. SARS-CoV-2 affected protein complexes

SARS-CoV-2 is the novel coronavirus that resulted in the pandemic which started at the end of 2019. The virus infects human cells through a spike protein and enters the cell, and subsequently, the proteins of the virus have the potential to interact with multiple human proteins. We preliminarily investigated the human protein complexes that may be affected by the virus, and thus contribute, directly or indirectly to COVID-19, the disease caused by SARS-CoV-2. This study was made possible by timely work identifying 332

human proteins that physically interact with SARS-CoV-2 proteins, discovered using AP-MS experiments [32].

We consider a protein complex to be linked to COVID-19 if any of the proteins in the human complex interact with a SARS-CoV-2 protein, *i.e.*, if the human protein complex contains one or more of the above-mentioned 332 human proteins. Super.Complex learns 234 complexes linked to COVID-19, all except one new, *i.e.*, not perfectly matching a known CORUM complex. While CORUM contains 430 complexes linked to COVID-19 ('original CORUM'), we obtain 214 complexes after cleaning ('refined CORUM'), *i.e.*, removing complexes with sizes lesser than or equal to 2 and using the merging algorithm described in the Materials and Methods section to ensure no more than 50% Jaccard overlap between any two SARS-CoV-2 complexes. All SARS-COV2 linked protein complexes from Super.Complex, scored based on the number of proteins in a complex interacting with a SARS-CoV-2 protein, along with lists of refined CORUM and original CORUM complexes linked to SARS-COV2 are available on the website (https://meghanapalukuri.github.io/Complexes/* where * is Complex2proteins_covid.html, CORUM_Complex2proteins.html, and originalCORUM_Complex2proteins_covid.html respectively). **Figure 3.5** shows examples of human protein complexes likely especially relevant to the SARS-CoV-2 life cycle, as they involve multiple interactions between SARS-CoV-2 proteins and members of the same (or a functionally related) complex. Visualizing the SARS-CoV-2 interacting proteins in terms of their native assemblies serves to emphasize the preferential interaction of SARS-CoV-2 proteins with certain cellular systems, as for nsp4 and orf6 both interacting with the nuclear pore complex, and may help highlight important aspects of the SARS-CoV-2 life cycle for future study.

3.6.3. Supplementary methods

3.6.3.1. Topological features

These features include 5 subgraph specific features - the no. of nodes in the subgraph, the 1st 3 singular values of the adjacency matrix of the subgraph, and the weighted intra-cluster density, obtained by using the weighted sum of edges instead of the number of edges in the numerator of the intra-cluster density. The remaining 13 features are statistical measures of the combinations of features defined for individual nodes of the subgraph. We use the median, mean, variance, and maximum value of the degree $d_v$of a vertex *v*, which is the sum of the weights of the edges connecting the node to its immediate neighbors. Next, we use the mean, variance, and maximum value of the clustering coefficient of a vertex, c_v, which gives the ratio of the number of triangles it is a part of, considering only its first nearest neighbors and the number of all such possible triangles, *i.e.*,

$$c_v = \frac{No.\ of\ edges\ connecting\ pairs\ of\ neighbors\ of\ v}{No.\ of\ all\ possible\ edges\ connecting\ pairs\ of\ neighbors\ of\ v}$$

$$= \frac{|\{(v', v'') \mid v', v'' \in N(v), v' \in N(v'')\}|/2}{|N(v)|^{*}(|N(v)| - 1)/2}, \tag{3.5}$$

where, *N(v)* is the set of neighbors of vertex v. We use the mean, variance, and maximum value of the degree correlation of a node *v*, which is the average degree of all neighbors of node *v*, *i.e.*,

$$DC_v = \frac{\sum_{v' \in N(v)} d_{v'}}{|N(v)|} \tag{3.6}$$

The last 3 features are the mean, variance, and maximum value of the weights of the edges of the subgraph.

3.6.3.2. Evaluation with existing measures

To compare the set of learned communities with known communities, we first construct a set of reduced communities containing nodes only present in known communities. We retain communities with 3 or more nodes only. We use a plethora of evaluation measures, including the proposed measures detailed in the results section, along with existing measures such as Qi et al precision, recall and F1 score, sensitivity. PPV, their associated accuracy, and MMR. We plot PR curves for edges learned. The learned communities are evaluated against all known communities, only training communities, and only testing communities. We report the results on the test communities for the yeast experiment in **Table 3.3**.

3.6.3.3. Time complexity

The average time complexity of the algorithm is determined by the two main steps - training the AutoML algorithm TPOT and the candidate community search step. The time complexity of TPOT is not explicitly mentioned in their papers [38]–[40], but from the method description, in our use, we can make a simple estimate. It would be at least $O\left(\frac{g\times p\times m\times s\times f}{c}\right)$, where g is the number of generations, p is the population size, m is the number of machine learning models and feature preprocessor types tried, s is the number of samples, f is the number of features used, and c is the number of processes in the single compute node running the AutoML step.

The candidate community search step has a complexity of $O\left(\frac{X\times G^4\times K\times S}{P}\right)$, keeping in mind that in the worst case, the feature extraction step has a complexity of $O(n^3)$ for a subgraph with n nodes. P is the total number of processes used to run this code in parallel, S is the number of seeds used to sample candidate communities which is typically N, the number of nodes in the network, G is the number of steps for each community's growth

and is also taken to be the final number of nodes in the subgraph after growth for complexity calculation (as both are similar), *K* is the average degree of the graph. *X* is the machine learning model inference time for one value, which is assumed to be a small constant here in cases with fairly simple models. We note that in an alternate method that we provide, one can specify the number of neighbors to check, say *M* instead of checking all neighbors, achieving gains when $M < K$, with the complexity now as $O\left(\frac{X \times G^4 \times M \times S}{P}\right)$. Using all nodes as seeds ($S = N$), in very large sparse protein interaction networks (small degree $K \ll N$) with small complexes (for example, we use $G < 10$ nodes in our yeast experiments) and considering that the machine learning model inference time is constant, the complexity can be considered $\sim O\left(\frac{N}{P}\right)$. In problems where communities are large making the polynomial greater than the number of nodes (*i.e.*, $G^4 > N$) time complexity will be determined instead by the community size *G*. For comparison, the authors note that SCI-SVM and SCI-BN have a complexity (in our notation) of $O(G^5 \times K \times S)$, while other supervised methods do not mention their time complexities. We note that all the other supervised algorithms in the best-case scenario, assuming efficient implementation of the growth strategy for a single node would still have a complexity of $O(N)$ due to the serial nature of the algorithms, compared to our best-case scenario of $O\left(\frac{N}{P}\right)$, where one can achieve gains by increasing the number of parallel processes *P*.

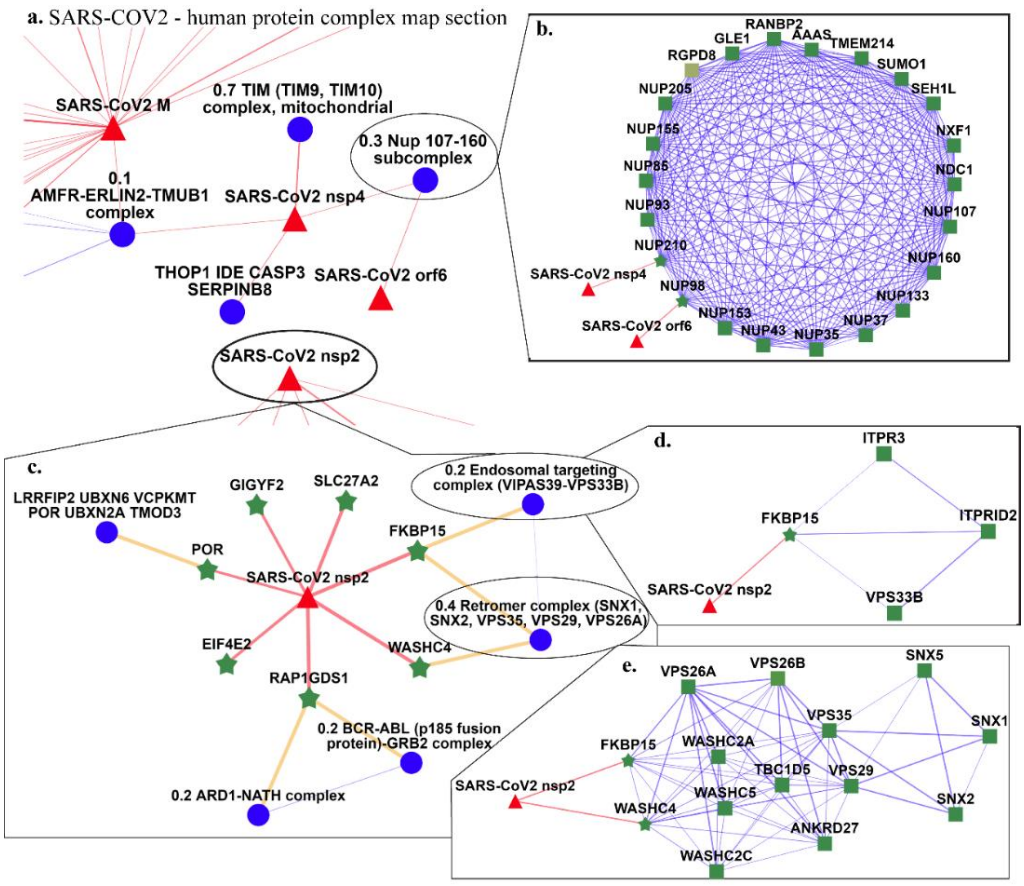


Figure 3.5. SARS-CoV-2 - human protein complex map showing complexes identified by Super.Complex.

a. A section of the full map, featuring SARS-CoV-2 nsp4 and orf6 and their interacting human protein complexes **b.** A protein complex with a 30% match to the Nup 107-160 subcomplex interacts with both SARS-CoV-2 nsp4 and orf6 **c.** Map of SARS-CoV-2 nsp2 interactions with human proteins and their corresponding complexes **d.** A complex with a 20% match to the Endosomal targeting complex, and **e.** A complex with a 40% match to the retromer complex, both of which interact with SARS-CoV-2 nsp2. An interactive map is available at https://meghanapalukuri.github.io/Complexes/SARS_COV2_Map_only_mapped_complexes_names.html.

3.7. SUMMARY OF IDEOLOGY & CONTRIBUTIONS

The steps of the pipeline are fairly independent and can be improved on their own with methods to test the accuracy/performance of each of the steps. Given that accuracy is more important to us than performance (time), our pipeline allows exploring several user-specified alternatives in each of the steps to find what works best for the given application. Future work can include providing default options of models and parameters that work best for the application so that these can be transferred to a different problem when performance is a constraint or when there is limited data in the other domain.

3.7.1.1. AutoML

Different types of machine learning models have been used in the supervised methods to learn community fitness functions as the function distinguishing communities and random walks. We first experiment with linear SVMs and 3-layer neural networks and cross-validate on their hyper-parameters to achieve reasonable performance. To achieve the best performance, we wished to try different machine learning models, and to accelerate this process, turned to AutoML methods. We incorporated *tpot* into our pipeline using a wrapper to list the different machine learning algorithms we were interested in exploring and configured it to run in a distributed mode, helping the AutoML method run faster. The method employs a genetic algorithm to explore different ML algorithms and hyper-parameters to return the ones that yield the best performance.

3.7.1.2. Distributed algorithms

While several heuristics have been proposed in the supervised community detection methods, all of them are serial and are thus not scalable to large networks. Oftentimes, the networks on which community detection algorithms are to be implemented are huge. For instance, the 2010 Twitter network has 1 billion edges and protein interaction networks in

different organisms have millions of edges. We design a distributed algorithm for identifying candidate communities that can run on multiple processors as well as multiple computers using an intermediary distributed graph storage format. We make the trade-off of some additional storage to allow embarrassingly parallel detection of possibly similar complexes and merge them at the end serially with an iterative merging algorithm.

3.7.1.3. Improving and providing multiple candidate community selection heuristics

The algorithm heuristic for growing nodes into communities combines greedy search with options of iterative simulated annealing or pseudo-metropolis criteria with the additional option of starting with maximal cliques as the seeds instead of single nodes. In our experiments, we find that single nodes perform as well, if not better than maximal cliques as the latter would bias the candidates to be denser which is not necessarily the case. We also find that the greedy search is more powerful than the following iterative simulated annealing or metropolis criteria. Firstly, we chose these heuristics and not some others used by the supervised community detection algorithms as they would theoretically lead to biased results. We believe evaluating all the neighboring nodes would be the least biased, which would allow the community fitness function to choose which would be the best node to add. Some of the heuristics used by other supervised algorithms such as choosing the maximum degree neighbor or choosing a neighbor which is connected to most of the existing nodes can induce bias by picking only candidates that have a star-like structure or are dense respectively. For performance, however, we also provide a version that picks the neighbor with the maximum edge weight connecting to the current subgraph. The idea behind using edge weights to decide the node to add is that, at least in the human PPIN we use, the edges represent probabilities of belonging to the same complex and

higher edge weights can lead to better candidate complexes. But even in other networks, it can be a good proxy ensuring strong interactions in the sampled complexes.

3.8. SOFTWARE DETAILS

We have bash files that perform all the steps of the pipeline sequentially. Each of the steps has a separate main file in the event one would like to perform only a single step of the pipeline and look at its results before moving on to the next step. All the configurations required including specifying data paths, parameter values, and heuristics to use can be changed in the single default configuration file provided for the whole pipeline, which can also be used to change configurations and run any particular step of the pipeline. Alternately, we allow command-line inputs to change the configurations for a step for those who prefer to use the command line. We use version python3.x and have built it initially using python2.x, therefore it should be back-compatible with previous versions as well. An integration test of the pipeline is provided with a small toy dataset that should achieve perfect results, along with passing the partial unit tests provided. We use the library pytest for testing and joblib for parallel processing. In the past versions, we have used full object-oriented programming with shared objects as the code was serial, but have opted to move to functional programming for the current parallel version that also supports the serial version. Apart from tpot [24], other python libraries used include NetworkX [41], scikit-learn [42], and TensorFlow [43] for the neural network. The full list can be found in the requirements of the latest version of the code which is available at https://github.com/marcottelab/super.complex.

3.8.1. Software design

The current design reduces RAM and eliminates communication between processes of the same node (other than communication initiating a task and indicating the end of a task by the process), opting for slightly more I/O and graph file transfers between compute nodes only at the start and end of the entire sampling process. This works best for systems that have low RAM that cannot hold the entire graph in memory and have high latency for communication overhead between processes and between compute nodes. We do note that the one-time copying of a graph onto each compute node is a time-consuming process, although each graph node is stored using joblib for fast I/O and transfers. Another limitation is that with the current design, our scalability extends only up to graphs that can be completely stored on a single compute node's disk memory. Note that this still is a massive graph, for instance, hu.MAP 2.0 with ~17million edges with just the node names and edge weights is around ~0.5 GB and a Stampede2 Skylake compute node has around ~144GB available as disk memory, allowing scalability for networks with billions of edges. To enable even exascale scalability, we propose an alternate architecture that relies a bit on reasonably low latency between compute nodes. Essentially, the approach can be summarized as a graph partitioned and stored over multiple compute nodes after bookkeeping which graph nodes are present on which compute node, and then each compute node grows the seeds stored on it and uses MPI between compute nodes to retrieve information about graph nodes stored on other compute nodes. Graph partitioning algorithms can be explored to partition the graph at a high level for this purpose. We are interested in exploring existing distributed graph libraries and databases as well such as Katana Graph.

Coming back to our current approach and even for the proposed architecture above, if the code becomes I/O bound on some systems (which has not happened on our cluster),

we would like to include a version of the algorithm which incorporates more efficient I/O practices. For this, rather than all the processes on a node doing independent read/writes, we will have one or a fewer number of processes that do the read/writes for all the processes and then use MPI (message-passing interface) for communication between the different processes. On each compute node, rather than multi-processing with python as we have done in the current implementation, we are also interested in using multithreading with OpenMP and writing the code in C++ to build a more efficient implementation. Especially in the new architecture proposed, since the graph is partitioned across multiple compute nodes, there may be a speedup associated with using in-memory storage if the number of graph nodes per compute node is small enough. Here, using the shared memory setting in OpenMP may be especially beneficial. In the current code, some of the serial parts store the graph in-memory, which needs to be converted to the distributed storage setting.

3.9. PREVIOUS VERSIONS OF SUPER.COMPLEX

3.9.1. Software design

Previous versions of Super.Complex employed a serial architecture where the entire graph was stored in memory using the NetworkX package. An initial multiprocessing parallel architecture we experimented with also had in-memory graph storage, however, this turned out to make the problem (most likely) in-memory bound (as we obtained several memory errors) which were resolved by using the current approach where we use disk memory instead and do on-demand I/O. In the interest of time, we did not perform extensive memory debugging to find the root cause, as we quickly came up with the current distributed architecture which further scales with the size of the network as it is practically possible to resort to disk memory for storing large graphs and decided to move away from in-memory. Nevertheless, it is interesting why the serial in-memory approach is CPU-

bound when compared to the parallel in-memory approach which became in-memory bound. In the latter case, we speculate that this could be because the graph was copied in memory for each process overloading the caches and/or RAM (since we operate in a multiprocessing environment, where every process has separate memory associated with it) and in the previous version of the algorithm, to optimize performance, we remove edges from the local graph copy that are present in the growing subgraph. Even if we were to remove this optimization and since copy on write happens as is the case with Linux allowing us to share the graph for read-only purposes by the individual processes or if we employ other shared memory techniques, multiple processes trying to access the graph object at the same time can lead to delays. This again may be improved by using a distributed graph storage format akin to what our current approach uses, stored compactly in-memory with a bitwise hash map. In-memory approaches like these can be explored and are viable for graphs that are not very large.

3.9.2. Super.Complex v2.0

Comparing and contrasting this with Super.Complex v3.0, here, for heuristics, we had a less efficient algorithm that is slightly different from the current one but performed some duplicate operations. Growing the subgraph in an iteration was a 2-stage process. In the 1st stage, we applied the greedy heuristic on the neighbors of each node of the current subgraph to select the candidate w.r.t that node. This is done after reducing the number of neighbors evaluated, first, by selecting a random subset, if the number of neighbors exceeded a large threshold *t3* and then, by selecting the top *m%* percentage of the neighbors based on descending order of edge weight if the number of neighbors exceeds a threshold *t4*. Note how in the current algorithm, *t1*, and *t2* are thresholds pertaining to the neighbors of a subgraph, whereas *t3* and *t4* are thresholds pertaining to the neighbors of a node of a

subgraph. In the 2nd stage, we applied the -greedy heuristic to select, out of the set of candidates (neighbors corresponding to each of the nodes in the subgraph obtained in the 1st stage), the neighbor to add to the subgraph in that iteration. While in the exhaustive case, this is redundant and equivalent to the current algorithm, the thresholds on the neighbors evaluated sequentially in the 2 stages lead to different results. We also had the additional heuristics in the current approach, including the pseudo-metropolis and iterative simulated annealing options built on top of the base heuristic. In the current algorithm (Super.Complex v3.0), by removing the duplicate neighbor evaluations, we gain a significant speed-up even allowing exhaustive neighbor search on hu.MAP and better results. In the post-processing part of Super.Complex 2.0, we also experimented with removing only the smaller complex that overlaps, which we now removed, as we can bias the algorithm to form bigger complexes by this. Super.Complex v2.0 including results are available on the website:

https://sites.google.com/view/supercomplex/results

3.9.3. Super.Complex v1.0

Super.Complex v1.0 used a linear SVM with C = 1 for learning the community fitness function, a discriminatory function between known communities and random walks. For growing candidate communities from a seed, a simple greedy edge weight heuristic was used. Here, the highest edge weight connected to the subgraph is added and the new subgraph is evaluated if it is still classified as a community by the SVM model. The process continues until the model determines that the subgraph is no longer a community, at which point, the last node and edges added are removed to form a candidate protein complex. These options are also made available in Super.Complex v3.0 code. A toy dataset with 100 nodes, comprising disjoint 10-cliques was constructed using MATLAB. hu.MAP 1.0 with

an edge weight threshold of 0.3 was applied to obtain a refined network with 64,047 edges and 7778 nodes. Building on this, 6 more datasets have been integrated into hu.MAP 1.0 to construct the next version hu.MAP 2.0 comprising 17 million edges. Additionally, using an older version of CORUM [44], we had around 1000 experimentally characterized benchmark protein complexes. Note that in Super.Complex 1.0, like in Super.Complex v2.0, disconnected and duplicate complexes are considered during training. In the test network, the search process yielded all the complexes in the network perfectly, *i.e.*, with all nodes and edges when we use 1 feature or 6 features and all 30 nodes as seeds for growth. Here, we train on 5 positives and 6 negatives and test on 5 positives and 5 negative complexes.

Table 3.5. Accuracies of ML models learning community fitness functions in Super.Complex v1.0 on hu.MAP 1.0 and hu.MAP 2.0.

Network:	hu.MAP	hu.MAP	hu.MAP	hu.MAP 2.0
No. of features	1	6	6	6
No. of train positives	503	503	503	572
No. of train negatives	476	950	1261	475
No. of test positives	522	522	522	584
No. of test negatives	504	987	1331	500
Accuracy for test positives	0.793	0.946	0.839	0.995
Accuracy for test negatives	0.522	0.986	0.986	0.998
Test precision	0.632	0.973	0.959	0.998
Test recall	0.793	0.946	0.839	0.995
Test F1 score	0.703	0.959	0.895	0.997

Table 3.6. Search results of Super.Complex v1.0 on hu.MAP.

No. of features:	1	6	6
Test F1 score	0.703	0.959	0.895
No. of seeds for complex search	1500	1500	7778
No. of predicted complexes	1320	1256	5620
Prediction precision	0.013	0.038	0.031
Prediction recall	0.018	0.043	0.091
Prediction F1 score	0.015	0.04	0.046

Note that the number of known complexes is 1025, against which predicted complexes are compared.

3.9.4. Subgraph feature selection for learning the community fitness function in Super.Complex

3.9.4.1. Features based on graph theory

Of the existing work, only ClusterSS performs feature selection to select the most important features and ClusterEPs indirectly does feature selection by constructing discriminatory feature sets. As we intend to generalize to different problems, it is fruitful to have a streamlined pipeline for feature selection so that the classification problem at hand achieves the best performance. As proof of concept, we pool together many of the discussed features and manually perform feature selection using PCA and a logistic regression model as detailed in the appendix, to demonstrate their importance and also to provide information on which features are most useful. Following this, we recognize that an automatic feature selection pipeline would be more efficient, and therefore use the AutoML pipeline *tpot* to learn the feature normalization and feature selection that works best for our application.

This pipeline for feature selection and dimensionality reduction will become more important in the future as more features are added. As proof of concept, we pose and answer the question of which topological features are not useful in differentiating protein complexes from random walks in a network. For this, we analyze a dataset of 18 topological features labeled as known protein complexes and random walks. Train and test data comprise 217 and 756 samples (rows) of protein subgraphs respectively. The dataset has the features, *nodes* - no. of nodes (proteins) in subgraph, *dens* - density of the subgraph; *sv1, sv2, sv3* - 1st 3 singular values of the adjacency matrix of the subgraph; *edge_wt_var*, *edge_wt_max, edge_wt_mean* - edge weight statistics (variance, maximum and mean); *CC_var*, *CC_mean*, *CC_max* - clustering coefficient statistics (variance, mean and maximum), *degree_median*, *degree_mean*, *degree_var*, *degree_max* - degree statistics

(median, mean, variance and maximum) and *DC_var*, *DC_max*, *DC_mean* - degree correlation statistics (variance, maximum and mean). Features with overall less variance are considered unimportant and are found using a PCA analysis. This is further validated using a stepwise feature selection model employing logistic regression for classification.

The procedure was performed in R. We first apply the standard scaler to the data, and perform PCA on the training dataset. On computing the percentage variation explained by PCs, we see that the first 11 PCs or more contribute to more than 99% variance of the data and the first 13 PCs contribute to more than 99.9% variance. We then perform dimension reduction with PCA from the original 18 features to the 11 and 13 PCs and observe their performance in classification using logistic regression and compare with the original dataset. We also use the method *glmstep* which uses logistic regression itself to perform stepwise feature selection, yielding the best feature set to be one with 9 original features - *dens*, *degree_mean*, *CC_max*, *CC_mean*, *edge_wt_mean*, *DC_mean*, *DC_var*, *sv2*, and *sv3*. From the density plot in **Figure 3.6**, we can see that the logistic regression model using reduced data with 13 PCs performs nicely, well separating the two classes. On plotting ROC curves for the reduced and full models, we can see that using 13 PCs gives us an AUC very similar to a model using all features, while there is quite some loss using only 11 PCs. Interestingly, the logistic regression model with only 9 features outperforms the original model with all features, showing us that feature selection is an important step.

We next perform feature selection using PCA as follows. The PCA rotation matrix gives the components of features along the principal components. We calculate the percentage of each feature along the top principal components (PCs) of interest (*perc_norm_imp_PCs*) as,

$$perc\_norm\_imp\_PCs\ for\ feature\ \hat{f} = \frac{\sqrt{\sum_{i=1}^{n} f_{PC_i}^2}}{\sqrt{\sum_{i=1}^{n_{total}} f_{PC_i}^2}} . \quad (3.7)$$

Here, we choose the top *n* PCs of interest out of n_{total} PCs. f_{PC_i} , where , $i \in \{1, 2, .. n_{total}\}$ is the component of a feature $\hat{f}$ along a principal component, *i.e.*,

$$\hat{f} = \sum_{i=1}^{n_{total}} f_{PC_i} \widehat{PC_i} . \quad (3.8)$$

If *perc_norm_imp_PCs* is high it means the feature contributes more to important PCs, and hence, to more of the variance of the data.

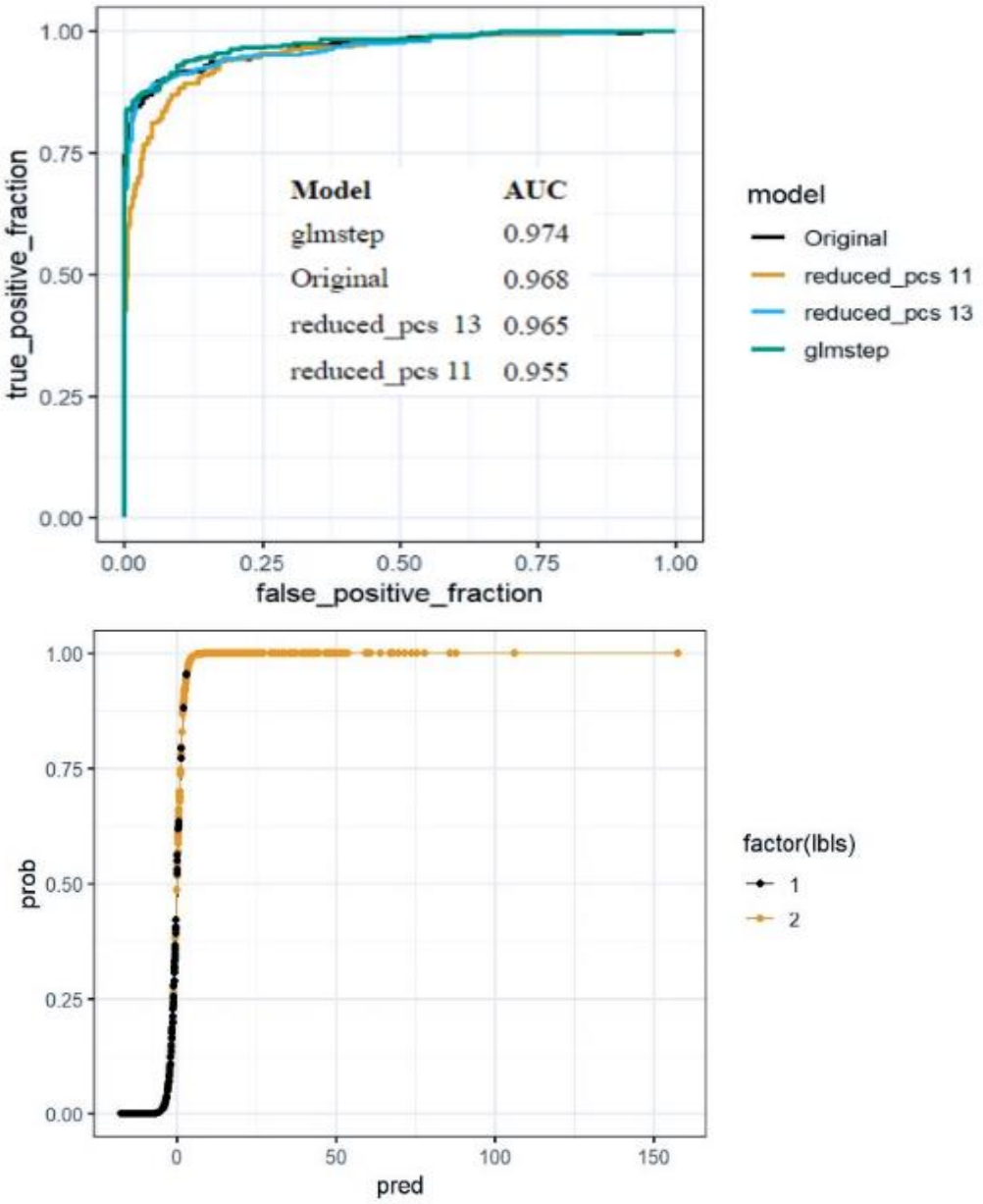


Figure 3.6. Feature selection for the model learning the community fitness function.

Top - Logistic regression density plots for different models with different features

Bottom - ROC curve for the best model

Table 3.7. The percentage of features along important principal components

Feature	***perc_norm_imp_PCs***
edge_wt_var	0.9989
CC_var	0.9987
CC_mean	0.9969
nodes	0.9961
sv2	0.9955
edge_wt_max	0.9950
CC_max	0.9934
sv3	0.9902
dens	0.9827
edge_wt_mean	0.9730
degree_var	0.7806
degree_max	0.7121
DC_var	0.7078
sv1	0.7053
DC_max	0.5665
degree_median	0.5122
degree_mean	0.4981
DC_mean	0.4601

The 4 features - DC_max, degree_median, degree_mean, and DC_mean correspond to the least important features in the PCA feature selection analysis above. Of these 4, the 2 features - DC_max and degree_median are unimportant as they do not appear in the logistic regression feature selection model and hence can be removed from the data without affecting the classification results much.

3.9.4.2. Supervised feature extraction for learning the community fitness function

We also experimented with extracting features specific for the task of community detection. The goal was to build representations that are specifically discriminatory between communities and random walks using the successful graph convolutional networks [45]. We attempted 7 architectures of GCNs (**Table 3.8**, presenting the best results for each architecture, tuning the number of epochs in each case) using the graph network topology, training to perform graph classification for the task of learning a community fitness function, using the same binary classification task setting as in Super.Complex, and achieved competitive results (0.79 APS compared to Super.Complex's 0.89).

Table 3.8. Classification results with different graph neural network architectures.

Model	Epochs	Training loss	Validation accuracy	Test Accuracy	Test F1 Score	Test APS Score
emb1TwoConvFourClassiLayerGCN	9999	0.49	83.02	85.71	0.87	0.79
emb1TwoConvThreeClassiLayerGCN	19998	0.43	81.13	84.82	0.86	0.78
emb1OneConvTwoClassiLayerGCN	9999	0.62	71.70	80.36	0.81	0.74
emb0OneConvTwoClassiLayerGCN	20008	0.70	56.60	50.00	0.67	0.50
emb0TwoConvThreeClassiLayerGCN	20017	0.69	56.60	50.00	0.67	0.50
emb1TwoConvSAGE	20008	0.69	56.60	50.00	0.67	0.50
emb0TwoConvFourClassiLayerGCN	9	0.69	47.17	50.00	0.67	0.50

Conv indicates graph convolutional layer, classi indicates classification layer. Emb1 or 0 indicates whether an embedding layer was used or not. APS stands for Average Precision Score

Code for this work is available at:

https://github.com/meghanapalukuri/graph_classification

3.10. FUTURE WORK

3.10.1. Improving community embeddings

3.10.1.1. Biological features

For additional biological features, features can be derived directly from the protein sequences to not induce bias into the prediction pipeline and to ensure trustworthiness in our data sources and predictions. With the help of a wrapper written by Claire McWhite, we obtained vector representations for proteins in hu.MAP using ProtBERT [46] followed by dimensionality reduction with PCA.

In particular, we believe including binding affinity information from direct coupling analysis (DCA) holds a lot of promise [47]. We plan to re-weight the network edges by integrating high confidence information from DCA in the form of the number and length of binding sites (sets of contiguous amino acids). We can also incorporate additional sources of data such as yeast two-hybrid experiments [48]. Information such as sub-cellular localization can also be included as node attributes. Also, we would like to explore methods that can learn co-complex membership directly from sequences by incorporating them as node attributes.

3.10.1.2. Topological features

As past work seems to suggest, incorporating more features, especially useful ones helps boost the performance and so, we plan to add additional features. An interesting feature, the topological change captures the changes in the topology of the community as different edge-weight cutoffs are applied to the graph, and therefore can be used to

investigate the effect of low-weight edges. This can perhaps account for the noise in PPIs. Typically, the network edges with low weights are filtered out. Determining the threshold for filtering out edges is an outstanding issue. So rather than filtering out edges, if the algorithms inherently take into account the noise in edge weights, this issue is solved. More examples of topological features are given below. We also plan to experiment with recent graph embeddings such as graph2vec [49] which incorporates multiple rooted subgraphs of the graph in question as informative topological structures to form graph representations. While only a few methods such as this exist that directly build a representation for the graph, many more methods exist for building graph node representations, such as node2vec [50] which builds low-dimensional representations that preserve network neighborhoods. These node embeddings can be pooled together to build a graph embedding.

Unique community-specific features are as follows. Let the number of different cutoffs you want to evaluate be n, and let the different graphs obtained by applying these cutoffs be G_i. Then the *n-1* topological changes are given by,

$$T_i = \frac{|E_i| - |E_{i+1}|}{|E_i|}, \text{where}, G_i = (V, E_i), i \in \{1,2,..n\}. \quad (3.9)$$

The average shortest path length is the average of shortest path lengths between different pairs of vertices in the community. The first singular values/eigenvalues of the adjacency matrix can be used.

Some examples of useful topological features of communities obtained from statistics of node features include the maximum degree, given by,

$$Maximum\ degree = max_{v \in C}(d_v). \quad (3.10)$$

Node wise topological features:

The topological coefficient gives a measure of the number of rectangles the node participates in. Let *M(v)* be the set of nodes that share at least one neighbor with node *v*, *i.e.*, the 2nd order neighbors of *v*. Then, the topological coefficient is,

$$T_v = \frac{1}{|N(v)|}\left(\frac{1}{|M(v)|}\sum_{v' \epsilon M(v)} J(v,v')\right);\ M(v) = \{v' \mid v' \epsilon N(N(v))\}. \quad (3.11)$$

where, *J(v, v')* is the number of shared neighbors of vertices v and v', *i.e.*,

$$J(v,v') = |\{v'' \mid v'' \epsilon N(v), v'' \epsilon N(v')\}| + \{1 \text{ if v } \epsilon \text{N(v'), 0 otherwise}\} \quad (3.12)$$

Let P_{vm} be the set of all subgraphs of the full graph *G* or the specific subgraph of interest *C* such that each member $p \epsilon P_{vm}$ is one of the shortest paths from node *v* to node *m*, *i.e.*, *p* is the set of nodes on this shortest path including both the end nodes. Betweenness centrality of a node v is given by,

$$Cb_v = \sum_{s \neq v \neq t} \frac{No.of\ shortest\ paths\ between\ s\ and\ t\ which\ include\ v}{No.of\ shortest\ paths\ between\ s\ and\ t} = \sum_{s \neq v \neq t} \frac{|P_{svt}|}{|P_{st}|} \quad (3.13)$$

Here, $P_{snt} = \{p \mid p \epsilon P_{st}, n \epsilon p\}$ and *s* and *t* are nodes not present in the community.

The average shortest path length of node *v* is given by,

$$L_v = \frac{\sum_{p \epsilon P_v} |p| - 1}{|P_v|}, \text{where}, P_v = \{p \mid p \epsilon P_{vm}\} \quad (3.14)$$

Closeness centrality of a node is the inverse of the average shortest path length of node *v*, *i.e.*,

$$Cc_v = \frac{1}{L_v}. \quad (3.15)$$

The eccentricity of a node v is the maximum non-infinite length of shortest paths from n, *i.e.*,

$$ecc_v = \max_{p \epsilon P_{vm}, p \neq inf} |p| - 1. \quad (3.16)$$

Let the diameter D_c be the shortest path length between the farthest pair of nodes in a community *c*, *i.e.*,

$$D_c = \max_{p \in P_{nm}\, \forall n,m \in C} |\, p \,| \,. \tag{3.17}$$

Radiality or centrality index of node *v* is given by,

$$r_v = \frac{D_c - L_v + 1}{D_c}. \tag{3.18}$$

Stress centrality of a node *v* is the number of shortest paths passing through the node *v*, *i.e.*,

$$S_v = |\{p \,|p \in P_{nm}\,, v \in p\; \forall n, m \in C \text{ or } G\}| \,. \tag{3.19}$$

3.10.2. Additional experiments

It would also be interesting to investigate different topologies of communities on hu.MAP and examine their effect on the community fitness function score. An initial method in this direction can be a simple unsupervised clustering in the known/predicted community embedding space to see if they form distinct natural clusters. We are also interested in exploring heuristics that remove nodes to see if they improve scores. More specifically, in each of the heuristics, for each predicted complex that contains 4 or more nodes, we remove a node one at a time to see if the score improves. Another idea is to incorporate removing a node at each step of the candidate subgraph growth.

3.10.3. Additional functionality

While we recommend cross-validation, we provide default hyper-parameters. We plan to also provide smarter hyper-parameters where possible, such as selecting the threshold number of steps as the maximum after ignoring outliers by using the interquartile rule (Q_3+ 1.5(Q_3-Q_1)).

3.10.4. Exploring other architectures and improving efficiency

Although our design performs well, there is an opportunity to further improve efficiency and also provide different architectures that are more efficient for other use-cases and different computing systems.

3.10.5. One-class methods

Instead of defining random walks to be negatives, we brainstormed the idea of using only positives. We first encountered a similar issue like this, when we were looking at training the edge weights of the co-complex protein-interaction networks using known co-complex interactions. Here as well, there is very little information as to what comprises non-co-complex interactions, and the methodology adopted was to use random pairings as negatives for binary classification. To do away with the idea of constructed negatives, we brainstormed and came up with one-class classification methods and explored one-class SVMs which are famous for applications such as anomaly detection. PU learning, *i.e.*, learning from positive and unlabeled samples is another interesting path to pursue in this direction. One-class SVMs can also be implemented to sample negative interactions [51].

Chapter 4. Molecular complex detection in protein interaction networks through reinforcement learning[3]

Candidate community search on a network has previously been performed with unsupervised and supervised learning methods by seeking to maximize a community fitness function in each step of growing a candidate community from a seed. Trying to achieve a high community fitness in each step misses potential communities whose subgraphs are not communities. The sequential addition of nodes to a seed node with the goal of achieving a community provides an opportunity to frame community detection as a reinforcement learning problem. In this chapter, we discuss RL complex detection, a community detection algorithm based on reinforcement learning that learns trajectories to walk on a network to achieve a community.

4.1. Abstract

Many, if not most, proteins assemble into higher-order complexes to perform their biological functions. Such protein-protein interactions (PPI) are often experimentally measured for pairs of proteins and summarized in a weighted PPI network, to which community detection algorithms can be applied to define the various higher-order protein complexes. Current methods, which include both unsupervised and supervised approaches, often assume that protein complexes manifest only as dense subgraphs, and in the case of supervised approaches, focus only on learning *which* subgraphs correspond to complexes, not *how* to find them in a network, a task that is currently solved using heuristics. However,

[3]This chapter is adapted from the paper, Palukuri, M. V., Patil, R. S., & Marcotte, E. M. (2022). "Molecular complex detection in protein interaction networks through reinforcement learning". bioRxiv. https://doi.org/10.1101/2022.06.20.496772. I supervised my mentee Ridhi Patil on this project. I was responsible for the algorithm conception and design, while Ridhi Patil wrote the main part of the code and performed experiments. Edward Marcotte and I provided ideas for improvement. All of us analyzed the candidate complexes with uncharacterized proteins. Edward Marcotte performed structural analysis of a candidate complex.

learning to walk trajectories on a network with the goal of finding protein complexes lends itself naturally to a reinforcement learning (RL) approach, a strategy that has not been extensively explored for community detection. Here, we evaluated the use of a reinforcement learning pipeline for community detection in weighted protein-protein interaction networks to detect new protein complexes. Using known complexes, the algorithm is trained to calculate the value of different possible subgraph densities in the process of walking on the network to find a protein complex. Then, a distributed prediction algorithm scales the RL pipeline to search for protein complexes on large PPI networks. The reinforcement learning pipeline applied to a human PPI network consisting of 8k proteins and 60k PPI results in 1,157 protein complexes and shows competitive accuracy with improved speed when compared to previous algorithms. We highlight protein complexes harboring minimally characterized proteins including C4orf19, C18orf21, and KIAA1522, suggest TMC04 to be a putative additional subunit of the KICSTOR complex, and confirm the participation of C15orf41 in a higher-order complex with CDAN1, ASF1A, and HIRA by 3D structural modeling. Reinforcement learning offers several distinct advantages for community detection, including scalability and knowledge of the walk trajectories defining those communities.

4.2. INTRODUCTION

Protein-protein interactions (PPIs) are essential to nearly all cellular functions and biological processes. From antibodies binding antigens to block infections to protein filaments comprising cellular cytoskeletons, protein interactions are an important organizational principle across biological scales and organisms. Such multi-protein complexes may additionally bind other molecules, such as DNA, RNA, or metabolites, and

play critical roles in cellular processes ranging from DNA replication to transcription to multicellular interactions to tissue organization.

As a consequence, a growing variety of experimental techniques have been developed to determine PPIs at large scale, notably including affinity purification/ mass spectroscopy (AP/MS), co-fractionation / mass spectrometry (CF/MS), cross-linking/ mass spectrometry (XL/MS), proximity labeling, and yeast two-hybrid assays (Y2H), which are collectively reviewed in [52]–[56]. The resulting PPIs define (often weighted) networks of interactions, in which each node represents a protein, an edge represents the interaction confidence, and certain proximal groups of nodes and edges correspond to multiprotein complexes. Importantly, the experimental methods are not completely accurate and suffer both false positive and negative observations. Hence, integrating PPIs across multiple experiments, as for *e.g.* the networks hu.MAP 1.0 [22] and hu.MAP 2.0 [57] that integrate over 9,000 and 15,000 mass spectrometry experiments respectively from AP/MS [58]–[61] and CF/MS data [62]–[65], can help to mitigate the effects of experimental errors. Combining such approaches with algorithms to cluster proteins and identify complexes from the PPI network should result in more accurate determination of protein complexes. Community detection algorithms can be applied to a PPI network to identify its communities, *i.e.*, protein complexes [66].

Community detection methods can be unsupervised, *i.e.*, not use any information from known communities in a network and instead rely only on the network topology to cluster it into its communities. Currently, existing unsupervised community detection algorithms tend to rely on many assumptions regarding the topological structures of communities. MCODE (Molecular COmplex DEtection) is an unsupervised method of detecting protein complexes running on the assumption that dense regions of a network represent complexes [67]. Another unsupervised algorithm, CMC (cluster-based on

maximal cliques), assumes that communities are mainly in the shape of cliques (again, highly dense subgraphs) [68]. This pattern of similar assumptions carries on to other unsupervised methods such as COACH (core-attachment-based method) [69], ClusterONE (clustering with overlapping neighborhood expansion) [70], and GCE (greedy clique expansion) [71], among others.

On the other hand, supervised community detection methods do consider different topological features of communities apart from density, and learn a community fitness function (*i.e.*, the probability of being a community) from known complexes using different learning algorithms. One such approach uses a support vector machine (SCI-SVM) and a Bayesian network (SCI-BN) [2]. For both models, subgraphs are represented using 33 features, and a local subgraph growth process is employed starting from a seed node, with the subgraph growth regulated by limited growth rounds, score improvement over iterations, and extent of overlap with other candidate communities. ClusterSS, cluster with supervised and structural information, is a supervised algorithm using a neural network, 17 subgraph features, and a structural scoring function [72]. All three methods, SCI-SVM, SCI-BN, and ClusterSS use a greedy heuristic algorithm for selecting the neighbor to add to the subgraph in the growth process, with ClusterSS considering only the top neighbors by degree for speed improvements. However, since the methods use serial candidate community sampling, this negatively impacts their scalability to large networks like hu.MAP 1.0 [22] with ~8k proteins and ~60k interactions, and hu.MAP 2.0 [57] with ~10k proteins and over 40k interactions. To combat this, Super.Complex (supervised complex detection algorithm) was developed for high scalability and accuracy [73]. By using AutoML (Automated Machine Learning), it explores different supervised algorithms to utilize the optimal one learned from known communities. Then, it samples candidate subgraphs using the learned fitness function by growing them with an epsilon-greedy

heuristic, along with one of four additional heuristics (pseudo metropolis, clique - pseudo metropolis, iterative simulated annealing, and greedy). However, the AutoML pipeline can take a long computation time and the method can still be improved in terms of accuracy.

While current supervised learning methods learn community fitness functions, they do not learn trajectories on the network that can lead to a protein complex, potentially missing out on complexes that cannot be traversed using the heuristics employed, for instance in a greedy setting where the community fitness function is maximized at each step. We can apply reinforcement learning to learn paths traversed on a graph to recall known complexes with high accuracy. MARL (Multi-Agent Reinforcement Learning) uses Q-learning to form clusters in networks based on a multi-agent environment [74]. In this algorithm, each node is viewed as an agent and each agent chooses actions to grow into a cluster. A team reward is given to train the action-value function, based on the modularity of the partition, which again assumes that all communities are dense subgraphs. The team reward can also lead to unstable learning of each agent's behaviors, apart from taking a long time to compute. Nevertheless, MARL has suggested that there is a lot of potential for the use of reinforcement learning in community detection algorithms.

Rather than using a multi-agent approach with a team reward based on modularity, an unsupervised measure, we use a single agent approach (where a subgraph is viewed as an agent) with a supervised reward from training communities based on whether or not the agent selects the right neighbor to grow the subgraph. The rewards are used to train a value iteration algorithm to learn an optimal value function that is used to predict new communities. During the prediction phase, we implement a parallel method with a single agent on each core (process) to increase the speed of the algorithm by predicting communities in a distributed fashion. Applying the reinforcement-learning (RL) algorithm on a human PPI network after learning from experimentally characterized complexes, we

can identify candidate protein complexes, and these candidates can then be experimentally characterized. We can create reliable models of protein complexes that allow us to extract more information about their stability, affinity, and specificity.

In the current work, we formulate community detection as a reinforcement learning task, and implement a value iteration algorithm, learning from known communities. The RL algorithm accurately and efficiently predicts candidate complexes by learning and using a value function from known communities, that maps the density of a subgraph to the probability (score) that traversing the subgraph will yield a protein complex. The algorithm trains on known complexes that have nodes and edges from the network, to accurately optimize scores for the various densities that could occur on the network. Then, the RL pipeline uses these scores to traverse the network by starting with different seed nodes to create candidate complexes in parallel. We apply the reinforcement learning algorithm to a human PPI network consisting of 8k proteins and 60k pairwise protein interactions, resulting in 1,157 candidate protein complexes, including complexes containing uncharacterized proteins that may suggest possible functions for these as yet understudied human proteins.

4.3. MATERIALS AND METHODS

4.3.1. Reinforcement learning

Reinforcement learning (RL) uses machine learning to enable an agent to make a sequence of decisions based on rewarding desired behaviors and punishing undesired ones that lead the agent to achieve some goal. These rewards reinforce the right decisions so that the agent repeats them. Over time, the algorithm finds the best possible decision or action to take in each situation. This process in RL makes it an intuitive method for community

detection as decision-making for long-term results is needed in the process of traversing a network with the final goal of finding a protein complex.

RL terminology (agent, actions, environment, state, reward, return, policy, and value) is best explained with an example. Consider a recommendation example, where an online retailer is interacting with a customer to sell his products. The online retailer is considered an *agent* who has to perform *actions* of showing different products to the customer, the *environment*, whose shopping cart is visible to the retailer as the customer's *state*. An *episode* is a sequential interaction between the *environment* and the *agent*, here, for instance, it begins when the customer enters the online shop and the retailer starts interacting by showing the customer an item recommendation for buying. The goal of the retailer is to maximize the recommendations the customer accepts. In the RL framework, *rewards* are designed to help achieve the goal. In the current interaction, the customer can add the item to the cart, giving the retailer a positive *reward*, or can ignore the item giving a negative *reward*. Based on the *state, i.e.*, the customer's cart, the *agent* now takes a new *action* (show a new product) to maximize his total reward, also termed as his *return*. The episode ends when the customer leaves. Based on multiple such interactions across different episodes, the agent learns what actions it needs to take to achieve its goals. In summary, an agent sequentially interacts with an environment of which it is aware by observing the environment's state. The agent performs actions and gets rewards from the environment based on its actions, and the agent tries to maximize its return, *i.e.*, the total reward it obtains by learning and using a policy.

A policy $\pi(s) -> \ a \epsilon A$ is an action-selection strategy, *i.e.*, given the agent is in a particular state *s*, the probabilities with which it should take different actions available (*A*). Finding an optimal policy is associated with optimizing some long-term performance measure, usually, maximization of the return or the total reward the agent would obtain by

following the policy. The discounted return at step *t* of an episode is the sum of rewards obtained after that step, with higher weights for immediate steps and lower weights for steps after a long time, *i.e.*,

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1} = R_{t+1} + \gamma G_{t+1}, \gamma \in [0,1], \quad (4.1)$$

where *t* is the step in an episode, R_t is the reward and γ is a discount factor. A value function is a score given to a state that estimates how beneficial it is for the agent to be in that state to achieve the goal. The value function *v*, of a state *s*, under policy π, is the expected (discounted) return if the agent were to take steps from the state *s*, *i.e.*,

$$v_\pi(s) \equiv E_\pi[G_t \mid S_t = s]. \quad (4.2)$$

4.3.1.1. Finding the optimal policy with the Bellman optimality equation

The optimal policy at a state is performing an action that takes the agent to the next best possible state that will maximize the probability of achieving the goal, *i.e.*, maximizing its return (cumulative reward). In other words, the optimal policy, of the states available, moves the agent to the state having the highest value. To learn an optimal policy, we learn the optimal value function, which indicates the optimal value of different states of the environment.

We now discuss how to find the optimal value function. A policy π is defined to be better than or equal to a policy π' if its expected return is greater than that of π' for all states, *i.e.*,

$$\pi \geq \pi' \equiv v_\pi(s) \geq v_{\pi'}(s) \; \forall \; s \in S. \quad (4.3)$$

Using this, the optimal value function is defined as,

$$v_*(s) \equiv \max_\pi v_\pi(s) \forall \; s \in S. \quad (4.4)$$

Any greedy policy w.r.t v_* is an optimal policy.

The action-value function is the expected (discounted) return when it starts in state *s*, takes action *a*, and then executes policy π, *i.e.*,

$$q_\pi(s,a) \equiv E_\pi[G_t \mid S_t = s, A_t = a]\,. \tag{4.5}$$

The optimal value function is given by,

$$v_*(s) = \max_{a \in A(s)} q_{\pi*}(s,a)\,. \tag{4.6}$$

Substituting the action value function and the return, we get,

$$\begin{aligned} v_*(s) &= \max_a E_{\pi*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \end{aligned} \tag{4.7}$$

On expanding the expectation using the model for environment dynamics, we get the Bellman Optimality Equation,

$$V^*(s_t) = max_{a_t} \left(\Sigma_{s_{t+1}} p(s_{t+1} \mid s_t, a_t) \left(R(a_t, s_t) + \gamma V^*(s_{t+1})\right)\right)\,. \tag{4.8}$$

Here, *R* is the defined reward associated with performing action a_t when the state is s_t, γ is a discount factor and *p* is the transition probability from s_t to s_{t+1} when it performs the action a_t.

4.3.1.2. Value iteration

The true or optimal value function, *i.e.*, a map of the states to their values can be learned using the value iteration algorithm, a classic reinforcement learning method for problems where a model of the environment dynamics is known, usually with a small number of discrete states. The algorithm is a dynamic programming method that solves the Bellman Optimality Equation (**Equation 4.8**) iteratively, converging to the optimal value function *V**.

The value iteration update rule, starting with a value of 0 for all states is given by,

$$V^*(s_t) = max_{a_t} \left(\Sigma_{s_{t+1}} p(s_{t+1} \mid s_t, a_t)(R(a_t, s_t) + \gamma V_{k-1}(s_{t+1}))\right)\,. \tag{4.9}$$

Here, $V_k\ (s_t)$ is the value function of the current state (at time *t* in the episode) in the current iteration *k* (of the value iteration algorithm) and $V_{k-1}\ (s_{t+1})$ is the value function (from the previous iteration *k-1*) of the next possible state.

4.3.2. Formulating community detection as a reinforcement learning problem

To successfully utilize an RL pipeline for the problem of community detection, the algorithm is first trained on known training communities or complexes. Once the training is deemed to be successful, the learned value function from the training is then used to find complexes on a network.

To learn the value function, each episode consists of starting with a seed node from a training complex and iteratively adding neighbors to grow the subgraph into the complex. This process is then repeated with a new seed node from the training complex. Once all the nodes of the complex have been used as seeds, training moves to the next complex. In this scenario, the agent and environment are defined as the current subgraph in the growth process and the full graph including all of its neighbors, respectively. We represent the state of the agent, *i.e.*, the current subgraph by its topological feature, density. The state (density *d*) of the current subgraph is the ratio of the actual number of edges in a subgraph to the total possible number of edges and can be calculated as follows,

$$d = \frac{2m}{n(n-1)} \tag{4.10}$$

Here, *m* is the sum of the edge weights of the edges in the subgraph and *n* is the number of nodes in the subgraph.

For a wide representation of the feature space, the states are discretized into 20 intervals ranging from 0 to 1. The actions performed by the agent comprise adding a neighbor to the current subgraph or terminating the growth process. Choosing a neighboring node in the known complex will provide the agent a positive reward of +0.2,

and a negative reward of -0.2 is given if the node chosen is not present in the known complex. The rewards aid the agent in avoiding previous mistakes for it to find an optimal path to create a complex. These rewards allow the state to develop a value function representing the probability of the state resulting in a final community. If none of the remaining neighbors are in the complex, the agent is encouraged to learn to terminate the growth process by receiving a reward of 0, as opposed to choosing a neighbor giving a reward of -0.2.

Once the training completes and an optimal value function is learned, the agent learns candidate complexes on the entire network by starting with seed nodes in parallel and adding neighbors giving the highest value function at each iteration, until the action of terminating a growth process gives a higher value than adding any of the neighbors.

4.3.2.1. Proof of applicability of RL to community detection

The environment is deterministic and the next state the subgraph moves into (s_{t+1}) is only dependent on the previous state (s_t), the current subgraph. It does not depend on any other states previously encountered by the agent, satisfying the Markov property with a memoryless process,

$$p(s_{t+1}|s_t) = p(s_{t+1}|s_t, s_{t-1}, \dots s_0)\,. \tag{4.11}$$

Here, on the left-hand side, p is the conditional probability of achieving a state given only the previous state, while on the right-hand side, the probability is conditional on all the previous states encountered. Therefore, with this formulation, community detection can be treated as a Markov Decision Process (MDP) and solved using reinforcement learning methods.

4.3.2.2. Value iteration for community detection

The value iteration update rule with our formulation of the community detection problem is given by,

$$V_k(s_t) = max_{a_t}\big(R(a_t, s_t) + \gamma V_{k-1}(s_{t+1})\big). \quad (4.12)$$

$V_k\ (s_t)$ is the value function of the current state (at time t in the episode) in the current iteration k (of the value iteration algorithm), R is the defined reward, γ is the discount factor (0.5), and $V_{k-1}\ (s_{t+1})$ is the value function (from the previous iteration $k-1$) of the next possible state. We obtain this simple update rule, derived from the Bellman optimality equation (**Equation 4.13**) using a transition probability $p(s_{t+1}|s_t, a_t)$ of 1 in **Equation 4.8**. A transition probability of 1 is used due to the deterministic nature of this problem, *i.e.*, the state transitions from s_t to only s_{t+1} when action a_t is taken.

$$V^*(s_t) = max_{a_t}\big(R(a_t, s_t) + \gamma V^*(s_{t+1})\big). \quad (4.13)$$

Here, V^* is the optimal value function, which the algorithm's value function converges to after a few iterations using the value iteration update rule (**Equation 4.12**), starting with a value of 0 for all states.

4.3.3. Reinforcement learning community detection algorithm

4.3.3.1. Overview

There are 3 main steps for community detection on a network using reinforcement learning:

1. **Training** the algorithm to walk across training complexes by learning a value function corresponding to each state (subgraph) encountered in the process.
2. **Finding** candidate complexes by using the learned value function to walk on the human protein-protein interaction network.
3. **Benchmarking** learned complexes against known complexes.

4.3.3.2. Training the algorithm

For training the RL pipeline, we use the known training complexes and represent each complex as a target subgraph of the entire protein-interaction network. The agent, *i.e.*, the current subgraph expands by adding, at each step, the neighbor that yields the highest value for the current subgraph (the value is calculated using the reward given for adding this node and the value of the next state, as shown in **Equation 4.12**). The algorithm updates the value of the density of the current subgraph to this new value (**Equation 4.12**). Each time a state (density) is encountered in the process of training on multiple training complexes, the value of that state is updated using the update rule (**Equation 4.12**), moving towards convergence of the value function and eventually learning a value function mapping densities to their probability of leading to a protein complex.

Figure 4.1 shows an example of learning the value function while traversing a single known complex. The initial subgraph or seed is an edge between an initial random node and the node's neighbor with whom the node shares an edge with the highest edge weight. To calculate the potential value function of the current subgraph, *i.e.*, the term within the *max* in **Equation 4.12** for each neighbor of the current subgraph, each neighbor is temporarily added to the subgraph. The density of that temporary subgraph is calculated, followed by querying the value function for that state along with the reward based on whether or not the neighbor is present in the final protein complex. After calculating the potential value function, the neighbor is removed from the subgraph for this process to be repeated for the rest of the neighbors. Once all of the neighbors have been evaluated, the value function of the current state is updated and the neighbor yielding the state that provided the maximum value function is added to the subgraph. This new subgraph will be the new "seed" as this process repeats itself. The subgraph, or complex, will be "complete" and the algorithm stops adding nodes if all the neighboring nodes return lower value

functions than the action of terminating the growth process, represented by adding an imaginary node with reward 0, leading to the same state as before, indicating that no new neighbors should be in the complex. Note how a reward of 0 encourages the algorithm to terminate the growth process when all other options are adding the wrong nodes, *i.e.*, choosing actions that have a reward of -0.2. Conversely, if a correct node is available, its reward of 0.2 encourages choosing that node over terminating growth. Once a subgraph is deemed as complete, the process is repeated starting with a different node of the same complex, so that other trajectories to build the same complex are explored and the value function is updated accordingly. After starting with each of the nodes of a complex as seeds, training moves to the next complex, starting with a random new seed from this complex, and the process repeats.

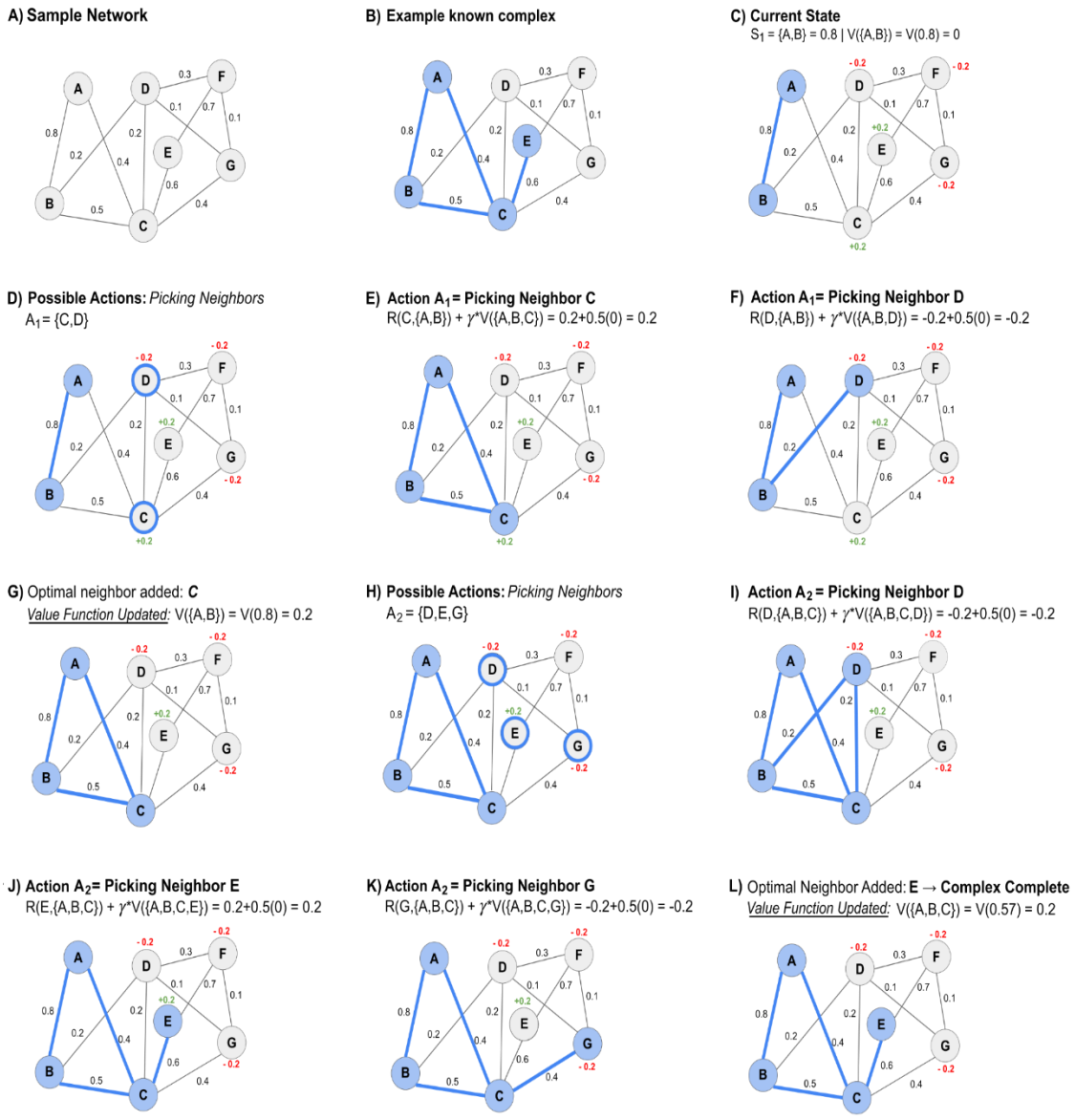


Figure 4.1. Example trajectory of training the RL pipeline on a sample network by learning a value function.

(A) This network comprises 7 nodes and 11 weighted edges **(B)** A known complex consists of the nodes A, B, C, and E. **(C)** First, a seed edge (A, B) is identified to begin the walk. At {A, B}, the state (density) $S_1 = 0.8$ and the value function $V(\{A, B\}) = V(0.8) = 0$ (since values of all densities are initialized to 0). When a node is added to the current subgraph, a reward of +0.2 is given if the node is present in the training complex and a

reward of -0.2 is given if absent. **(D)** We evaluate all possible neighbors *i.e.*, C and D, to add to the current subgraph {A, B}. Using the value iteration update rule with a discount factor γ of 0.5, we add each neighbor as the next state and compute a corresponding value for the current state. **(E)** The temporary complex {A, B, C}, corresponds to $S_2 = 0.57$ and $V(\{A, B, C\}) = V(0.57) = 0$ (since this state has not been encountered before). Using the reward for node C (+0.2), adding node C to the complex {A, B} would give $V(\{A, B\}) = V(0.8) = 0.2$. **(F)** The temporary complex {A, B, D} corresponds to $S_2 = 0.33$, $V(\{A, B, D\}) = V(0.33) = 0$. Using the reward for node D (-0.2), adding node D to the current complex {A, B}, would give $V(\{A, B\}) = V(0.8) = -0.2$. **(G)** Once all neighbors have been evaluated, the neighbor providing the highest value function (C) is added to the candidate complex. The value function of the original state $S_1 = \{A, B\} = 0.8$ is updated from 0 to +0.2. **(H)** Again, we evaluate all possible neighbors of the newly updated complex {A, B, C}, *i.e.*, D, E, and G, to consider the possible actions the agent can take. **(I)** The temporary complex {A, B, C, D} corresponds to $S_3 = 0.35$, $V(\{A, B, C, D\}) = 0$. Using the reward for node D (-0.2), adding node D to the current complex {A, B, C} gives $V(\{A, B, C\}) = V(0.57) = -0.2$. **(J)** The temporary complex {A, B, C, E} corresponds to $S_3 = 0.38$, $V(\{A, B, C, E\}) = 0$. Using the reward for node E (+0.2), adding node E to the current complex {A, B, C}, results in $V(\{A, B, C\}) = V(0.57) = 0.2$. **(K)** The temporary complex {A, B, C, G}, corresponds to $S_3 = 0.35$, $V(\{A, B, C, G\}) = 0$. Using the reward for node G (-0.2), if we add node G to the current state of the complex {A, B, C}, the resulting value function is -0.2. **(L)** Node E provides the highest value function, so it is the optimal neighbor and is added to the complex. The value function of the complex state $S_2 = \{A, B, C\} = 0.57$ is updated from 0 to +0.2. This process of evaluating neighbors is repeated until the action of terminating the growth process is chosen, represented by adding an imaginary node with reward 0, leading to the same state as before. As the remaining neighbors D, F, and G have a reward of -0.2, the imaginary node is chosen as it results in a value function (0.1) higher than the neighbors (D: -0.2, F: -0.2, G: -0.2). At this point, the candidate complex is finalized {A, B, C, E} and the process stops. A new seed edge is chosen from the network and this process repeats, updating the same value function.

4.3.3.3. Finding candidate complexes

Once we observe that the value functions of different states have converged in the training process, we can use the learned value function to walk paths on the network to find complexes.

For each node of the network, we choose its corresponding highest edge-weight as a seed edge to grow a candidate complex. For each seed edge, the neighbors are evaluated and the neighbor yielding the subgraph with the maximum value function is added. The growth process stops when adding any neighbor lowers the value function of the current subgraph. For instance, consider a subgraph with a value function of 0.2. If on evaluating each of the subgraph's neighbors, each of them returns a value function lesser than 0.2, the algorithm stops and the subgraph will be considered a candidate complex. This process is repeated for the next seed edge in the network. These steps are detailed in **Algorithm 1** and an example of finding a complex on the network is shown in **Figure 4.2**.

Algorithm 1. Finding candidate complexes:

For each seed edge in the network:

1. Using the edges of the network, all the neighbors of the seed edge are found.
2. For each neighbor:
    - It is temporarily added to the subgraph.
    - The density (state) is calculated.
    - The corresponding value of the state is obtained from the learned value function and is noted.
    - The neighbor is removed.
3. The neighbor that resulted in the complex having the highest value function is added if this value is greater than the value function of the current subgraph, otherwise, the growth process terminates and the current subgraph is returned as a candidate complex.

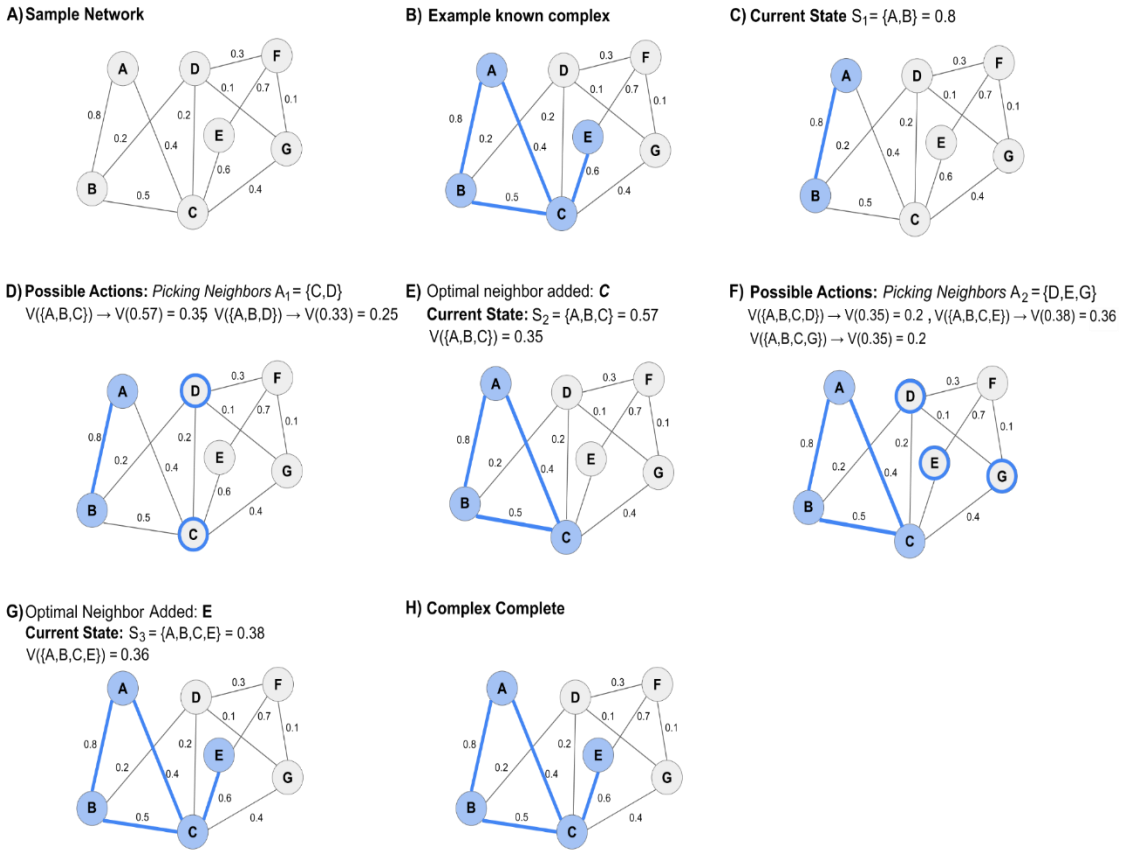


Figure 4.2. Example trajectory for finding a complex on a sample network with the RL pipeline using a learned value function.

(A). This network comprises 7 nodes and 11 edges, with edge weights shown next to each corresponding edge. **(B)** In this network, a known complex consists of the nodes A, B, C, and E. The goal of the algorithm is to predict this known complex from the network using the learned value function. **(C)** The first step is to identify a seed edge to begin the walk (edge AB, with an edge weight of 0.8). At this seed edge, the complex is at state (density) $S_1 = 0.8$. **(D)** Then, we evaluate all possible neighbors of nodes A and B, *i.e.*, C and D. Adding node C gives a temporary complex {A, B, C} with $S_2 = 0.57$, and a learned value $V(\{A, B, C\}) = V(0.57) = 0.35$, while adding node D gives a temporary complex {A, B, D} with $S_2 = 0.33$, $V(\{A, B, D\}) = V(0.33) = 0.25$. **(E)** The neighbor with the highest value function is node C and hence, node C is added to the subgraph resulting in $S_2 = 0.57$, $V(\{A, B, C\}) = V(0.57) = 0.35$. **(F)** The next neighbors of this updated complex are evaluated, *i.e.*, D, E, and G. Adding node D leads to $S_3 = 0.35$, $V(\{A, B, C, D\}) = V(0.35) = 0.2$, node

E results in $S_3 =$ 0.38, $V(\{A, B, C, E\}) = V(0.38) = 0.36$, and node G leads to $S_3 =$ 0.35, $V(\{A, B, C, G\}) = V(0.35) = 0.2$. **(G)** Since the neighbor yielding the highest value function is node E, this node is added to the complex resulting in $S_3 = 0.38$, $V(\{A, B, C, E\}) = V(0.38) = 0.36$. **(H)** Each of the remaining neighbors (D, F, and G) results in a value function less than that of the current complex {A, B, C, E}. Thus, no neighbor is added and the predicted community is complete.

4.3.3.4. Post-processing and evaluation

Once we have found all the candidate subgraphs corresponding to the specified seed nodes (in our experiments we use all the nodes of the graph as seed nodes), we perform a post-processing step to merge highly overlapping complexes. Adapting the pairwise merging algorithm in Super.Complex [73], if the overlap of two complexes is more than a specified threshold, we retain the complex with the highest value function of the two complexes and the merged variant and remove the others. To obtain the optimal overlap threshold, similar to [73], we test various thresholds for the *Qi overlap* measure (**Equation 4.14**) and choose the threshold which gives the highest F-similarity-based Maximal Matching F-score (FMMF) (**Figure 2.1**).

$$\text{Qi overlap measure: } \frac{|\, C_p \cap C_k \,|}{|\, C_p \,|} > t \text{ and } \frac{|\, C_p \cap C_k \,|}{|\, C_k \,|} > t \qquad (4.14)$$

Here, t is a user-specified overlap threshold, C_p is a predicted complex and C_k is a known complex.

To gauge the accuracy of the RL algorithm, we use different evaluation measures to compare learned complexes with known complexes. The learned complexes are compared with the known complexes after removing nodes missing in the set of known complexes. We employ a variety of evaluation measures such as the FMMF, Community-wise Maximum F-similarity-based F-score (CMMF), and Unbiased Sn-PPV Accuracy (UnSPA) (**Figure 2.1**), in addition to the Qi et al F-score **(Equation 2.7)** [2], F-grand k-clique and F-weighted k-clique [22].

As discussed in [73], one of the more accurate methods for comparison is the FMMF which combines an adapted Maximal Matching Ratio (MMR) with its corresponding precision. The adapted MMR is computed as the fraction of known

complexes matching predicted complexes, where the matches are computed as the sum of edge weights of a maximal matching in a bipartite graph between learned and known complexes. The edge weights used in the graph, matching learned and known complexes, are the F-similarity scores (*F1*), given by,

$$\frac{1}{F1} = \frac{1}{p'} + \frac{1}{r'}, \tag{4.15}$$

$$where, \quad p' = \frac{|C_p \cap C_k|}{|C_p|}, \tag{4.16}$$

$$and \quad r' = \frac{|C_p \cap C_k|}{|C_k|}. \tag{4.17}$$

Here, C_p is a predicted complex and C_k is a known complex.

4.3.3.5. Experimental details - synthetic and protein interaction datasets.

We first test the RL algorithm on a synthetic toy network (**Figure 4.3**) with 62 nodes and 78 edges, comprising 14 complexes. Of these 14 complexes, 7 are used as training complexes and 7 are used as testing complexes. The testing and training evaluation results can be found in (Supplementary) **Table 4.4**.

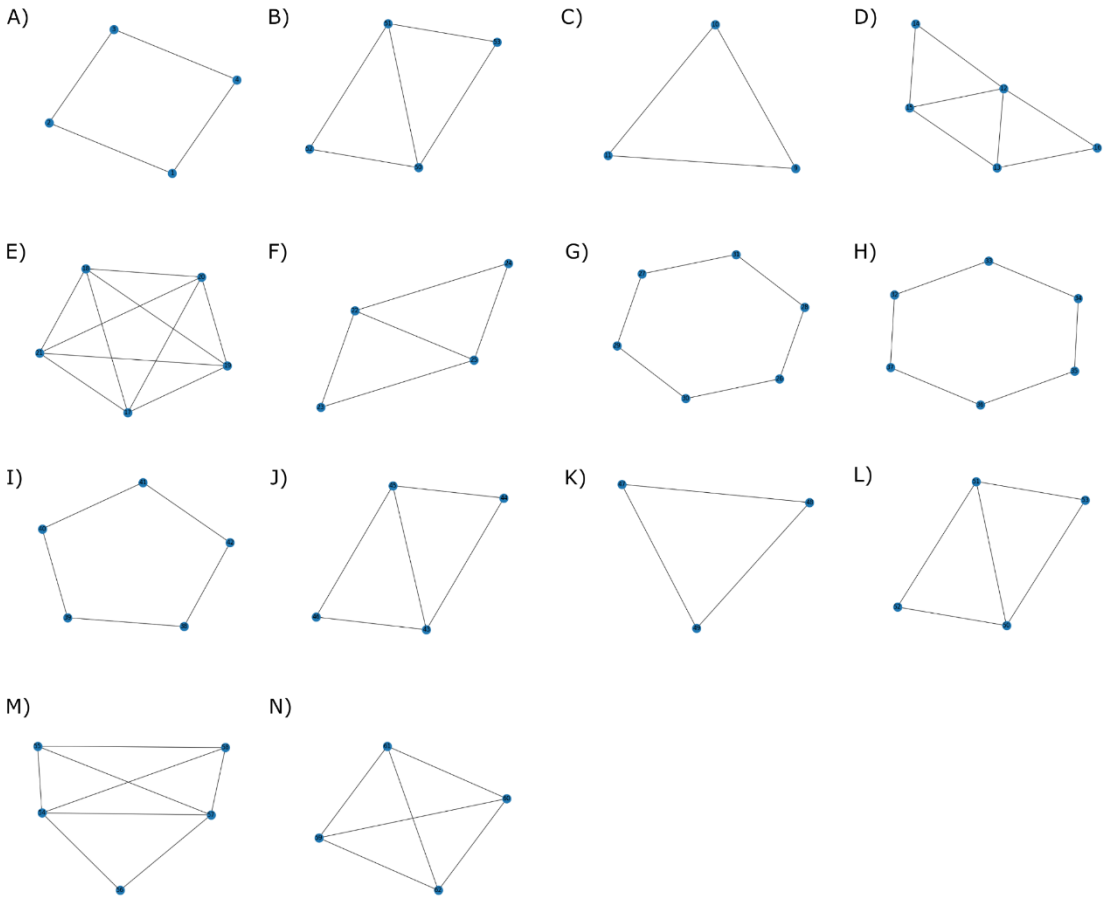


Figure 4.3. A synthetic disconnected toy network of complexes.

Complexes A-G are used for training and H-N are used for testing the RL algorithm.

Next, we apply the RL algorithm on a human protein interaction network (hu.MAP 1.0 [22]) comprising 7778 nodes and 56,712 edges to learn candidate complexes using 188 known complexes; these complexes are obtained by pre-processing the CORUM protein complex database [28]. The pre-processing primarily involves discarding complexes that are internally disconnected or have fewer than 3 nodes, and merging complexes with a pairwise overlap of more than 0.6 Jaccard coefficient (see methods section of Super.Complex [73]). For a perfect comparison, we use the same pre-processing steps as in Super.Complex for both the network and complexes, as well as the same training and testing complexes (all input data was obtained from Super.Complex input data [73]). Since we only use the density feature in our algorithm, we also run the Super.Complex pipeline with only the density feature. The testing and training evaluation results can be found in (Supplementary) **Table 4.5**.

The RL algorithm is also tested on hu.MAP 2.0 [57], a human PPI network consisting of 10,433 nodes and 43,581 edges, obtained by considering only the edges with a weight of at least 0.02. Again, to compare with Super.Complex, we use the same training and testing complexes from hu.MAP 1.0, and the same preprocessing steps. We perform two experiments; in the first experiment, we transfer the value function trained on hu.MAP 1.0 and in the second, we train a new value function on hu.MAP 2.0. The testing and training evaluation results for hu.MAP 2.0 can be found in (Supplementary) **Table 4.6** in the supplement.

4.4. RESULTS AND DISCUSSION

4.4.1. The value function converges in the training phase

During the training phase, we track the value for each encountered state (density) over time. Once the value of each of the states starts to converge, it can be assumed that

the value has reached its optimum. **Figure 4.4** demonstrates the successful convergence of the value for each density. We also investigate the relationship between a state's density and its value. **Figure 4.5** shows that in the path to a final complex, subgraphs of higher densities are favored since they have higher values.

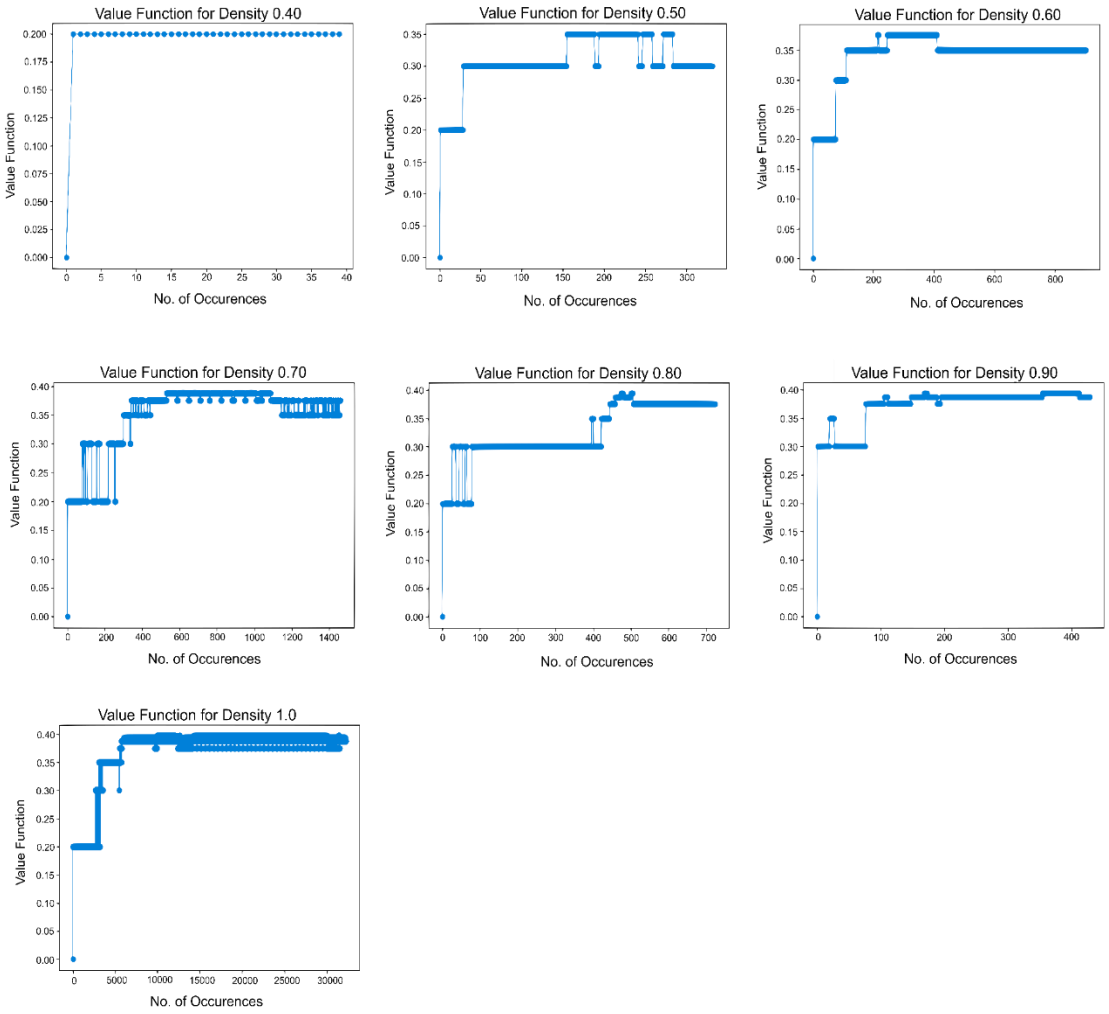


Figure 4.4. Convergence of the value for each density.

Each density (0.4 through 1) encountered in the training for hu.MAP 1.0 is plotted to see how its value updates over iterations. Although the values fluctuate initially, they converge eventually indicating successful training.

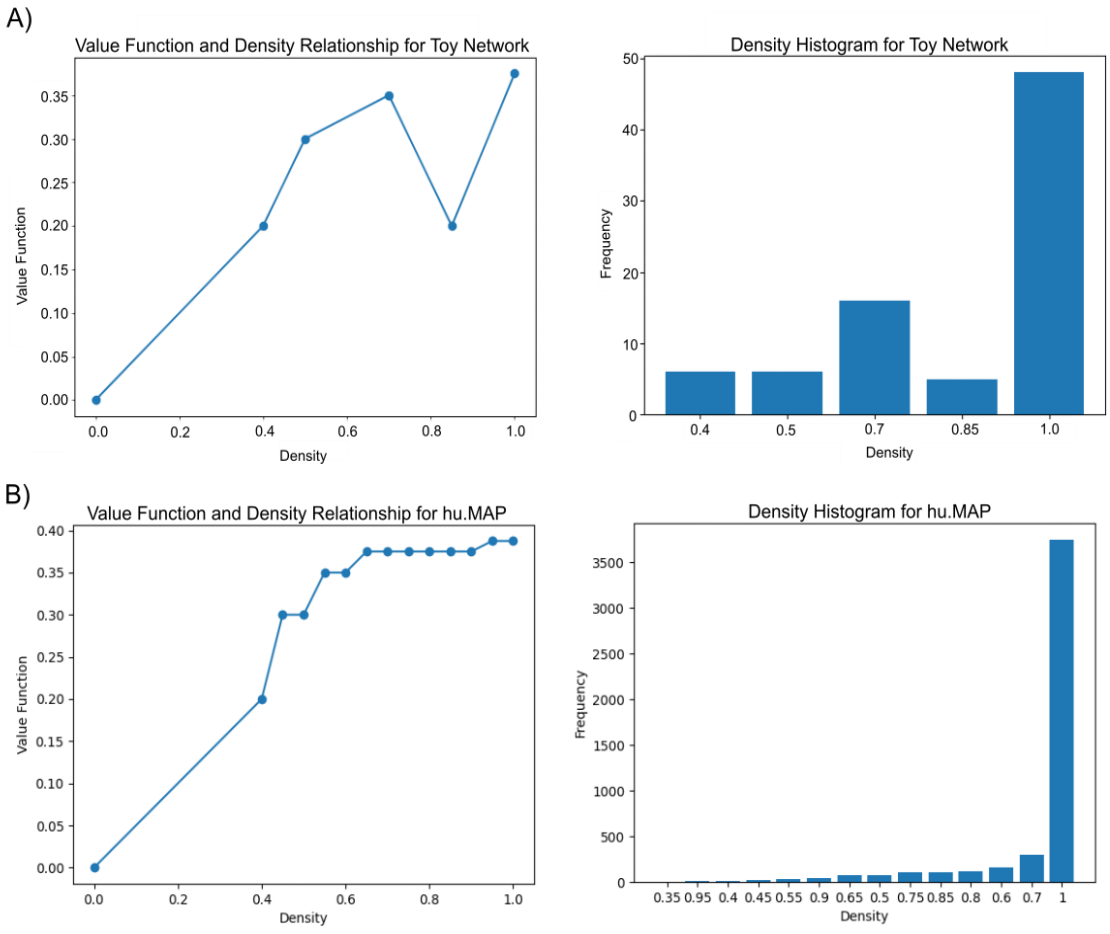


Figure 4.5. Higher values favor higher densities.

(A) The graph on the toy network shows a positive correlation between value function and density and therefore, complexes and trajectories with higher densities are favored. The density histogram for the training phase shows a higher frequency for higher density subgraphs. **(B)** For hu.MAP 1.0, the graph shows a stronger correlation between density and value function. The density histogram again shows that higher densities are more frequently observed.

The learned value functions for the synthetic dataset and hu.MAP 1.0 enable us to accurately predict complexes on the respective networks, as shown in the next section. Further, employing transfer learning, we use the value function learned on hu.MAP 1.0 to accurately predict complexes on hu.MAP 2.0 (Supplementary **Table 4.6**). This demonstrates that the value functions learned by the RL algorithm can be transferred for community detection problems on similar networks. We also directly train a value function on hu.MAP 2.0 using the same training complexes, and find that predicting complexes on hu.MAP 2.0 with this value function also gives accurate results (Supplementary **Table 4.6**).

4.4.2. The RL algorithm learns accurate communities on synthetic and real datasets

Recall the synthetic dataset containing 14 communities used to test the RL algorithm. The performance of the algorithm across different evaluation measures is excellent as summarized in **Table 4.1**. Next, we apply the algorithm to the real dataset, hu.MAP 1.0, by training it on 132 complexes. We test different Qi overlap thresholds (**Figure 4.6A**), in the RL algorithm, to merge highly overlapping complexes. The peak in **Figure 4.6A** occurring at 0.325 Qi overlap measure corresponds to the best FMMF score. For this value of the Qi overlap measure, the RL algorithm learns 1,157 complexes.

Table 4.1. The RL algorithm has a strong performance on the synthetic toy dataset.

The algorithm was trained on 7 toy complexes from a synthetic network of 62 nodes and 78 edges. It predicted 14 complexes which are evaluated against the 14 true complexes.

	FMM Precision	FMM Recall	FMM F-score	CMMF	UnSPA	Qi et al F1 score	SPA	F-weighted K-Clique
RL Algorithm	0.963	0.963	0.963	0.963	0.969	1.00	0.959	1.00

Abbreviations:

- FMM: F-similarity-based Maximal Matching;
- CMMF: Community-wise Maximum F-similarity-based F-score;
- UnSPA: Unbiased Sn-PPV Accuracy;
- SPA: Sn-PPV Accuracy.

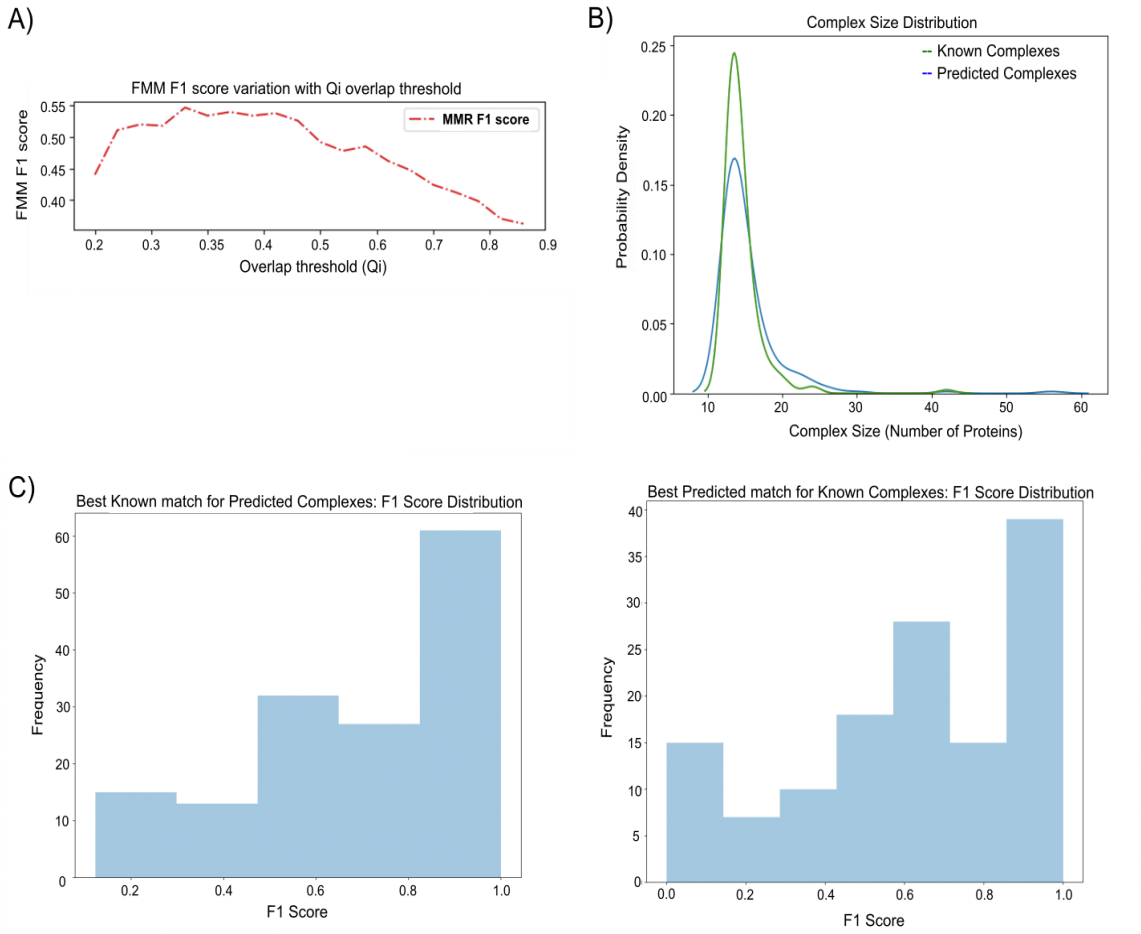


Figure 4.6. Evaluating the predictions of the RL algorithm on hu.MAP 1.0.

(A) The optimal Qi threshold is 0.325. We tested various overlap thresholds, *i.e.*, Qi values (**Equation 7**) between 0.2 and 0.9 in 0.25 intervals. **(B) Size distributions of known and predicted complexes.** This graph shows that the distribution of the sizes (no. of proteins) of the predicted and known complexes is very similar. (**C) F1 score distributions of the best-predicted match for known complexes and vice-versa.** In both cases, higher F1 scores have higher counts indicating accurate predictions.

For a perfect comparison, Super.Complex is tested on hu.MAP 1.0 using only the subgraph feature density. The best results from Super.Complex are obtained using a k-nearest neighbors classifier (with k = 76) to train a community fitness function, and from a search process for candidate complexes using maximal cliques as starting seeds and a pseudo-metropolis heuristic (with a probability of 0.1) for complex growth (with exploration probability ϵ 0.01). The candidate complexes are then merged with an overlap threshold of 0.2 Jaccard coefficient to yield 798 final complexes. In contrast, the RL algorithm predicts a higher number (1157) of complexes possibly resulting in a slightly higher (FMM) recall measure (**Table 4.2)**. The RL method achieves comparable performance to Super.Complex (**Table 4.2**), demonstrating the potential of applying reinforcement learning to community detection.

Table 4.2. The RL algorithm yields competitive accuracy compared to Super.Complex on hu.MAP 1.0.

The learned complexes on hu.MAP 1.0 are evaluated against all the known cleaned CORUM complexes.

	FMM Precision	**FMM Recall**	**FMM F-score**	**CMMF**	**UnSPA**	**Qi et al F1 score**	**SPA**	**F-weighted K-Clique**
RL Algorithm	0.612	0.482	0.547	0.654	0.772	0.559	0.652	0.988
Super.Complex	0.835	0.457	0.591	0.720	0.803	0.658	0.692	0.999

Abbreviations:

- FMM: F-similarity-based Maximal Matching;
- CMMF: Community-wise Maximum F-similarity-based F-score;
- UnSPA: Unbiased Sn-PPV Accuracy;
- SPA: Sn-PPV Accuracy.

Comparing **Tables 4.1** and **4.2**, we observe better accuracies for the algorithm on the toy dataset than those on hu.MAP 1.0. This could be attributed to the algorithm being better suited to finding non-overlapping communities, such as the toy communities, when compared to finding overlapping communities, such as the CORUM complexes on hu.MAP 1.0. Due to the current framework, the RL algorithm may not train well on overlapping communities because it trains on only one community in each episode, by giving a negative reward on adding a neighbor that does not belong to the community, even if it belongs to another overlapping community.

While accuracies are comparable, we note that the RL algorithm achieves faster running time relative to Super.Complex (**Table 4.3**). Specifically, the RL algorithm trains for ~9s on one core of a personal computer (M1 chip @ 3.2 GHz), making the training significantly faster than Super.Complex's training with the AutoML pipeline, which runs for ~540s on 20 cores of a supercomputer (Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz). Growing the candidate communities took ~300s when running in parallel across 8 cores (3.2GHz) for the RL algorithm, compared to ~20s when running in parallel across 72 cores (2.3GHz) for Super.Complex. This indicates that growing new complexes is also fast in the RL algorithm due to the simple inference using the value function lookup. We employ the 4 heuristics available in Super.Complex, with default parameters, to find the best one and perform a parameter sweep of 7 merge overlap thresholds for each heuristic. In total, we evaluate 28 heuristic-parameter combinations in Super.Complex; the same number of overlap thresholds used in the RL method. Overall, with the best parameters, the RL algorithm took ~350s on the personal computer with 8 cores (note that only the prediction step is parallelized here), compared to Super.Complex which took ~ 650s on a supercomputer with 72 cores (note that both the learning and the prediction step are parallelized here).

Table 4.3. The RL algorithm achieves a faster running time when compared to Super.Complex.

The time reported here corresponds to runs using the best overlap threshold found for both methods and also using the best heuristic found in the case of Super.Complex.

Method	Processor specifications	Training		Prediction		Post-processing		Total time (s)
		No. of cores	Time (s)	No. of cores	Time (s)	No. of cores	Time (s)	
RL Algorithm	M1 chip @ 3.2 GHz	1	9	8	320	1	11	340
Super.Complex	Intel(R) Xeon(R) E5-2699 v3 @ 2.30GHz	20	540	72	17	1	112	669

The average time complexity of the prediction phase of the RL algorithm is $O(G^2K^2S/P)$, where G is the average number of nodes in a complex, K is the average degree of the network and S is the number of seeds chosen (in our experiments, S is the number of nodes in the network). For Super.Complex with all subgraph features, the time complexity of the prediction phase is $O(XG^4KS/P)$. We note that the prediction phase of the RL algorithm scales better than that of Super.Complex. This is because the complexity of the subgraph feature extraction step reduces from $O(G^3)$ in Super.Complex to $O(GK)$ in the RL algorithm; this reduction happens since the RL algorithm uses only the feature density with a constant model inference time (X). The time complexity of the RL training algorithm is $O(G^3K^2T)$, where T is the number of training complexes. In contrast, the training complexity of Super.Complex is $O(G^3Tgpm/c)$, where g is the number of generations, p is the population size, m is the number of machine learning models and feature preprocessor types tried, and c is the number of processes on the single compute node running the AutoML step.

We note that the RL algorithm can be very useful in community detection problems with a small number of known complexes, as demonstrated in our experiments (7 and 132 training complexes in the synthetic and real datasets). Even if the number of known complexes is small, for each complex, the value iteration procedure in the RL algorithm explores several trajectories to learn the complex, incidentally increasing the size of the training dataset used to learn the complexes. On the other hand, existing supervised community detection methods train on a dataset with a size equal to the number of training complexes. Other benefits of the RL algorithm include the lack of need for extensive hyperparameter tuning and the ability to predict complexes that do not contain smaller complexes. For comparison, in Super.Complex, at each stage of growth in a candidate complex, the pipeline seeks to yield a final protein complex, attempting 4 different

heuristics, each with 1-2 hyperparameters. Contrastingly, the RL pipeline learns and traverses the optimal trajectory to find a complex without optimizing for intermediate complexes, and without the need for heuristics, thus saving on searching for parameters in the candidate complex growth step. Thus, the RL algorithm finds the best sequence of steps to grow a complex, while also being efficient.

In summary, relative to more sophisticated supervised ML strategies, the simplicity of the value iteration algorithm and the comparable accuracy along with improved efficiency demonstrates the great potential of the RL algorithm for solving community detection problems.

4.4.3. The resulting clusters suggest functions for uncharacterized proteins

Importantly, the RL algorithm returns many well-known human protein complexes accurately (as would be expected from the precision measurements on withheld test complexes), several of which are illustrated in **Figure 4.7**. Notably, the algorithm also identifies several additional interaction partners and even potential new subunits within these systems, such as, for example, clustering the guanine nucleotide exchange factor RCC1L with proteins of the mitochondrial ribosome large subunit, consistent with a known role for RCC1L in mitochondrial ribosome biogenesis [75]. Similarly, the RL algorithm recapitulates the nutrient-response-related KICSTOR complex (SZT2, KPTN, ITFG2, and C12orf66) [76] on both hu.MAP 1.0 (SZT2, KPTN, ITFG2, TMCO4 and C12orf66) and hu.MAP 2.0 (SZT2, KPTN, ITFG2, TMCO4, BMT2, and KICS2), suggesting the uncharacterized transmembrane protein TMCO4 to be a potential new interaction partner, and it reconstructs the WAVE1/WAVE2 protein complexes, known regulators of actin filament and lamellipodia formation [77], [78] while also suggesting involvement of KIAA1522, consistent with a recent suggestion for its involvement by Cho and colleagues

[79]. In order to investigate the identified complexes interactively, visualizations are available for the 1157 learned complexes on the supporting website (see the Code and data availability section).

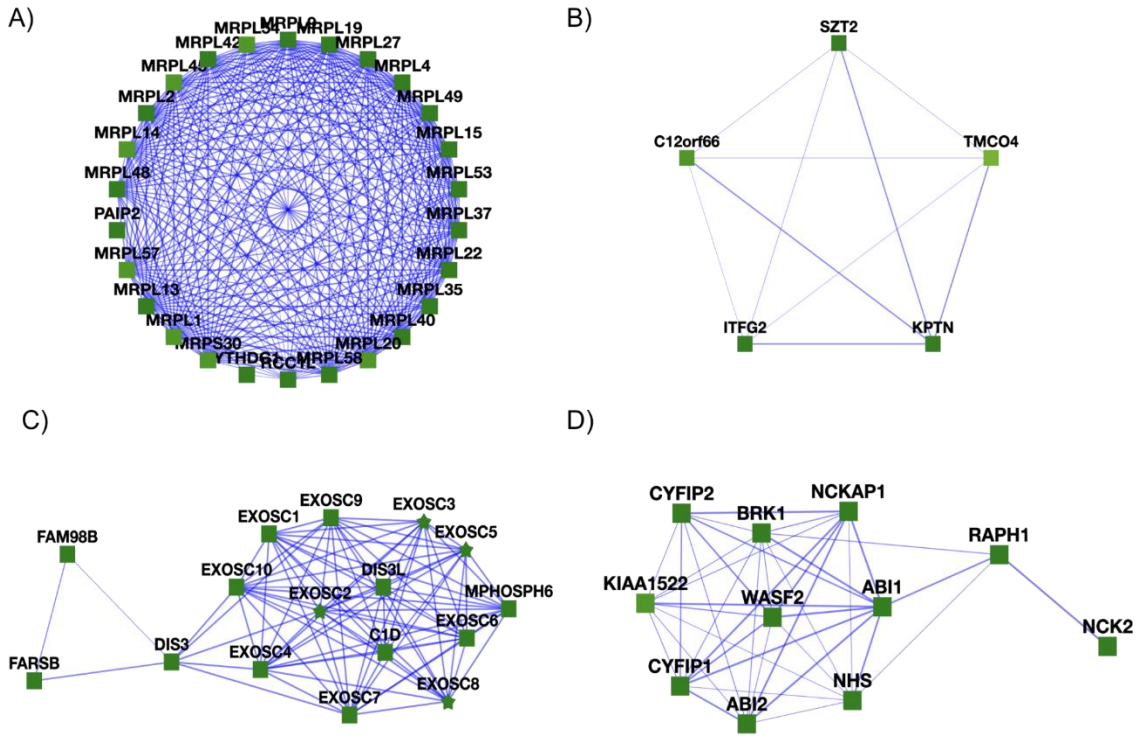


Figure 4.7. Learned complexes from the RL pipeline.

(A) The large (39S) subunit of the mitochondrial ribosome is present in the RL-determined complexes, but broken up into multiple complexes, one of which is shown here. Note the presence of the guanine nucleotide exchange factor RCC1L, which is known to be essential for mitochondrial ribosome biogenesis [75]. **(B)** RL recapitulates the KICSTOR complex (C12orf66, KPTN, ITFG2, and SZT2), a multiprotein complex known to regulate mTORC1 and cells' responses to available nutrient levels [76], but finds one additional putative subunit, the uncharacterized transmembrane protein TMCO4. **(C)** The exosome RNA processing complex is well-reconstructed by RL, with additional interactions observed to the tRNA synthetase FARSB and to FAM98B, a component of tRNA splicing ligase, consistent with possible associations among these systems [80]. **(D)** The WAVE1/WAVE2 protein complexes, known to regulate actin filament and lamellipodia formation [77], [78], are reconstructed by RL, along with evidence for interaction with the uncharacterized protein KIAA1522. Notably, KIAA1522 was recently suggested by Cho and colleagues to bind WAVE and participate in a community of associated actin-organizing proteins [79].

Of particular interest are complexes corresponding to proteins with low annotation scores, as finding the proteins in complexes with better-annotated proteins may help suggest potential functions for these otherwise minimally characterized proteins [81]. We searched specifically for such cases and highlighted complexes with uncharacterized proteins based on available UniProt annotations [33]. Some examples of learned complexes with uncharacterized or minimally characterized proteins are provided in **Figure 4.8**.

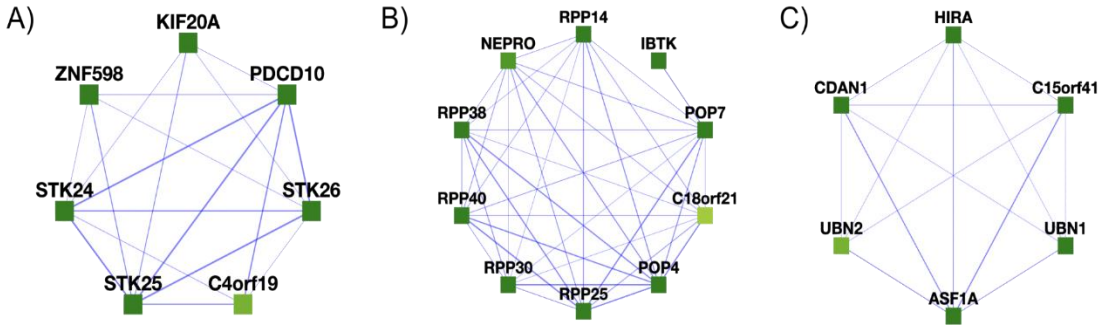


Figure 4.8. Participation in protein complexes by the uncharacterized proteins C4orf19 and C18orf21 and the minimally characterized protein C15orf41.

(A) We find C4orf19 to belong to a larger complex composed of KIF20A, C4orf19, PDCD10, STK25, ZNF598, STK26, and STK24. **(B)** C18orf21 is found in a complex with 50% similarity to the Rnase/Mrp complex. **(C)** C15orf41 is found in a complex with 30% similarity to the cytosolic Codanin-1-Asf1- H3.1-histone H4–importin-4 complex.

For example, in **Figure 4.8A**, C4orf19 (chromosome 4 open reading frame 19) is broadly expressed across human cell types and tissues [82], with high protein levels in the kidney, liver, and GI tract [34][83] and while little is known about its function, an observed relationship between C4orf19 and colorectal cancer suggests that high expression levels might have some value as a marker for colorectal cancer [84], although elevated C4orf19 expression is also reported to show a favorable association with renal cancer survival [34][83]. Notably, four of the other proteins in this cluster (PDCD10, STK24, STK25, STK26) are known to associate into a complex with roles in maintaining epithelial integrity [85], [86] and kidney water balance by regulating aquaporin trafficking and abundance in kidney tubule epithelial cells [87], suggesting a potential role for C4orf19 in normal kidney function. As for C4orf19, many of the proteins have been reported as potential biomarkers for bladder, gastric, pancreatic, and colorectal cancers [88]–[91].

As another example of a minimally characterized protein, C18orf21 (chromosome 18 open reading frame 21) is reported to possibly regulate the Rnase/Mrp complex, a ribonucleoprotein complex involved in RNA processing [92]. Both RL algorithm and Super.Complex concur on a connection for C18orf21 to RNA processing: from the learned complexes of Super.Complex, C18orf21 was found to be a part of a complex with a 50% overlap to the Rnase/Mrp complex, comprising all the proteins found in the RL algorithm's learned complex (**Figure 4.8B**), adding support for this protein's possible function in ribonuclease P RNA binding. Further, the RL algorithm learns a similar complex (C18orf21, IBTK, RPP30, POP4, and RPP25L) on hu.MAP 2.0 adding additional support to C18orf21's function from the learned complexes on hu.MAP 1.0.

Somewhat more information can be gleaned for C15orf41 (chromosome 15 open reading frame 41), which, while minimally characterized, has recently been detected to interact with Codanin-1 (CDAN1) in human cells, and this interaction forms a tight, near

stoichiometric complex [93]. Moreover, these studies reveal that mutation of C15orf41 can lead to the development of Congenital Dyserythropoietic type 1 disease (CDA-1) [93]. While its function is unknown, studies have noted a high sequence similarity between C15orf41 and archaeal Holliday junction resolvases, which are DNA repair enzymes that remove Holliday junctions [93], and it has been implicated in erythrocyte differentiation [94]. Its putative interaction partners within the complex (**Figure 4.8C**), HIRA and ASF1A, cooperate to promote chromatin assembly [95], and HIRA, ASF1A, and UBN1/2 form a complex and function in histone deposition of variant H3.3 into chromatin, independent of DNA replication [96]. CDAN1 and C15orf41 mutations lead to similar erythroid phenotypes and they were both eliminated from the same animal taxa, suggesting that these 2 proteins may participate in a shared pathway [97].

To obtain more support for the overall physical association of proteins in this cluster, we modeled the 3D structure of the C15orf41-CDAN1 interaction using AlphaFold-Multimer [98], as implemented in Google Colab [99]. The AlphaFold model indicated a high-confidence interface spanning two distinct domains of CDAN1, one contributed from a domain spanning amino acids 1017-1203 and one covering one face of the larger N-terminal domain (2-997) centered on amino acids 427-472 and 843-997; these, in turn, interact with opposing surfaces of C15orf41 (**Figure 4.9**). The predicted structure is consistent with the prior experimental observation that the C-terminal 227 residues of CDAN1 (residues 1000-1227) are critical for the interaction [100]. To investigate the possibility of additional direct interactions between the C15orf41-CDAN1 heterodimer and one or more of the remaining proteins in the cluster, we took advantage of an available X-ray crystal structure that delineated the ASF1A interaction with HIRA residues amino acids 446-466 (PDB entry 2I32) [101] in order to further evaluate a larger complex. Using AF2-multimer, we modeled C15orf41, CDAN1 residues 2-74 and 286-1203 (omitting the

intrinsically disordered segments, as determined by [99]), ASF1A residues 1-155 (omitting the intrinsically disordered tail), and HIRA residues 421-479, a somewhat larger segment known to be critical for the interaction with ASF1A [101]. As illustrated in full in **Figure 4.9**, AlphaFold suggested a binding site for ASF1A distinct from the C15orf41 binding site that, importantly, did not occlude the experimentally determined HIRA binding site, which AlphaFold also recapitulated. Thus, 3D structural modeling confirmed that 4 of the proteins in this cluster can be accommodated within the same overall multiprotein complex.

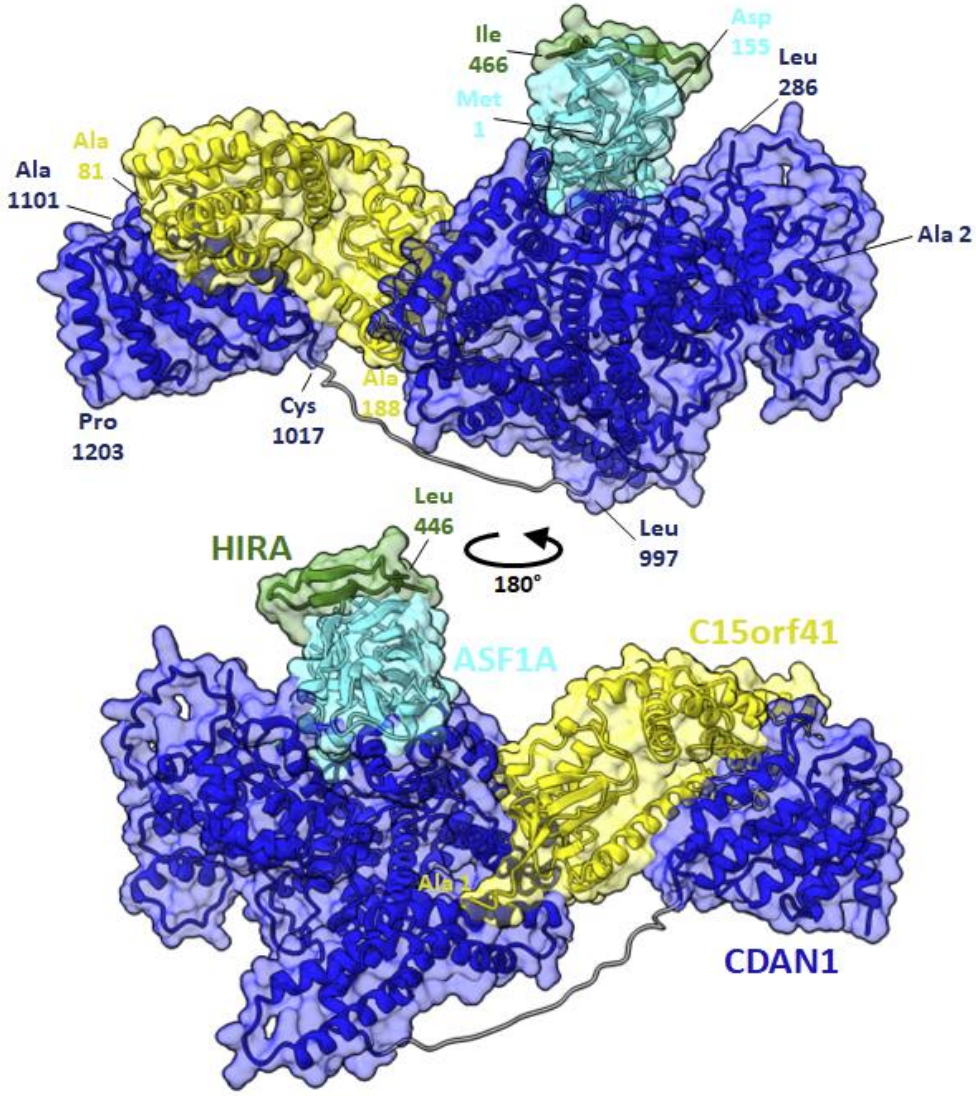


Figure 4.9. Structural modeling supports C15orf41, CDAN1, ASF1A, and HIRA participating in a large multiprotein complex.

Using AlphaFold-multimer, we find that all 4 proteins can be simultaneously accommodated within a single multiprotein complex, here showing C15orf41's modeled interaction with CDAN1 residues 2-74 and 286-1203, ASF1A residues 1-155, and HIRA residues 421-479. For illustration purposes, the known crystal structure of HIRA 446-466 [101] has been superimposed onto the AlphaFold model, which is available in full from the supporting GitHub repository with accompanying quality measurements.

Finally, C11orf42 was found as a subunit in a complex (C11orf42, SNX1, SNX5, VPS29, SNX2, COMMD9) that corresponds to a subcomplex of the Retromer or SNX/BAR complex (e.g. as in [102]), with supporting independent evidence from a learned complex from Super.Complex (C11orf42, SNX1, SNX5, and VPS29). This indicates that C11orf42 may be involved in trafficking with Retromer complex proteins, a notion supported by its localization to intracellular vesicles similar in nature to the other proteins in the complex [34], [103]–[107]. Another example, C16orf91 constituted a complex (C16orf91, UQCC1, COX20, UQCC2) resembling a learned complex from Super.Complex (C16orf91, UQCC1, COX20).

4.5. CONCLUSION

In conclusion, we asked if reinforcement learning could be applied to learn to walk trajectories on a protein interaction network and in this way more accurately determine protein complexes. Application of the method to currently available human protein interaction networks performed competitively with other algorithms, with comparable accuracy but a notable savings in computational time, and in turn led to clear predictions of protein function and interactions for several uncharacterized human proteins. We could support at least one of these, C15orf41, with independent evidence from 3D structural modeling.

4.6. FUTURE WORK

Three main avenues can be explored to potentially improve the RL community detection algorithm, namely, improving the subgraph representations, the RL formulation, and the candidate community search process.

We currently represent a subgraph by a single feature, its density, which gives a problem formulation with a small state space. While performance may be negatively impacted, to improve accuracy, more subgraph features could be included in addition to density. Examples of other subgraph features that could be added include edge weight statistics, node clustering coefficient statistics, and degree correlation statistics. However, as the number of features increases, the state space increases exponentially. For instance, if we incorporate 18 topological features with a discretization of each feature into 10 bins, we increase the number of states to 10^{18}. Different sample-based reinforcement learning methods could be applied to address this challenge and potentially give more accurate results (discussed in section 4.6.2).

The RL formulation could be modified to better accommodate overlapping community detection, by giving a positive reward if a neighbor is present in any of the training complexes, rather than only in the current complex being trained on. Note however that in this scenario, the reward given on choosing a neighbor changes dynamically if adding it in a future step no longer leads to a complex, due to the current subgraph building a different overlapping complex. Due to the dynamic rewards that need to be computed for each neighbor at each iteration, by checking whether the new subgraph is a part of any of the training complexes, computational time would increase significantly compared to the current static reward system, however, it may improve accuracy. Also, different rewards and discount factors can be experimented with in the training phase of the algorithm. Alternate RL frameworks are provided in section 4.6.1.

Finally, in the candidate community search process, there may be scenarios where there are multiple highest-scoring neighbors to add at an iteration in the growth process of a subgraph. Currently, we have only added one of the highest-scoring neighbors to grow

the complex. Each of the other highest-scoring neighbors can be added as well to grow complexes we may have missed in the current method.

4.6.1. Alternative RL formulations for community detection

We can award a reward of 0 or -*c* at each step of the episode, till the end of the episode, when the reward will be the probability of the subgraph being a community. This probability will be equal to 1 when trained on known complexes, or this can be equal to the score from the community fitness function learned in Super.Complex. Using scores from a learned community fitness function yields more data for training when compared to using only the known complexes. If the reward is 0 at all steps except the last, the value function of a state is its probability of leading to a final community.

The current framework can also be modified slightly as follows. Recall that we use negative rewards when the agent picks a neighbor not present in the complex during training. The purpose of the negative rewards is to encourage the RL agent to make minimal mistakes and find the shortest trajectory to the final community. Here, on picking a correct node the next state corresponds to the updated subgraph, and on picking a wrong node, the state does not change. Alternatively, instead of +1, the reward can be the number of complexes the picked node is a part of since we have overlapping known complexes. Note however that this can induce bias into the model, so perhaps it should be the last thing to try to improve results.

4.6.2. Alternate RL methods

Policy iteration can be used to find the value function instead of value iteration. However, both methods may have issues as the state space of the problem increases, which may occur as we incorporate more features. To address this challenge, we can use sample-

based methods, where we sample trajectories from a simulator and use them to estimate the value function in methods such as Monte-Carlo. Other methods that can work without the full trajectories sampled but only known fixed horizon look ahead trajectories include temporal-difference learning (TD) methods such as Q-learning and SARSA. We first plan to test which of these methods- TD and Monte Carlo yields the fastest convergence on the simple case and pick this, let us call it *update rule X*. Following this, we extend to the full problem and observe performance with *update rule X*. Several states may still not be explored and so their value functions will not be learned. So we will implement function approximation with *update rule X* and are interested in exploring deep RL methods, as we can generate a lot of data by sampling a lot of trajectories on the known complexes.

4.6.2.1. Monte Carlo and temporal difference methods

In Monte Carlo methods, the full return (till the end of the episode or infinity) is used and the value function update is given by,

$$V(S_t) \leftarrow V(S_t) + \alpha[\, G_t - V(S_t)] \,. \tag{4.18}$$

In temporal difference methods, we use the return for a number of steps rather than the full return so that the full trajectories are not needed and so that it is faster. The n-step return is given by,

$$G_{t:t+n} \equiv \sum_{i=0}^{n-1} \gamma^i R_{t+i+1} + \gamma^n V_t(S_{t+n}) \,. \tag{4.19}$$

The n-step TD update, *i.e.*, with an n time-step look ahead is given by,

$$V_{t+n}(S_t) \equiv V_{t+n-1}(S_t) + \alpha[\, G_{t:t+n} - V_{t+n-1}(S_t)] \,. \tag{4.20}$$

4.6.2.2. Function approximation

In problems with a large number of states, it is useful to learn a value function that can generalize over unseen states, since it is difficult to visit all the states. The value

function can be represented as a parameterized function $\hat{v}(s, \boldsymbol{w})$ with parameter $w \epsilon \Re^d$ where d is the dimension of the feature vector representing the state. Here, updating the value of one state will indirectly update other states through w. The functional form can be linear, for instance as a weighted sum of the features of the state like in linear regression or a nonlinear function computed by the neural network, where w is the vector of connection weights. The function can therefore be learned using supervised learning methods.

Since data on the true values of states can be sparse, we can learn the value function indirectly, by learning the update to the value function that is performed in an algorithm of our choice such as TD, Monte-Carlo, etc. In other words, the n-step return $G_{t:t+n}$ for a state S_t is learned. In a supervised learning framework, we would construct data where each sample is a feature vector of a state and the label is the return obtained for that state w.r.t to the algorithm we are using, *i.e.*, one-step return, n-step return, or Monte-Carlo. The learned weights correspond to the importance given to each of the features of the state in obtaining a return.

4.7. ACKNOWLEDGMENTS

This research was funded by grants to E.M.M. from the National Institute of General Medical Sciences (R35 GM122480), National Institute of Child Health and Human Development (R01 HD085901), and Welch Foundation (F-1515).

4.8. CODE AND DATA AVAILABILITY

Code, available on GitHub repository:

https://github.com/marcottelab/RL_complex_detection

Data:

All learned complexes from hu.MAP 1.0 with their corresponding scores:

https://marcottelab.github.io/RL_humap_prediction/humap/res_pred_names.txt

All learned complexes from hu.MAP 2.0 with their corresponding scores:

https://marcottelab.github.io/RL_humap_prediction/humap2/res_pred_names_humap2.txt

Interactive visualizations of results on hu.MAP 1.0 and hu.MAP 2.0:

To investigate the complexes identified by the RL algorithm interactively, visualizations are available for the learned complexes on hu.MAP 1.0 and hu.MAP 2.0 on this website:

https://marcottelab.github.io/RL_humap_prediction/

The website also provides the functionality of sorting complexes and proteins by their annotation score and the number of interactions with SARS-CoV-2 proteins.

Structural model for the C15orf41, CDAN1, ASF1A, and HIRA heterotetramer, with sequences and assessments of model quality, can be downloaded from:

https://github.com/marcottelab/RL_humap_prediction/

(folder - CDIN1_CDAN1_2-74_286-1203_ASF1A_1-155_HIRA_421-479_1ModelRelaxed.zip)

4.9. SUPPLEMENTARY FIGURES

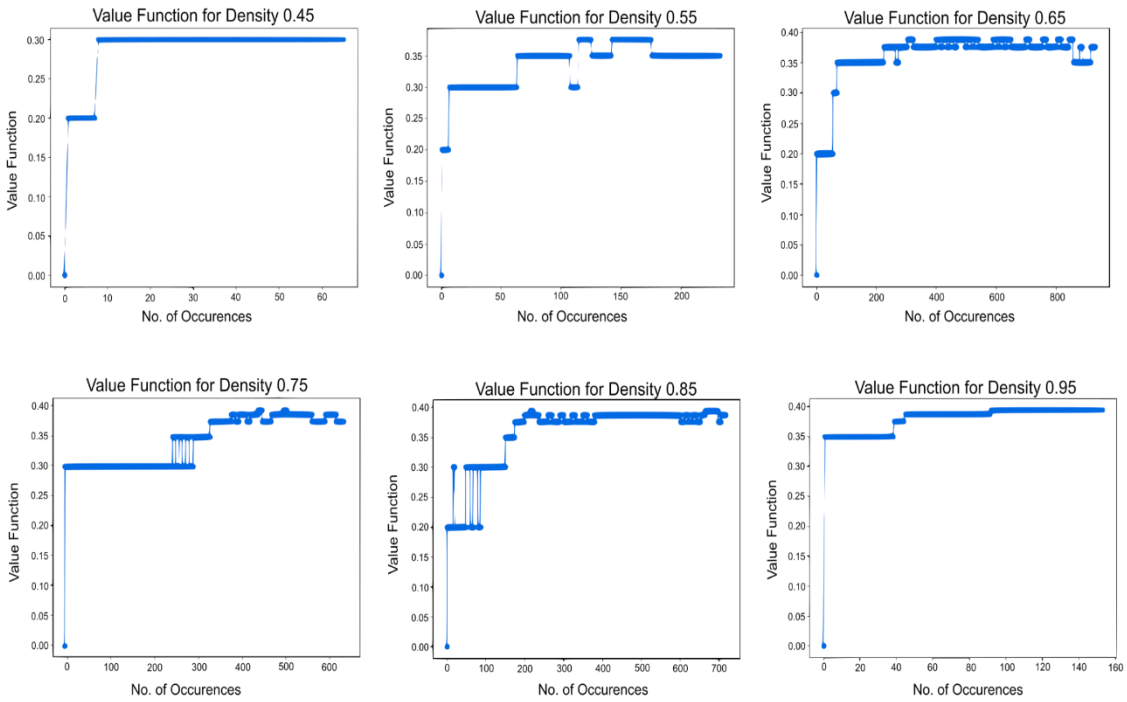


Figure 4.10. Convergence of other scores from the training RL algorithm.

The values of the remaining densities encountered while training on hu.MAP 1.0 also converge eventually.

4.10. SUPPLEMENTARY TABLES

Table 4.4. RL algorithm performance on training and testing toy complexes.

On the synthetic toy network, the RL algorithm predicted 14 toy complexes evaluated against each of the 7 training and 7 testing complexes.

	Evaluation set	FMM Precision	FMM Recall	FMM F-score	UnSPA	Qi et al F1 score	F-weighted K-Clique
RL Algorithm	Training	0.971	0.971	0.971	0.976	1.00	1.00
	Testing	0.956	0.956	0.956	0.961	1.00	1.00

Abbreviations:

- FMM: F-similarity-based Maximal Matching;
- UnSPA: Unbiased Sn-PPV Accuracy;

Table 4.5. RL algorithm performance on training and testing hu.MAP 1.0 complexes.

The RL algorithm predicted 93 complexes with nodes from the training set, evaluated against the 132 training complexes, and 47 complexes with nodes from the testing set, evaluated against the 56 testing complexes. Super.Complex yields 56 and 49 complexes compared with the training and testing sets respectively after removing proteins absent in the known complexes.

	Evaluation set	FMM Precision	FMM Recall	FMM F-score	UnSPA	Qi et al F1 score	F-weighted K-Clique
RL Algorithm	Training	0.639	0.520	0.606	0.788	0.564	1.00
	Testing	0.642	0.538	0.618	0.702	0.552	0.996
Super.Complex (density)	Training	0.877	0.372	0.523	0.776	0.6	1.00
	Testing	0.821	0.704	0.758	0.856	0.779	0.999

Table 4.6. RL algorithm performance on hu.MAP 2.0 complexes.

No.	Method	No. of predicted complexes	Evaluation set	FMM Precision	FMM Recall	FMM F-score	Qi et al F1 score	F-weighted K-Clique
1	**RL Algorithm (hu.MAP 1.0 trained)**	1348	Training	0.523	0.337	0.410	0.326	1.00
			Testing	0.623	0.467	0.534	0.542	1.00
			All	0.55	0.386	0.453	0.396	0.998
2	**Super. Complex (density and hu.MAP 1.0 trained)**	807	Training	0.824	0.374	0.515	0.628	1.00
			Testing	0.559	0.648	0.6	0.626	0.998
			All	0.681	0.449	0.541	0.625	0.999
3	**RL Algorithm (hu.MAP 2.0 trained)**	2319	Training	0.523	0.321	0.398	0.362	1.00
			Testing	0.406	0.239	0.301	0.247	1.00
			All	0.475	0.329	0.388	0.339	0.994
4	**RL Algorithm (Union of 0.1 and 0.02 cutoff , hu.MAP 1.0 trained)**	3614	Training	0.438	0.468	0.452	0.460	1.00
			Testing	0.349	0.493	0.409	0.455	1.00
			All	0.399	0.502	0.445	0.462	0.997
5	**Super. Complex (all features and hu.MAP 2.0 trained)**	582	Training	0.922	0.453	0.608	0.705	1.00
			Testing	0.908	0.696	0.788	0.846	0.998
			All	0.913	0.517	0.66	0.751	0.99
6	**2 stage clustering (ClusterONE + MCL)**	6948	Training	0.379	0.690	0.489	0.784	0.999
			Testing	0.335	0.856	0.482	0.905	0.959
			All	0.359	0.736	0.483	0.807	0.962

Row 1. The RL algorithm uses the value function trained on hu.MAP 1.0 to predict complexes on hu.MAP 2.0 with an edge weight threshold of 0.1. This yields 1348 complexes, of which 85, 42, and 132 are compared with the training, testing, and all sets respectively after removing proteins absent in the known complexes. Parameters used include a Qi overlap threshold of 0.35.
Row 2. Super.Complex (using only feature density) uses the community fitness function trained on hu.MAP 1.0, to predict complexes on hu.MAP 2.0 with an edge weight threshold of 0.1. This yields 807 complexes, of which 56, 48, and 103 are compared with the training, testing, and both sets respectively after removing proteins absent in the known complexes. Parameters used include the pseudo-metropolis search heuristic starting with maximal cliques, merging overlaps with a Jaccard overlap threshold of 0.3.
Row 3. The RL algorithm (hu.MAP 2.0 trained) yields 2319 complexes, of which 81, 33, and 130 are compared with the training, testing, and both sets respectively after removing proteins absent in the known complexes. Parameters used include a Qi overlap threshold of 0.30.
Row 4. The union of the learned complexes by the RL algorithm on hu.MAP 2.0 with an edge weight threshold of 0.1 and 0.02 was learned using the value function trained on hu.MAP 1.0. These are also the hu.MAP 2.0 complexes in the Data availability section. This yields 3614 complexes, of which 150, 53, and 247 are compared with the training, testing, and both sets respectively after removing proteins absent in known complexes. Parameters used include a Qi overlap threshold of 0.325.
Row 5. Super.Complex (all features), trained on hu.MAP 2.0 yields 582 complexes (after merging results from different edge weight thresholds), of which 55, 36, and 90 complexes are compared with the training, testing, and both sets respectively after removing proteins absent in the known complexes. Parameters used include merging overlaps with a Jaccard overlap threshold of 0.1.
Row 6. 2 stage clustering from hu.MAP 2.0 yields 6948 complexes (after taking the union of results from different edge weight thresholds), of which 240, 143, and 385 complexes are compared with the training, testing, and both sets respectively after removing proteins absent in the known complexes.

Chapter 5. DeepSLICEM - Clustering CryoEM particles using deep image and similarity graph representations[4]

In this chapter, we discuss DeepSLICEM, a community detection algorithm, designed as an efficient AutoML pipeline, making use of node image attributes and clustering graph node embeddings, along with attempting classic unsupervised graph clustering strategies. In the previous 2 chapters, we used community detection methods to find protein complexes from protein-interaction networks. We next turn to the problem of determining their structures with the help of cryogenic-electron microscopy. DeepSLICEM is applied to the problem of grouping 2D projections of protein complexes, a necessary step to determine their 3D structures.

5.1. ABSTRACT

Finding the 3D structure of proteins and their complexes has several applications, such as developing vaccines that target viral proteins effectively. Methods such as cryogenic electron microscopy (cryo-EM) have improved in their ability to capture high-resolution images, and when applied to a purified sample containing copies of a macromolecule, produce a high-quality snapshot of different 2D orientations of the macromolecule which can be put together to reconstruct its 3D structure. Instead of purifying a sample so that it contains only one macromolecule, a process that can be difficult, time-taking, and expensive, a cell sample containing multiple particles can be photographed directly and separated into its constituent particles using computational methods.

[4]This chapter is from the paper in prep, Palukuri, M. V., & Marcotte, E. M. (2022), "DeepSLICEM: Clustering CryoEM particles using deep image and similarity graph representations." The algorithm development and experiments were performed by me.

Previous work, SLICEM, has separated 2D projection images of different particles into their respective groups using 2 methods, clustering a graph with edges weighted by pairwise similarities of common lines of the 2D projections. In this work, we develop DeepSLICEM, a pipeline that clusters rich representations of the 2D projections, obtained by combining graphical features from the common lines similarity graph with additional image features extracted from a convolutional neural network. DeepSLICEM explores 6 pre-trained convolutional neural networks and one supervised Siamese CNN for image representation; 10 pre-trained deep graph neural networks for similarity graph node representations and, 4 methods for clustering, along with 8 methods directly clustering the similarity graph. On 6 synthetic and experimental datasets, the DeepSLICEM pipeline finds 92 method combinations achieving better clustering accuracy when compared to previous methods from SLICEM.

5.2. INTRODUCTION

Awarded the Nobel Prize in Chemistry in 2017, cryogenic-electron microscopy, or cryo-EM, a method photographing flash-frozen samples of macromolecules from cellular extracts [108] by bombarding electrons has produced over 20k structures of various macromolecules in the Electron Microscopy Databank (EMDB), many of near-atomic resolution, as of June 2022. While growth in the number of structures solved by cryo-EM has been exponential [109], structures have mostly been determined from studies using single-particle cryo-EM on highly purified samples. Importantly, determining the structures of macromolecules has several applications, as for instance, the cryo-EM structure of the SARS-CoV-2 spike protein [110] helped develop vaccines for COVID-19.

Recently, the cryo-EM technique is being extended to identifying the most abundant particles in more complex cellular extracts [111], [112], rather than on

homogenous purified samples. This approach presents the additional challenge of computationally sorting the images coming from different macromolecules prior to reconstructing their 3D structures. In spite of this challenge, in principle, many more structures can be solved by computationally separating different particles from micrographs of heterogeneous mixtures of macromolecules in a cellular extract (the process visualized in **Figure 5.1**).

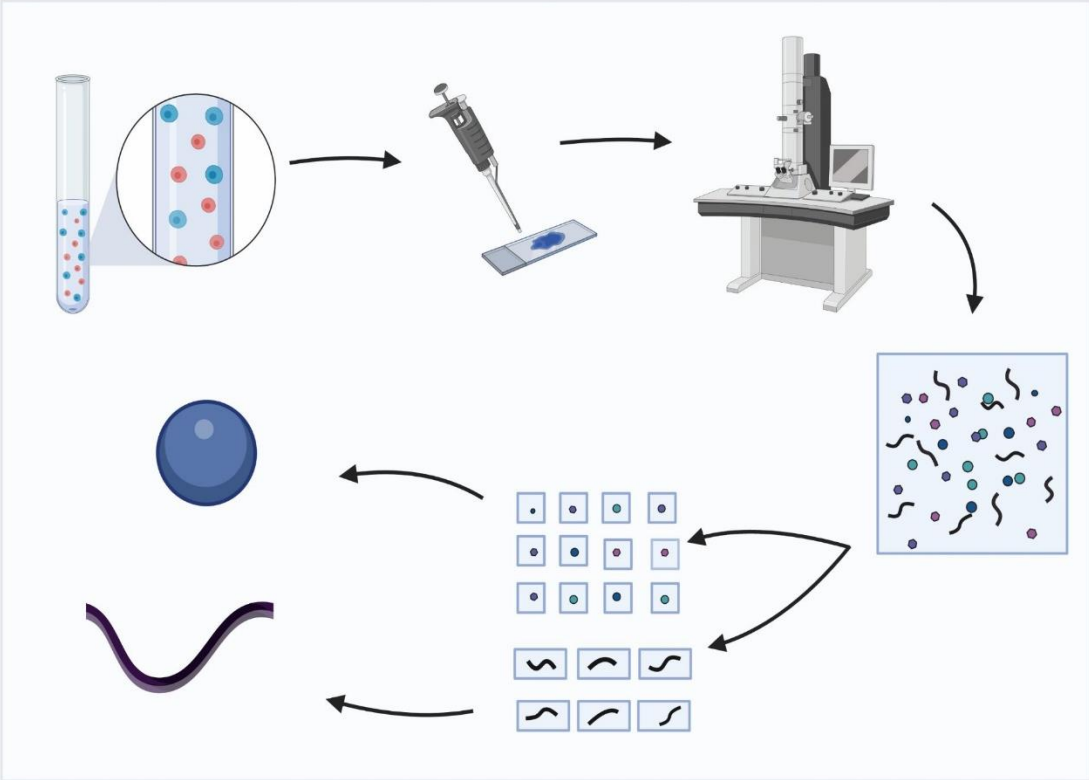


Figure 5.1. Overview of the cryo-EM structure determination process for multiple particles in a mixture.

A sample with multiple proteins is photographed with a cryo-electron microscope. From the resulting micrograph, constituent particles are picked out and separated into different groups of 2D projections of particles. Each group of 2D projections is used to reconstruct each of the 3D structures of the different particles.

Previous methods for analyzing such data compute similarities between projections of particles photographed and cluster the projections into groups corresponding to particles, however, most of them [113]–[115], were demonstrated on solving a similar problem of separating heterogeneous mixtures of different conformations of the same particle. Applied to heterogeneous mixtures of different particles, SLICEM [116] takes advantage of the property of the projection slice theorem in cryo-EM [117]. This theorem states that a common line, *i.e.*, a common 1D line projection exists for any two 2D projections of the same 3D object. Thus, SLICEM used common lines of 2D projections, constructing a graph with similarity of the common lines of two projections as the edge weight; finally clustering the graph to group 2D projections of the same object.

In related work, common-line-based embeddings were learned by a neural network model to represent embeddings of 2D projections from heterogeneous samples, along with a Variational Autoencoder trained to reconstruct 3D objects from X-ray imaging [118]. Given the success of both methods, in this work, we combine representations learned from common lines using graph neural networks and learned image embeddings from pre-trained and supervised CNNs to construct embeddings of 2D projections, which are then clustered into their respective objects. Different applications in computational biology have applied frameworks such as these, for example, clustering was performed on embeddings from a Siamese network trained on must-link and cannot-link sequence similarities to find groups of sequences from the same metagenome [119]. For learning fine-tuned image embeddings in our application, we train Siamese networks with similar and dissimilar projections, learning accurate image embeddings.

In this work, for the task of clustering 2D projections of different particles into their respective groups, we present DeepSLICEM, a pipeline that explores 7 image representation methods with deep neural networks (VGG, ResNet-15, DenseNet, AlexNet,

Efficient-Net-B1 and B7, and Siamese), 10 similarity graph node representation methods with graph neural networks (Node2Vec, Metapath2Vec, GraphWave, Watch Your Step, Attri2Vec, GraphSAGE, Deep Graph Infomax using each of the models - GCN, GAT, APPNP, and Cluster-GCN), 4 clustering methods (Birch, DBSCAN, Affinity Propagation, OPTICS) and 8 graph clustering methods (strongly and weakly connected components, walk trap, edge betweenness, greedy modularity, k-clique, semi-synchronous and asynchronous label propagation). On an experimental dataset and 5 synthetic datasets of varying clustering difficulty (obtained by adding noise to the images, and changing the number of projections and classes of particles), the best methods from DeepSLICEM robustly achieve high clustering accuracy, with the pipeline yielding 92 combinations of methods across the 6 datasets which achieve better clustering accuracy when compared to the previous method from SLICEM (graph clustering with Walktrap or edge betweenness community detection). The improved performance underscores the merit of learning translationally and rotationally invariant image features extracted from deep neural networks and combining them with representations of features from common line-based similarity graphs using graph neural networks. We develop DeepSLICEM as an efficient AutoML pipeline for clustering 2D projections of different objects, available at https://github.com/marcottelab/2D_projection_clustering.

5.3. MATERIALS AND METHODS

Images of 2D projections of different particles are separated into their respective groups by DeepSLICEM. This is achieved by clustering the vector representations of the particles using different clustering algorithms. The SLICEM algorithm is used to construct a similarity graph of the 2D projections based on the similarity between their common lines. The final image representations are found by combining image embeddings of the

images using Siamese neural networks with graph node representations of the images from the similarity graph using graph neural networks.

5.3.1. Experimental datasets

We use the experimental and synthetic datasets from SLICEM [116]. The experimental dataset has a total of 100 2D projections from 4 complexes - 40S, 60S, and 80S ribosomes, and apoferritin. The synthetic dataset has a total of 204 2D projections (2-12 projections per complex) from 35 PDB entries- 1A0I, 1HHO, 1NW9, 1WA5, 3JCK, 5A63, 1A36, 1HNW, 1PJR, 2FFL, 3JCR, 5GJQ, 1AON, 1I6H, 1RYP, 2MYS, 3VKH, 5VOX, 1FA0, 1JLB, 1S5L, 2NN6, 4F3T, 6B3R, 1FPY, 1MUH, 1SXJ, 2SRC, 4V6C, 6D6V, 1GFL, 1NJI, 1TAU, 3JB9 and 5A1A. To better represent real experimental conditions, we additionally construct a synthetic noisy dataset, by adding random noise comprising Gaussian noise (mean and variance of the pixels in the image are used as parameters) to simulate the grainy texture of raw micrographs and salt and pepper noise is added to 1% of the pixels to simulate hot and dead pixels. Two examples of noisy images compared to clean images are given in **Figure 5.2**.

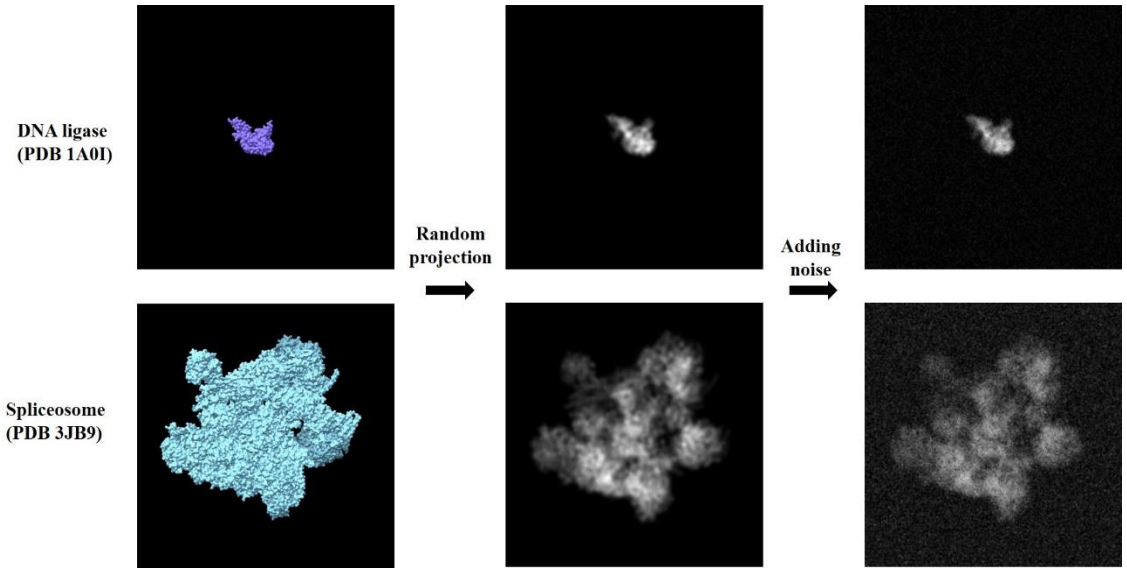


Figure 5.2. Examples of noise added to synthetic projections to construct noisy synthetic images are shown here for random projections of PDB 1A0I (top) - ATP-Dependent DNA Ligase and 3JB9 (bottom) - Spliceosome.

We also investigate the effect of sampling more projections per complex and construct a 'synthetic more projections' dataset, and its corresponding noisy dataset with a total of 558 projections (13-20 projections per complex, randomly sampled from 20-30 uniformly projected, centered 2D images of the 3D structure at a resolution of 9Å per pixel using EMAN1.9's pdb2mrc function). All images are re-scaled to 100x100 pixels after padding smaller images to the largest image size. To examine the potential for bias contributed by the biggest complex, the ribosome, we additionally removed the ribosome to construct a 'synthetic more projections without big ribosome' dataset, and compare this performance to study its impact. For the datasets with more projections, we resize the projections to 100 x 100 pixels from the original 350 x 350 pixels in the synthetic dataset for better memory management. To train a Siamese neural network for the experimental dataset with more samples, we also construct a combined experimental-synthetic dataset, by combining the experimental dataset with the 'synthetic more projections noisy' dataset, after resizing the 2D projections from the experimental dataset to 100 x 100 pixels from their original 96x96 pixels. All images are converted into RGB format before finding or training their embeddings in the algorithm.

5.3.2. Representing projections with image embedding techniques

The 2D projection images are represented in vector space using image embeddings extracted from a convolutional neural network-based method. We explore using 6 different pre-trained neural networks (AlexNet [120], VGG-11 [121], DenseNet [122], ResNet-18 [123], EfficientNet-B1, and EfficientNet-B7 [124]) to extract unsupervised representations of the images using the img2vec python library [125].

5.3.2.1. Siamese neural network embedding

We also implement a supervised learning strategy using a Siamese neural network [126], building on a ResNet-50 (with weights pre-trained on ImageNet [127]) to fine-tune the embeddings such that 2D projections of the same complex are closer in vector space than those from different complexes. This is achieved by learning to estimate the similarity between images, training on triples of images comprising an anchor image (*A*) and another projection of the same complex (positive image *P*), and a projection image of a different complex (negative image *N*). For this, the Siamese neural network comprises 3 identical subnetworks which generate embeddings for each of the images in the triple and compare them using a triplet loss function (*L*) (adapted from the loss function of FaceNet [128]):

$$L(A, P, N) = max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + margin, 0) \quad (5.1)$$

Here, *f* is the image embedding, *i.e.*, the map from an image to its representation in vector space, and *margin* is a hyperparameter that tries to enforce a margin between pairs of similar projections and dissimilar projections, which we define to be 0.5 in our experiments. The two distances for the loss are returned by a custom distance layer we define in the Siamese neural network. We use a batch size of 2 for better memory management. We connect 3 dense layers (512, 256, and 256 units respectively) to the ResNet-50, with the first 2 using a ReLU activation function and batch normalization. All weights of all layers of the ResNet-50 model till layer 'conv5_block1_out' are frozen, while the others are trainable. The model is trained using the Adam Optimizer with a learning rate of 0.0001 using 10 epochs.

To find the most accurate representations of the projections trainable with the current Siamese network architecture and parameters, and to evaluate the accuracy of the subsequent clustering methods alone, we perform an experiment training the Siamese neural network on all the data. For accurate evaluation of the entire pipeline on testing data,

we perform an experiment training the Siamese neural network on a training set of complexes, obtained with a 70-30 split on the complexes into training and testing data (for this experiment we use a batch size of 32). We generate triples for each pair of 2D projections of a complex by sampling 10 negatives randomly from the set of projections of the other complexes. We also experimented by randomly sampling a negative projection image from each of the other complexes for each pair of projections from the same complex. Note that for the dataset combining both the experimental and synthetic image sets, triples are generated separately for each set and then combined. The triples dataset is split into training and validation sets using an 80-20 split. The triples dataset is split into training and validation sets using an 80-20 split. We build 7 Siamese neural networks, corresponding to the 7 datasets, referring to them as Siamese - synthetic, Siamese- synthetic noisy, Siamese - synthetic more projections, Siamese - synthetic more projections noisy, Siamese - more projections without ribosome, Siamese - real, and Siamese - real_synthetic. The additional Siamese neural network built using more negatives for the synthetic dataset is denoted as Siamese - synthetic more negatives.

5.3.3. Incorporating projection similarities using graph node embedding techniques

The similarity graph between the 2D projections is obtained using the SLICEM algorithm [116]. Here the similarity edge weight between two nodes (2D projection images) is the highest similarity measure out of all-by-all pairwise similarities between 1D projections of the 2D projections images projected at different angles, spanning 0 to 180 degrees at 5 degree intervals. We experiment with different similarity measures, the L1 and L2 norms. For the synthetic dataset, we also try the similarity measures - cosine similarity, correlation coefficient, and Wasserstein distance. For each node, the edge weights are updated as the Z-score relative to all edge scores, and the 5 neighbors with the highest edge

weight are chosen to construct a directed similarity graph. For the experimental dataset, we also evaluate a similarity graph with the top 3000 edge weights in the original graph.

The nodes of the similarity graph are represented in vector space using a graph neural network-based node embedding method. We explore using 10 different unsupervised graph node embedding methods, of which 4 methods (Node2Vec [50], Metapath2Vec [129], GraphWave [130], and Watch Your Step [131]) solely represent nodes based on the graph structure, while 6 methods (Attri2Vec [132], GraphSAGE [133], Deep Graph Infomax [134] using each of the models - GCN [45], GAT [135], APPNP [136], and Cluster-GCN [137]) incorporate image embeddings as node attributes and use this information along with the graph structure to represent nodes. All the methods were applied using the python StellarGraph library [138].

5.3.4. Combining image and graph representations of projections

We concatenate the image and graph node embeddings and apply dimensionality reduction using PCA [139] if the vectors are dense or truncated SVD [140] if they are sparse (sparsity > 0.5) and remove dimensions contributing lesser than 2% variance. We also alternatively apply dimensionality reduction to the image and graph node embeddings separately before concatenating them to yield the final embedding vectors.

5.3.5. Clustering the embeddings to separate complexes

The final embeddings for the 2D projections are clustered with 4 different clustering methods (DBSCAN [141], OPTICS [142], BIRCH [143], and Affinity Propagation [144]) using the scikit-learn library [42]. We perform a 70-30 split on the complexes into training and testing data (the same split as was done for the Siamese neural network training) and choose the hyperparameters for each clustering algorithm that give

the best FMMF score [145] on the training data. For the clustering algorithm, we also choose the distance measure out of 22 measures including linear and non-linear metrics (Bray-Curtis, Canberra, Chebyshev, city-block, Pearson correlation, cosine similarity, Dice dissimilarity, Euclidean, Hamming, Jaccard, Jensen-Shannon, Kulsinski, Mahalanobis, matching dissimilarity, Minkowski, Rogers Tanimoto, Russell-Rao, standardized Euclidean, Sokal-Michener, Sokal-Sneath, squared Euclidean, and Yule) that gives the best silhouette score [146] on the training data. The hyperparameter ranges explored are given in **Table 5.1**.

Table 5.1. Hyperparameter ranges explored for the clustering algorithms.

Method	Parameter	Range
DBSCAN	eps	0.25 to 3 in intervals of 0.25
	min_samples	2 to 10 in intervals of 1
OPTICS	max_eps	0.25 to 3 in intervals of 0.25
	min_samples	2 to 10 in intervals of 1
BIRCH	threshold	0.1 to 1 in intervals of 0.1
	branching_factor	10 to 100 in intervals of 10
Affinity Propagation	damping	0.5 to 1 in intervals of 0.1

Apart from clustering the final embeddings, we also directly cluster the similarity graph (represented both as an undirected and directed graph) using different graph clustering techniques, namely, connected components, walk trap [147] (with the number of steps as 4) and edge betweenness [148], using their default parameters in the igraph library [149] and greedy modularity [150], k-clique [151], semi-synchronous [152] and asynchronous [153] label propagation, using their default parameters in the networkx library [41]. The RL community detection method from [154] is also experimented with for the synthetic dataset. All the clustering methods chosen do not need an estimate of the number of clusters making them especially suitable for this application.

5.3.6. Evaluating learned complexes and their representations

The resulting clusters are compared with the ground truth clusters using different evaluation measures including F-similarity-based Maximal Matching F-score (FMMF), Community-wise Maximum F-similarity-based F-score (CMMF), and Unbiased Sn-PPV Accuracy (UnSPA) **(Figure 2.1)** and Qi *et al*. F-score (**Equation 2.7**) [2], F-grand k-clique and F-weighted k-clique [22]. We also compute unsupervised scores of the clustering using the silhouette score, Calinski-Harabasz score [155], and the Davies-Bouldin score [156]. The clustering results reported in the tables are evaluations performed using all the learned complexes, compared with all known complexes. Several 'junk' particle images do not correspond to any complex for the experimental dataset. For this case, we report the evaluation measures, both while comparing the learned complexes with the junk particle projections and after removing the junk particle projections.

We also evaluate the quality of the embeddings for each of the projections with t-SNE [157] plots, using the best distance measure, corresponding to the highest silhouette score on the training data.

5.4. RESULTS AND DISCUSSION

5.4.1. DeepSLICEM improves on SLICEM clustering accuracy of particles in synthetic and experimental datasets

DeepSLICEM trains several clustering methods in a semi-supervised fashion as discussed in the Methods section and selects the best one, achieving better performance on different datasets when compared to the unsupervised SLICEM method (**Table 5.2**). Comparing performance across different datasets (**Table 5.2**, **Figures 5.3** and **5.4**), we observe that both DeepSLICEM and SLICEM achieve better performance on the (i) clean dataset, when compared to the noisy dataset, indicating, as can be expected, that adding noise makes the clustering task harder, and (ii) dataset with more 2D projections when compared to a smaller number of projections, showing that with more 2D orientations available, more pairwise similarity data is available, making the clustering task easier. While SLICEM, like DeepSLICEM, achieves perfect performance on the clean dataset with more projections, its performance degrades more than that of DeepSLICEM on adding noise or decreasing the number of projections. This could be attributed to DeepSLICEM being more robust to translational and rotational variation when compared to SLICEM which is based on the theoretical property of common lines between two 2D projections of the same object. Robustness of both SLICEM and DeepSLICEM to minor variations in the number of clusters is demonstrated by continuing to achieve perfect performance on the synthetic more projections dataset on removing the projections corresponding to one of the clusters, the ribosome.

Table 5.2. DeepSLICEM improves on SLICEM in clustering 2D projections of different particles into their respective clusters.

Dataset	Method	No. of clusters predicted	FMM Precision	FMM Recall	FMM F1 score	Qi *et al.* F1 score	Method details
Synthetic (35 clusters)	Deep SLICEM	41	0.78	0.914	**0.842**	0.842	Reduced Siamese synthetic + reduced Watch Your Step embeddings from top 5 neighbors directed cosine similarity graph, clustered with BIRCH
	SLICEM	20	0.865	0.494	0.629	0.691	Top 5 neighbors directed L2 similarity graph clustered with Walk trap
Synthetic - noisy (35 clusters)	Deep SLICEM	32	0.78	0.713	**0.745**	0.746	Siamese - noisy synthetic + Watch Your Step embeddings from top 5 neighbors directed L2 graph, reduced and clustered with OPTICS
	SLICEM	17	0.786	0.382	0.514	0.5	Top 5 neighbors directed L2 similarity graph clustered with Walk trap
Synthetic more projections (35 clusters)	Deep SLICEM	35	1	1	**1**	1	Reduced Siamese more projections + reduced Watch Your Step embeddings from top 5 neighbors undirected L2 graph clustered with Affinity Propagation
	SLICEM	35	1	1	1	1	Top 5 neighbors directed L2 similarity graph clustered with Walk trap
Synthetic more projections noisy (35 clusters)	Deep SLICEM	32	0.872	0.797	**0.833**	0.925	Reduced Siamese more projections noisy + reduced Node2Vec embeddings from top 5 neighbors directed L2 graph clustered with Affinity Propagation
	SLICEM	23	0.809	0.532	0.642	0.655	Top 5 neighbors directed

							L2 similarity graph clustered with Walk trap
Synthetic more projections w/o ribosome (34 clusters)	Deep SLICEM	34	1	1	**1**	1	Reduced Siamese more projections w/o ribosome + reduced Watch Your Step embeddings from top 5 neighbors undirected L2 graph clustered with Affinity Propagation*
	SLICEM	34	1	1	1	1	Top 5 neighbors L2 similarity graph clustered with Walk trap
Experimental (4 clusters)	Deep SLICEM	4	0.969	0.969	**0.969**	1	EfficientNet-B1 embeddings as node attributes in top 3k edges of undirected L1 similarity graph embedded with APPNP and clustered with Affinity Propagation
	SLICEM[a]	5	0.944	0.944	0.944	1	Top 3k edges of directed L1 similarity graph clustered with Edge betweenness

SLICEM[a] - For the experimental dataset, we obtain better results running SLICEM using top 3k edges when compared to running SLICEM with top 5 nearest neighbors as was done in the original SLICEM paper.

*Using node2vec instead of Watch your step also achieves the same reported metrics. Watch your step is the better method in this case, as it achieves a better Silhouette coefficient, 0.98 than node2vec's 0.88.

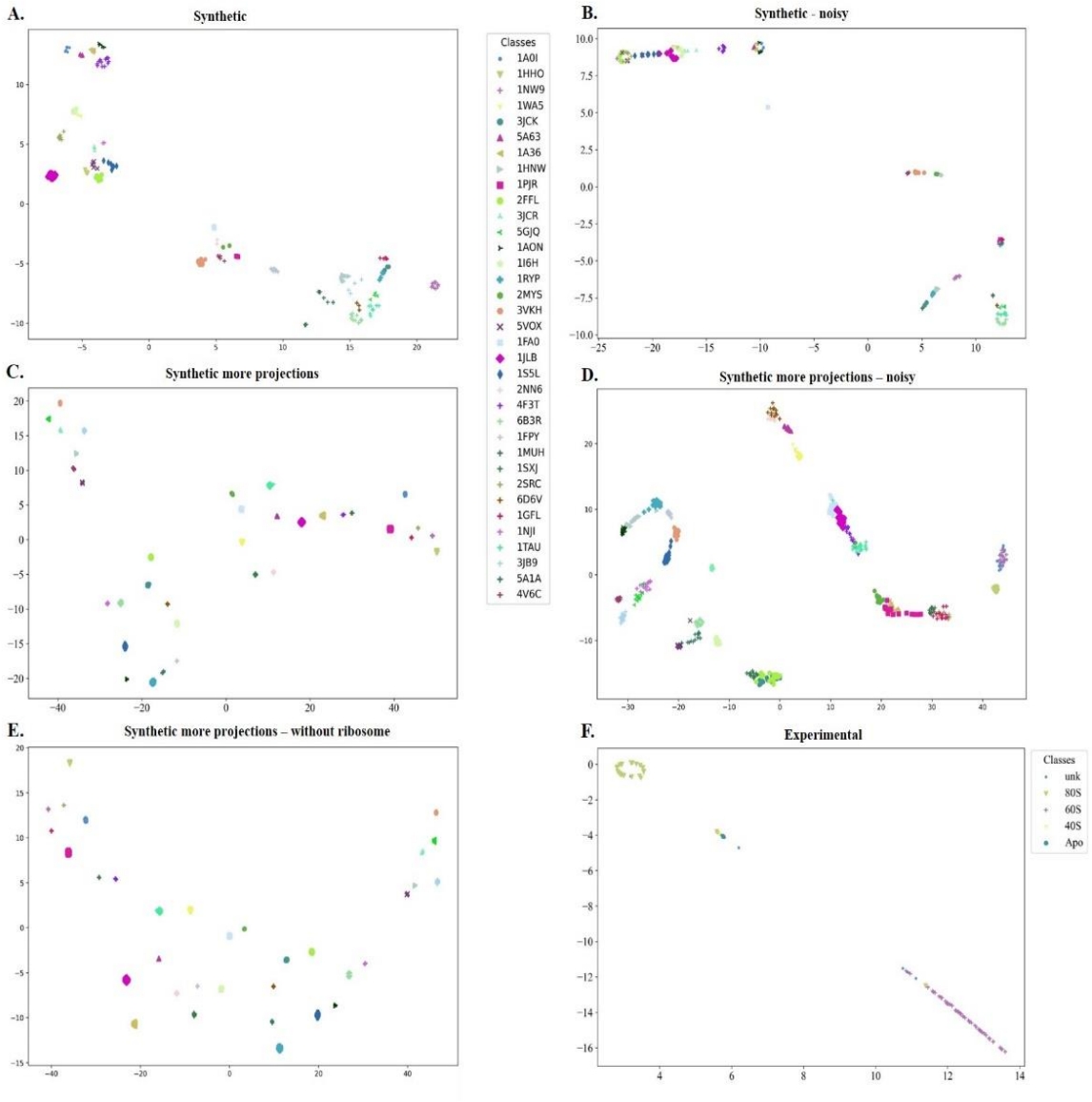


Figure 5.3. t-SNE plots of DeepSLICEM's Siamese embeddings of different datasets (giving the best clustering results).

A. Synthetic B. Synthetic - noisy C. Synthetic more projections D. Synthetic more projections noisy E. Synthetic more projections w/o ribosome F. Experimental

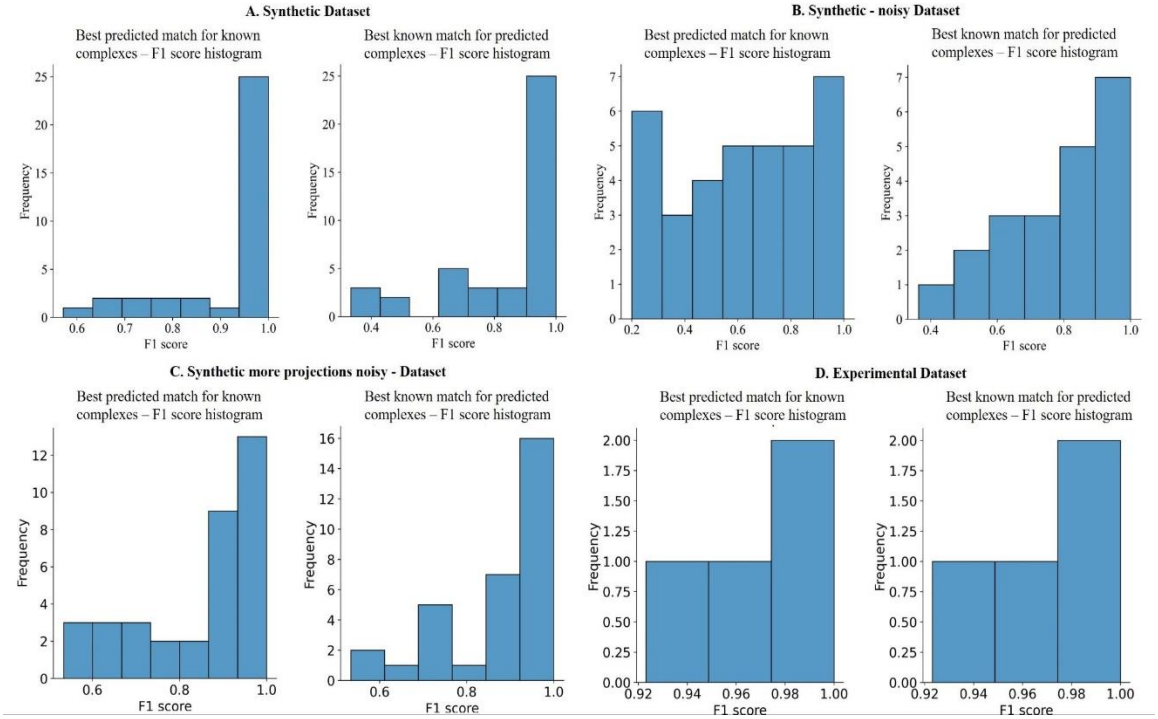


Figure 5.4. F1 score histograms of DeepSLICEM's best clustering on different datasets. A. Synthetic B. Synthetic - noisy C. Synthetic more projections noisy D. Experimental

5.4.2. Clustering combined graph node embeddings and image embeddings outperforms clustering them individually

A case study on the synthetic noisy dataset, the most difficult of the datasets tested, is discussed here. First, we cluster the directed similarity graph directly using different graph clustering methods (**Table 5.3**), finding that Asynchronous label propagation performs the best (0.61 FMMF). We then explore clustering the graph node embeddings in vector space (**Table 5.4**) and achieve better performance than before with Watch Your Step and OPTICS clustering (0.73 FMMF). Next, we cluster image embeddings alone and observe the best performance (0.4 FMMF) with the embeddings from a Siamese neural network trained on the same dataset, which outperforms other unsupervised (pre-trained) embeddings clustering (**Table 5.5**), showing the importance of fine-tuning a neural network to the dataset in question for better performance. We then combine the graph node embeddings with image embeddings and get the best performance (0.75 FMMF) clustering for Siamese - noisy synthetic embeddings concatenated with Watch Your Step, followed by dimensionality reduction and OPTICS clustering. This marginally improves the best results from the graph embedding clustering (0.73 FMMF), showing that the image embeddings add some information to aid the clustering.

Table 5.3. Evaluating similarity graph clustering for the synthetic noisy dataset with different methods.

(L2 norm, 5 nearest neighbors). Asynchronous label propagation on the directed graph outperforms other methods, including walk trap on the directed graph (the algorithm used by SLICEM). The true number of clusters is 35.

Method	Graph type	No. of clusters	FMM Precision	FMM Recall	FMM F1 score	Qi *et al.* F1 score
Asynchronous label propagation	Directed	21	0.815	0.489	**0.611**	0.607
Asynchronous label propagation	Undirected	22	0.749	0.471	0.578	0.491
Label propagation	Undirected	20	0.733	0.419	0.533	0.473
Walktrap (SLICEM)	Directed	17	0.786	0.382	**0.514**	0.500
Greedy modularity	Directed	14	0.788	0.315	0.451	0.408
Greedy modularity	Undirected	13	0.767	0.285	0.416	0.375
k-clique	Undirected	7	0.577	0.115	0.192	0.048

Table 5.4. Evaluating directed similarity graph node embedding clustering for the synthetic noisy dataset. (L2 norm, 5 nearest neighbors).

Node Embedding Method	Clustering Method	Clustering parameters	No. of clusters	FMM Precision	FMM Recall	FMM F1 score	Qi *et al.* F1 score
Watch Your Step	OPTICS	max_eps=2.75, metric='cosine', min_samples=3	31	0.777	0.688	**0.73**	0.788
Node2Vec	OPTICS	max_eps=1.5, metric='cosine'	22	0.776	0.488	**0.6**	0.632
Metapath2Vec	OPTICS	max_eps=1.5, metric='cosine'	18	0.744	0.383	0.505	0.491
GraphWave	Affinity Propagation	damping=0.9	17	0.408	0.198	0.267	0.077

Table 5.5. Evaluating different image embedding methods followed by clustering for the synthetic noisy datasets.

The best clustering method for each of the embeddings is reported here. The Siamese network trained on the noisy synthetic data outperforms other methods by a significant margin, demonstrating the improvement achieved by fine-tuning embeddings to the dataset being evaluated.

Embedding Method	Clustering Method	Clustering parameters	FMM Precision	FMM Recall	FMM F1 score	CMMF
Siamese - noisy synthetic	BIRCH	branching_factor=90 threshold=0.2	0.576	0.313	**0.405**	0.511
Siamese - synthetic	BIRCH	branching_factor=20 threshold=0.4	0.088	0.280	0.134	0.136
ResNet-18	BIRCH	branching_factor=10 threshold=0.9	0.077	0.322	0.124	0.124
Siamese - synthetic more projections	BIRCH	branching_factor=10 threshold=0.1	0.081	0.244	0.121	0.125
EfficientNet-B1	BIRCH	branching_factor=90 threshold=0.1	0.074	0.315	0.120	0.120
DenseNet	BIRCH	branching_factor=10 threshold=0.9	0.066	0.348	0.111	0.111
AlexNet	Affinity Propagation	damping=0.5	0.121	0.055	0.076	0.112
EfficientNet-B7	Affinity Propagation	damping=0.9	0.110	0.057	0.075	0.109
VGG	Affinity Propagation	damping=0.6	0.112	0.054	0.073	0.110

Table 5.6. Evaluating different image embedding methods concatenated with graph node embedding methods followed by dimensionality reduction and clustering for the synthetic noisy datasets.

The results presented are those giving better performance than SLICEM.

Image embedding Method	Node Embedding Method	Clustering Method	Clustering parameters	No. of clusters	FMM Precision	FMM Recall	FMM F1 score
Siamese - noisy synthetic	Watch Your Step	OPTICS	max_eps=0.5, metric='cosine', min_samples=3	32	0.780	0.713	**0.745**
Siamese - synthetic more projections	Watch Your Step	OPTICS	max_eps=2.75, metric='cosine', min_samples=3	32	0.766	0.700	0.731
Siamese - noisy synthetic	Node2Vec	DBSCAN	metric='sqeuclidean', min_samples=3	22	0.857	0.539	0.662
Siamese - synthetic more projections	Node2Vec	OPTICS	max_eps=0.25, metric='cosine', min_samples=4	23	0.771	0.507	0.612
Siamese - synthetic	Watch Your Step	OPTICS	max_eps=0.5, metric='cosine'	21	0.788	0.473	0.591
Siamese - noisy synthetic	GraphWave	OPTICS	max_eps=2.25, metric='sqeuclidean', min_samples=3	30	0.605	0.518	0.558
Siamese - synthetic	Node2Vec	Affinity Propagation	damping=0.9	19	0.788	0.428	0.555

5.4.3. Multiple methods in DeepSLICEM achieve better clustering than SLICEM's Walktrap graph clustering

On the synthetic noisy dataset, compared to SLICEM (walk trap clustering on L2 similarity graph) which achieves 0.51 FMMF, 30 methods attempted by DeepSLICEM give better performance, *i.e.*, 3 graph clustering methods - asynchronous and synchronous label propagation on the directed graph, and synchronous label propagation on the undirected graph (**Table 5.3**), 2 graph node embeddings' clustering in vector space - Watch Your Step and Node2Vec embeddings clustered by the OPTICS algorithm (**Table 5.4**), 7 methods clustering reduced dimension vectors from image embeddings concatenated with graph node embeddings (**Table 5.6**), 13 methods clustering reduced image embeddings concatenated with reduced graph node embeddings (Supplementary **Table 5.7**) and 5 methods clustering graph node embeddings with image embeddings incorporated as node attributes (Supplementary **Table 5.8**).

On the synthetic more projections noisy dataset, compared to SLICEM (walk trap clustering on L2 similarity graph) which achieves 0.64 FMMF, 31 methods attempted by DeepSLICEM give better performance, *i.e.*, 3 graph clustering methods - on the undirected graph, asynchronous label propagation (achieving best performance with 0.667 FMMF) and synchronous label propagation, and synchronous label propagation on the directed graph (Supplementary **Table 5.9**), 13 methods clustering reduced dimension vectors from image embeddings concatenated with graph node embeddings (Supplementary **Table 5.9**) (6 on the directed graph and 7 on the undirected graph) and 15 methods clustering reduced image embeddings concatenated with reduced graph node embeddings (Supplementary **Table 5.9**) (8 on the directed graph and 7 on the undirected graph).

On the synthetic dataset, compared to SLICEM (walk trap clustering on L2 similarity graph) which achieves 0.63 FMMF, 27 methods attempted by DeepSLICEM give better performance, *i.e.*, 5 graph clustering methods - asynchronous label propagation on the directed cosine similarity graph (achieving best performance with 0.83 FMMF) and L2 similarity graph, and undirected L2 similarity graph, and synchronous label propagation on the undirected cosine similarity graph (Table S4), 4 methods clustering reduced dimension vectors from image embeddings concatenated with graph node embeddings (Supplementary **Table 5.10**), 10 methods clustering reduced image embeddings concatenated with reduced graph node embeddings (Supplementary **Table 5.10**) and 8 methods clustering graph node embeddings with image embeddings incorporated as node attributes (Supplementary **Table 5.10**).

On the synthetic-more-projections dataset, one of the attempted methods by DeepSLICEM (Reduced Siamese more projections + reduced Watch Your Step embeddings from top 5 neighbors undirected L2 graph clustered with Affinity Propagation), reported in **Table 5.2** outperforms SLICEM, while two methods from DeepSLICEM (Reduced Siamese more projections w/o ribosome + reduced Watch Your Step or Node2vec embeddings from top 5 neighbors undirected L2 graph clustered with Affinity Propagation) reported in **Table 5.2** achieve better performance than SLICEM on the synthetic more projections dataset without the ribosome. On the experimental dataset, with the top 3k edges of the undirected L1 similarity graph, SLICEM (walk trap clustering) is outperformed by one method attempted by DeepSLICEM (**Table 5.2**) - EfficientNet-B1 embeddings as node attributes in the graph embedded with APPNP, clustered with Affinity Propagation.

Given the plethora of methods explored by DeepSLICEM, we investigated which methods were achieving good performance consistently across datasets out of the total 92

experiments where DeepSLICEM achieves better performance than SLICEM. Of 81 experiments clustering vector representations, DeepSLICEM finds that the clustering method chosen as the best is OPTICS, followed by Affinity propagation, Birch, and DBSCAN (**Figure 5.5A**). Next, we examine which node embedding methods performed well. Of 81 experiments clustering node embeddings, Watch Your Step was chosen as the best 49% of the time, followed by node2vec (28%) and other methods each taking up <6% of wins (**Figure 5.5B**). We then examine which image embedding methods were chosen as the best (in conjunction with a node embedding method) in 74 experiments outperforming SLICEM, we observe that Siamese embeddings win most of the time compared to unsupervised methods, of which EfficientNets win most of the time (**Figure 5.5C**). From 11 experiments of different graph clustering methods, we find that label propagation methods on directed and undirected graphs perform well (**Figure 5.5D**) with asynchronous label propagation on both types of graphs dominating accuracy as was observed earlier. From **Table 5.2**, we observe that the best-performing methods for different datasets comprise predominantly Siamese embeddings, followed by EfficientNet-B1 embeddings for image representation; Watch your step, followed by node2vec for graph node embeddings, and Affinity Propagation, followed by OPTICS and Birch for clustering - consistent with their dominance in the total 92 combinations of methods performing better than SLICEM.

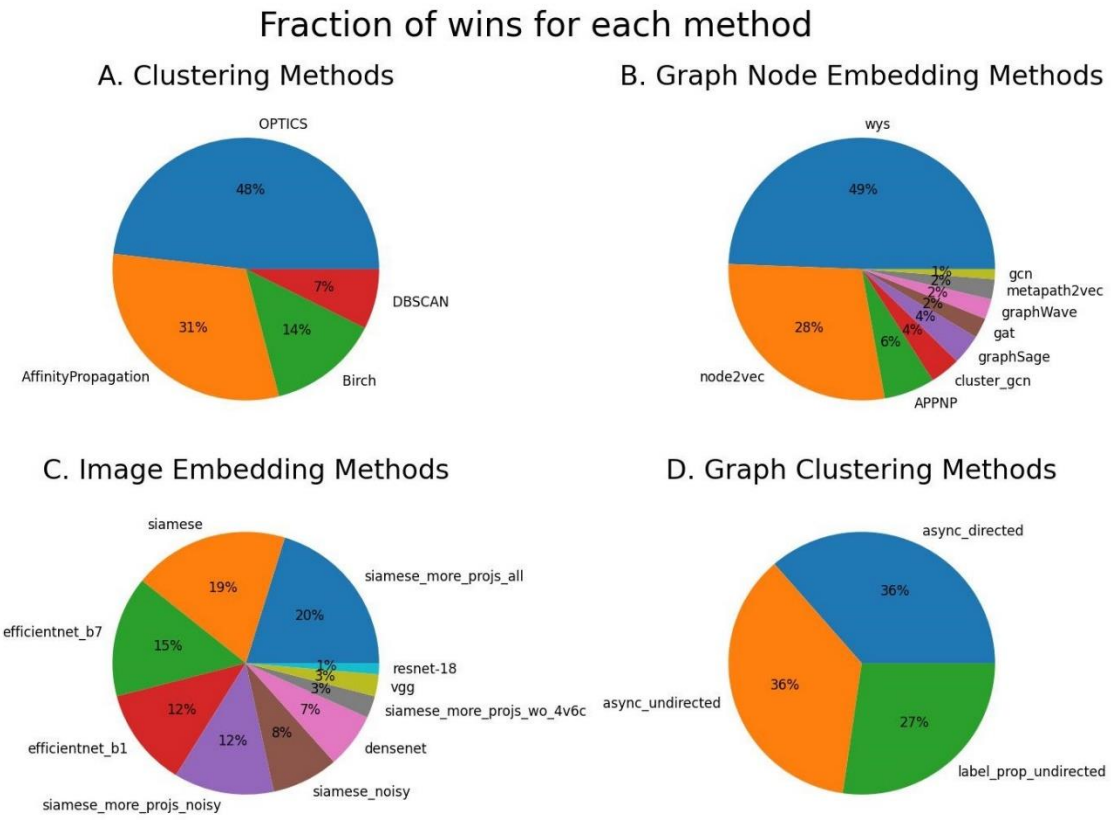


Figure 5.5. The best methods chosen by DeepSLICEM.

A. Clustering methods B. Graph node embedding methods C. Image embedding methods D. Graph clustering methods

5.5. CONCLUSIONS AND FUTURE WORK

Accurately separating a mixture of 2D projections of different macromolecules computationally will help reduce the need for experimental purification processes, which may also lead to the loss of some groups of molecules. Previous work, SLICEM, has explored using similarity between common lines of projections to build a pairwise similarity graph, which was then clustered using 2 methods. In this work, we develop DeepSLICEM, a pipeline that evaluates 10 graph neural networks for representing the similarity graph, combined with 7 different CNNs that add additional features of the images to the common line similarity graph, and clustering with 13 different methods. Across 6 datasets from synthetic and experimental macromolecules, DeepSLICEM finds 91 methods (combinations of image embedding methods, graph node embedding methods, and clustering methods) that achieve better clustering accuracy than SLICEM. The best methods dominating performance in DeepSLICEM are combinations of Watch your step graph node embeddings, clustering with Affinity Propagation, and Siamese neural network image embeddings, the latter demonstrating how supervision can help improve performance achieved with the unsupervised method, SLICEM.

Different future directions can be explored to improve performance. The current image embeddings used are from CNNs pre-trained on images from ImageNet. Since ImageNet has millions of images, on average having ~1000 images per synonym set, with objects diversely represented, *i.e.*, in different poses, from different viewpoints, having background noise, with occlusions and localized to different parts of the images, the CNNs learn rotational and translational invariance of the object in question, making it suitable to the current problem of learning similar embeddings for different 2D projections of the same 3D object. However, ImageNet does not have any images of micrographs, so the weights of the CNNs can be re-trained to learn Cryo-EM images' features as demonstrated with the

Siamese neural network re-training a ResNet-50. This can be tried with some of the other CNNs performing well such as the EfficienNet-B1. Alternately autoencoders such as CryoDRGN [158] can be trained on large datasets of molecules from CryoEM images to learn good image embeddings that can be used for clustering. Perspective transformer nets [159] have been proposed to transform 2D projections of objects into their 3D representations. The encoder part of their neural network can be fine-tuned for our problem and be used to generate image embeddings.

Considering the success of DeepSLICEM in clustering projections of macromolecules from Cryo-EM, this AutoML pipeline can also be explored in application to any problem where different perspectives of multiple objects are to be grouped.

5.6. CODE AND DATA AVAILABILITY

The software developed in this work is made publicly available on GitHub at https://github.com/marcottelab/2D_projection_clustering.

5.7. ACKNOWLEDGMENTS

The authors are grateful to Eric Verbeke for helpful discussion and assistance with SLICEM and Caitie McCafferty for her valuable insights, assistance with EMAN, and contribution of code for expanding the synthetic dataset. We also thank Kalyani Palukuri for making **Figure 5.1** of the paper. This research was funded by grants to E.M.M. from the National Institute of General Medical Sciences (R35 GM122480), National Institute of Child Health and Human Development (R01 HD085901), and Welch Foundation (F-1515).

5.8. SUPPLEMENTARY TABLES

Table 5.7. Evaluating different reduced image embedding methods concatenated with reduced graph node embedding methods and clustering for the synthetic noisy datasets.

The results presented are those giving better performance than SLICEM.

Image embedding Method	Node Embedding Method	Clustering Method	Clustering parameters	No. of clusters	FMM Precision	FMM Recall	FMM F1 score
Siamese - noisy synthetic	Watch Your Step	OPTICS	max_eps=0.25, metric='cosine', min_samples=3	31	0.79	0.7	0.742
Siamese - synthetic more projections	Watch Your Step	Birch	n_clusters=None, branching_factor =50, threshold=0.8	27	0.759	0.585	0.661
Siamese - noisy synthetic	Node2Vec	OPTICS	max_eps=2.75, metric='sqeuclidean', min_samples=3	34	0.639	0.621	0.630
Siamese - synthetic more projections	Node2Vec	OPTICS	max_eps=0.25, metric='cosine'	22	0.773	0.486	0.596
EfficientNet -B7	Watch Your Step	Affinity Propagation	damping=0.9	22	0.755	0.474	0.583
VGG	Watch Your Step	OPTICS	max_eps=2.5, metric='braycurtis', min_samples=3	22	0.747	0.469	0.576
EfficientNet -B7	Node2Vec	Affinity Propagation	damping=0.9	23	0.718	0.472	0.569
Siamese - noisy synthetic	Metapath2Vec	BIRCH	branching_factor =30, n_clusters=None	19	0.804	0.437	0.566
Siamese - synthetic	Node2Vec	Affinity Propagation	damping=0.9	19	0.796	0.432	0.560

VGG	Node2Vec	Affinity Propagation	damping=0.9	23	0.674	0.443	0.534
DenseNet	Watch Your Step	OPTICS	max_eps=2.5, metric='braycurtis', min_samples=4	20	0.732	0.419	0.533
Siamese - synthetic	Watch Your Step	OPTICS	max_eps=0.25, metric='correlation', min_samples=3	24	0.649	0.445	0.528
EfficientNet -B1	Watch Your Step	OPTICS	max_eps=0.5, metric='braycurtis', min_samples=4	19	0.742	0.403	0.522

Table 5.8. Evaluating different methods of clustering graph node embedding methods with image embeddings as node attributes for the synthetic noisy datasets.

The results presented are those giving better performance than SLICEM.

Image embedding Method	Node Embedding Method	Clustering Method	Clustering parameters	No. of clusters	FMM Precision	FMM Recall	FMM F1 score
EfficientNet-B7	APPNP	BIRCH	branching_factor=80, threshold=0.3	34	0.602	0.584	0.593
EfficientNet-B7	Cluster-GCN	BIRCH	branching_factor=80	34	0.579	0.563	0.571
EfficientNet-B1	Cluster-GCN	BIRCH	branching_factor=20	27	0.622	0.480	0.542
EfficientNet-B7	GraphSage	BIRCH	branching_factor=70, threshold=0.2	39	0.505	0.563	0.532
EfficientNet-B1	APPNP	BIRCH	branching_factor=60, n_clusters=None, threshold=0.2	55	0.433	0.681	0.529

Table 5.9. Supplement/ synthetic_more_projs_noisy_all_results_compiled_better_than_slicem.csv

Table 5.10. Supplement/ synthetic_all_results_compiled_better_than_slicem.csv

Supplement: https://drive.google.com/drive/folders/1Q9sI_026HnT_9j-PAmWhG47w-c61bnZN?usp=sharing/

Chapter 6. Conclusions and looking ahead

In this dissertation, I discussed 3 community detection methods that learn from known communities and use this knowledge to find new communities. The information learned is accurately represented as a community fitness function in the case of Super.Complex, a value function in the case of RL complex detection, and community-informed embeddings in the case of DeepSLICEM with the help of the Siamese neural network. Further, considering the shortcomings of existing evaluation measures, I developed 3 measures to evaluate the performance of community detection algorithms. With growing data, and large network size and information, efficient methods are of importance, therefore, we develop distributed and fast algorithms. While our methods are fairly generalizable, we provide a framework to incorporate domain-specific information in terms of graph node attributes and demonstrate success in the application of detecting protein complexes by applying community detection to protein-interaction networks and via structure determination by grouping 2D projection images of different protein complexes. Note that community detection is an NP-hard problem, hence heuristic-based methods have been used to solve this problem in Super.Complex with guarantees such as internal connectivity (which some global community detection methods such as Louvain fail at), and we learn to learn those heuristics in RL complex detection, demonstrating a problem framework satisfying the Markov property, opening up community detection to the field of reinforcement learning. We apply Super.Complex and RL complex detection to two human protein interaction networks to yield 1000+ protein complexes each, including protein complexes with uncharacterized proteins, suggesting their potential functions via their associations with known proteins. We also predict complexes potentially linked to SARS-CoV-2, the virus responsible for COVID-19, and the pandemic that started in the middle of my Ph.D. We make our code and results easily accessible and build

interactive websites with visualizations for biologists to look up complex memberships for proteins of interest and find out more information about them with hyperlinks to protein databases.

The work in this dissertation has several original aspects, a few of which are summarized here. Super.Complex has a cross-validation pipeline to select the heuristic and the parameters that work best for the application at hand. Minimal hyper-parameter selection is required in our algorithm with default parameters provided when smart hyperparameters cannot be inferred, reducing the need for extensive parameter sweeps. To our knowledge, greedy heuristics in conjunction with other heuristics such as iterative simulated annealing have not been applied in the past for community detection. Along with these heuristics, Super.Complex implements an original distributed architecture and software design with an original graph storage method. As for guarantees, apart from internal connectivity of communities, Super.Complex's merging algorithm ensures that no two communities can be merged to yield a higher scoring community if such a set of communities is desired.

Our methods are well suited for applications having a limited amount of known communities, since, during splitting known complexes into train and test sets, unlike previous methods which discarded complexes to achieve independent train and test sets, we emphasize the preservation of known complexes in our train-test splits while ensuring 70-30 splits, independence, and similar size distributions. Similarly, to maintain a high number of known complexes and predicted protein complexes, a minimal number of merges is attempted in the merging algorithms devised. Further, our methods are translatable to domains with limited/no knowledge by transferring models and community functions. While known community information may be available, information about non-communities may be harder to obtain. The reinforcement learning methods proposed have

the advantage of not needing non-existent negatives required for binary supervised classification in supervised community detection methods.

6.1. FUTURE DIRECTIONS

With a growing body of information about nodes in a network in different applications, it becomes imperative to move from dealing with simple weighted networks, where only edge weights carry information that is used in community detection to working directly with information-rich networks, in terms of node and edge attributes. Methods that effectively combine information from node attributes and graph structure are therefore of importance, a theme that was explored in DeepSLICEM.

With the advent of large natural language processing models in machine learning making great strides in language understanding, such as GPT being able to create stories given a small prompt, and achieving a deep understanding of language; the hidden language of protein sequences can be uncovered with the application of such methods. Transformer models learn to predict words by masking words in sentences, and models have been trained such as BERT [160] to represent English words and sentences in vector space for various downstream applications. These ideas have also been adopted in computational biology, for instance, ProtBERT [46] was trained to represent protein sequences. We hypothesize that incorporating sequence information as node attributes and relearning the community fitness function will help the model learn information about proteins contributing to their assembly into complexes, that is not currently captured using the edge weights based on mass spectrometry experiments, for instance, taking into account losses experienced from going in vivo to in vitro, a problem especially observed with membrane proteins. The community fitness function can be used as before in Super.Complex for local community detection. Alternately, ProtBERT can be connected

to the graph neural network so that an end-to-end fully supervised setting is provided, to tweak the vectors representing the sequences such that they carry information most relevant for the task at hand, protein complex detection.

Using a function approximation method for reinforcement learning for community detection will allow learning values for a large state space. With such a method it would be possible to incorporate rich graph representations including node attributes such as sequences, to build a deep reinforcement learning method for community detection with the subgraph directly as input, extracting a subgraph representation from the graph structure along with node and edge attributes.

Learning to walk trajectories on a network to achieve the final goal of learning a community with reinforcement learning, *i.e.*, maximizing the objective function of the community fitness function, when considered in the light of learning parameters for heuristics adopted, talks to the overarching theme of learning to learn parameters efficiently, rather than relying on perturbations, which may not achieve optima in non-convex settings. Essentially, this translates to using reinforcement learning to traverse the loss function manifold, a promising research direction for the field of machine learning towards more efficient training of models.

For small networks that fit in memory, or cases where a graph is partitioned into portions that can be treated as independent graphs, a global setting for community detection is efficient and by representing graph nodes in vector space, allows for combining node attribute vectors, turning it into a clustering problem in vector space, for example, clustering graph node embeddings in applications such as the 2D projection clustering on which DeepSLICEM was applied. The clustering can be made more efficient using data structures such as faiss [161] or k-d trees to make the computation of pairwise distances faster. Alternately, instead of working in vector space, one can traverse the 2D projection

similarity graph with heuristics or reinforcement learning using subgraph embeddings constructed from the combined image and graph node embeddings, either in an unsupervised fashion with methods such as subgraph2vec and graph2vec, or in a supervised manner, extracted from a graph neural network learning a community fitness function. Moving from multi-step combinations to a streamlined end-to-end supervised model would make feature extraction more task-specific. For instance, as in the case of tweaking ProtBERT's weights for the task of community detection, an end-to-end supervised model or deep reinforcement learning model can be implemented for the 2D projection clustering application, tweaking the embedding layer weights in the (image and graph) neural networks to produce features most relevant for community detection.

Appendix A. A computational framework for studying the gut-brain axis in Autism Spectrum Disorder[5]

A.1. ABSTRACT

A.1.1. Introduction

The integrity of the intestinal epithelium is crucial for human health and is harmed in autism spectrum disorder (ASD). An aberrant gut microbial composition resulting in gut-derived metabolic toxins was found to damage the intestinal epithelium, jeopardizing tissue integrity. These toxins further reach the brain *via* the gut-brain axis, disrupting the normal function of the brain. A mechanistic understanding of metabolic disturbances in the brain and gut is essential to design effective therapeutics and early intervention to block disease progression. Herein, we present a novel computational framework integrating constraint-based tissue-specific metabolic (CBM) model and whole-body physiological pharmacokinetics (PBPK) modeling for ASD. Furthermore, the role of gut microbiota, diet, and oxidative stress is analyzed in ASD.

A.1.2. Methods

A representative gut model capturing host-bacteria and bacteria-bacteria interaction was developed using CBM techniques and patient data. Simultaneously, a PBPK model of toxin metabolism was assembled, incorporating multi-scale metabolic information. Furthermore, dynamic flux balance analysis was performed to integrate CBM and PBPK. The effectiveness of a probiotic and dietary intervention to improve autism symptoms was tested on the integrated model.

[5]This chapter is reproduced from the paper Mohammad FK, Palukuri MV, Shivakumar S, Rengaswamy R and Sahoo S (2022) "A Computational Framework for Studying Gut-Brain Axis in Autism Spectrum Disorder". Front. Physiol. 13:760753. https://doi.org/10.3389/fphys.2022.760753. Along with FM and SSh, I built the models, analyzed the data, and interpreted the results. Along with FM, I developed the algorithms and analysis tools.

A.1.3. Results

The model accurately highlighted critical metabolic pathways of the gut and brain that are associated with ASD. These include central carbon, nucleotide, and vitamin metabolism in the host gut, and mitochondrial energy and amino acid metabolisms in the brain. The proposed dietary intervention revealed that a high-fiber diet is more effective than a western diet in reducing toxins produced inside the gut. The addition of probiotic bacteria *Lactobacillus acidophilus*, *Bifidobacterium longum*, *Akkermansia muciniphila*, and *Prevotella ruminicola* to the diet restores gut microbiota balance, thereby lowering oxidative stress in the gut and brain.

A.1.4. Conclusion

The proposed computational framework is novel in its applicability, as demonstrated by the determination of the whole-body distribution of ROS toxins and metabolic association in ASD. In addition, it emphasized the potential for developing novel therapeutic strategies to alleviate autism symptoms. Notably, the presented integrated model validates the importance of combining PBPK modeling with COBRA-specific tissue details for understanding disease pathogenesis.

A.2. INTRODUCTION

Autism spectrum disorder (ASD) is a complex neurodevelopment disorder. Social impairment, reduced cognitive capability, communication deficits, and stereotyped body behavior are typical clinical characteristics of ASD [162]. Although genetic and environmental factors have traditionally defined the development of ASD [163]–[165], recent studies claim that this is only present in a minority of cases, and that there has been more emphasis on the importance of inherent metabolic disturbances [162], [166]. In many cases, comorbidity patterns describe the pathogenesis of autism, most notably

gastrointestinal defects and abnormal gut microbiome composition [166]–[170]. These factors can be transmitted to the brain *via* the gut-brain axis, resulting in neuronal dysfunction [171].

Home to a diverse and dynamic bacterial population, the human gastrointestinal tract is influenced by diseases and homeostasis [172]. A variety of factors affect the interactions between gut microbiota and the host. One of the most crucial factors of host intestinal epithelial dysfunction is gut microbial disequilibrium (dysbiosis), which contributes to increased toxin production within the Gut [173]. Toxins produced by dysbiosis, such as propionic acid [174], lipopolysaccharides [175], and most importantly, reactive oxygen species (ROS; [168]), stimulate the production of inflammatory cytokines. As a result, it increases permeability in the gut, allowing toxins into the bloodstream where they can cross the blood-brain barrier and cause symptoms of autism [176], [177]. Therefore, it reinforces the concept that any intervention successfully treating gastrointestinal dysfunction may alleviate ASD-related brain symptoms.

Investigating bacterial composition in patients with ASD reveals dysbiosis features, such as the increased abundance of *Clostridium*, *Bacteroides*, *Desulfovibrio*, *Ruminococcus*, and *Shigella* [171], [178]–[180] followed by reduced levels of *Bifidobacterium*, *Lactobacillus*, *Prevotella*, and *Akkermansia* [180]–[183]. Intestinal epithelium cells absorb nutrients and play an essential role as a first-line defense against microbial dysbiosis. Elevated levels of reactive oxygen species (ROS), *i.e.*, hydrogen peroxide (H_2O_2) and superoxide (O_2^{-1}), is reported in the autistic gut as a cause of epithelial tissue damage [168], [184]–[186]. Increased gut permeability, thereby, allows these ROS toxins to be transported into the bloodstream and can further breach the blood-brain barrier, affecting the normal function of the brain. These findings, therefore, link the gut microbiome with ASD *via* the microbe-gut-brain axis and strongly support the leaky-gut

hypothesis. The leaky gut or intestinal hyper-permeability is the widening of tight junctions in the gut wall, leading to gut epithelial cells losing the ability to discern between molecules passing from the gut to the bloodstream and vice versa.

The first goal of this study is to determine the abnormal level of ROS toxins produced in the autistic gut, including superoxide (SOX) and hydrogen peroxide (H_2O_2). To calculate ROS toxins produced by metabolic disturbances in the autistic host gut based on dysbiosis features, we present a multicellular metabolic model of gut microbiota and human small intestine that aids in the investigation of the metabolic perturbation occurring in the ASD host intestine. Furthermore, the proposed gut representative model would simulate complex interactions between the gut microbiome and intestinal cells, and predict metabolic changes as potential intervention strategies to mitigate the dysbiosis associated with autism.

Metabolic studies and analyses of elevated concentrations of oxidizable molecules in nervous tissue suggest that proteins involved in normal brain function are susceptible to oxidative modifications, which alter their activity. Specifically, glutamic acid decarboxylase (GAD), an enzyme that converts glutamate to γ-aminobutyric acid (GABA), is vulnerable to oxidative stress [187]–[189]. Furthermore, reduced GABA and elevated extracellular glutamate are known for increasing excitotoxicity, which is reported in individuals with ASD [190]–[192]. Excitotoxicity is also linked to oxidative stress. Thus, our second goal is to construct a combined neuronal model representing the ASD brain. This model would help us calculate the level of oxidative stress and resulting GABA and glutamate in the brain and its relationship with oxidative factors arising from gut dysbiosis (**Figure A.1**).

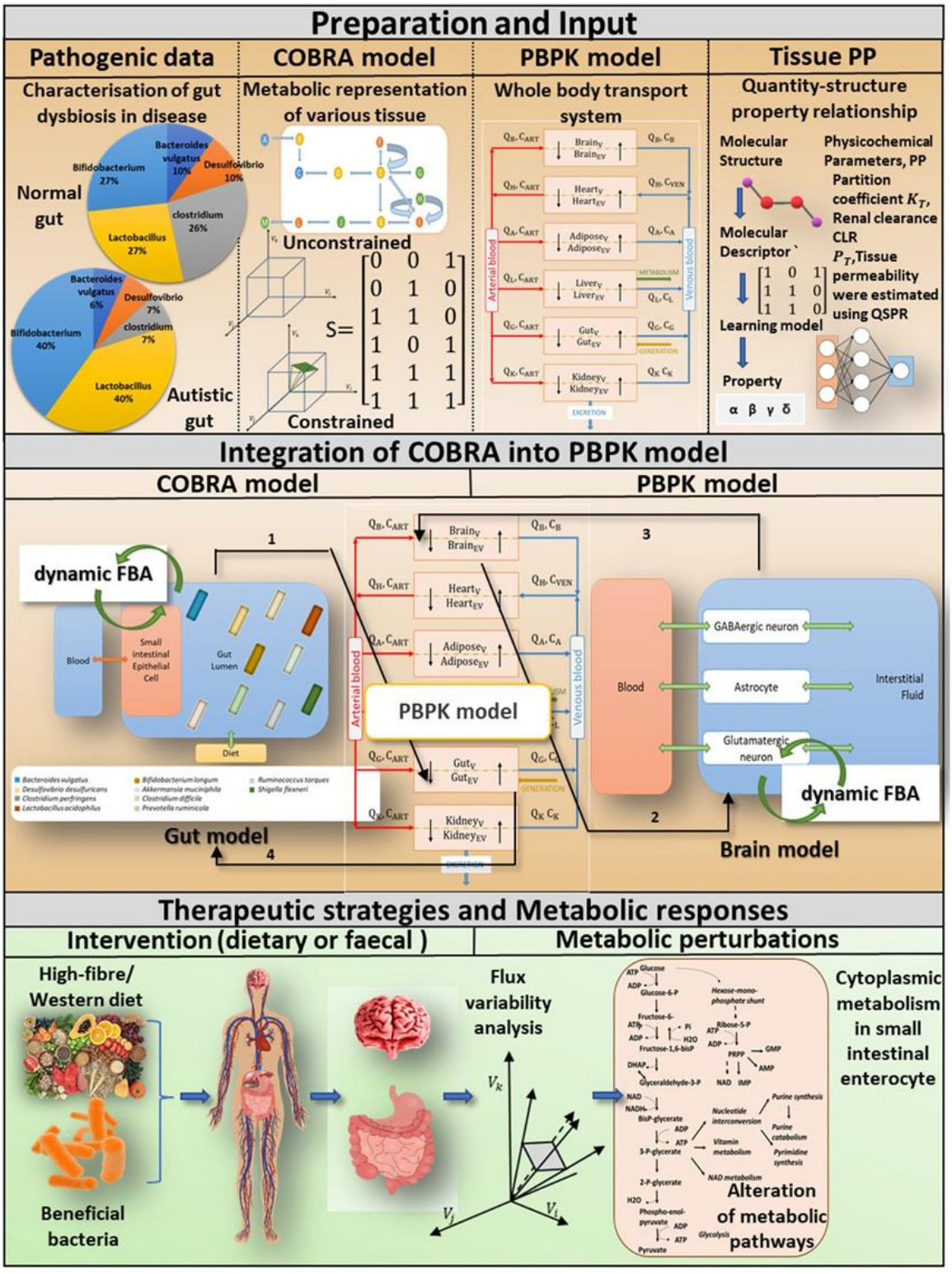


Figure A.1. A computational approach for incorporating tissue-specific metabolic

information into a whole-body transport model to investigate the pivotal role of probiotics and diet in gut dysbiosis-induced diseases.

Preparations and input: We combine four different levels of information. COBRA is a representative gut model developed by combining patient-specific metagenomic gut dysbiosis traits with omics data from metabolic networks. A comprehensive PBPK model is integrated with the COBRA gut and brain models, and the physicochemical properties of the PBPK model are calculated using the Quantity-structure property relationship QSPR.

Integration of COBRA into the PBPK model: The new computational framework enables researchers to investigate the metabolic effects of ROS toxins on the host gut and brain. Tissue-specific toxin metabolism is used to quantify the production/consumption of toxins. The toxin reaction rate is constrained in the organ-specific COBRA model by the PBPK-derived toxin distribution. A static version of the dynamic Flux balance analysis algorithm (dynamic FBA) is used to calculate altered flux distributions in the drug perturbed organ-specific COBRA at a given time instant.

Therapeutic strategies and metabolic responses: Toxin-induced metabolic perturbations, which result in altered intracellular and extracellular reaction rates, can be predicted using a composite representation of whole-body toxin metabolism. The integrated model explicitly considers specific dietary and probiotic interventions, as well as their impact on autistic symptoms.

Toxin generation and exposure to various tissues are challenging to quantify in clinical practice [176], [193], [194]. Furthermore, the determination of toxin-induced neuronal abnormalities using just a bacterial community encompassed a gut model [195]. As a result, a functional assessment of such metabolic dysfunction inevitably necessitates a computational workflow that connects metabolic disturbance in the host gastrointestinal tract to the brain *via* a whole-body transport system to understand gut dysbiosis in the host and plan therapeutic strategies (**Figure A.1**). In multi-compartment PBPK models, where compartments correspond to different organs of the body, a system of differential equations describes substance concentration [196], [197]. On the other hand, constraint-based reconstruction and analysis (COBRA; [198]) enables the analysis of metabolic perturbations and quantitative prediction of possible physicochemical and biochemically phenotypic states by providing metabolic insight (**Figure A.1**). Thus, the third goal of this research is to create a computational framework that will allow researchers to investigate ROS toxin-induced metabolic perturbations in the host intestine and brain using an effective integration platform (*i.e.*, the whole-body transport PBPK model and tissue-specific COBRA models), as shown in **Figure A.2**. Moreover, integrating the two will provide a comprehensive analysis of symptoms of autism in the gut and their influence on the distant organ brain.

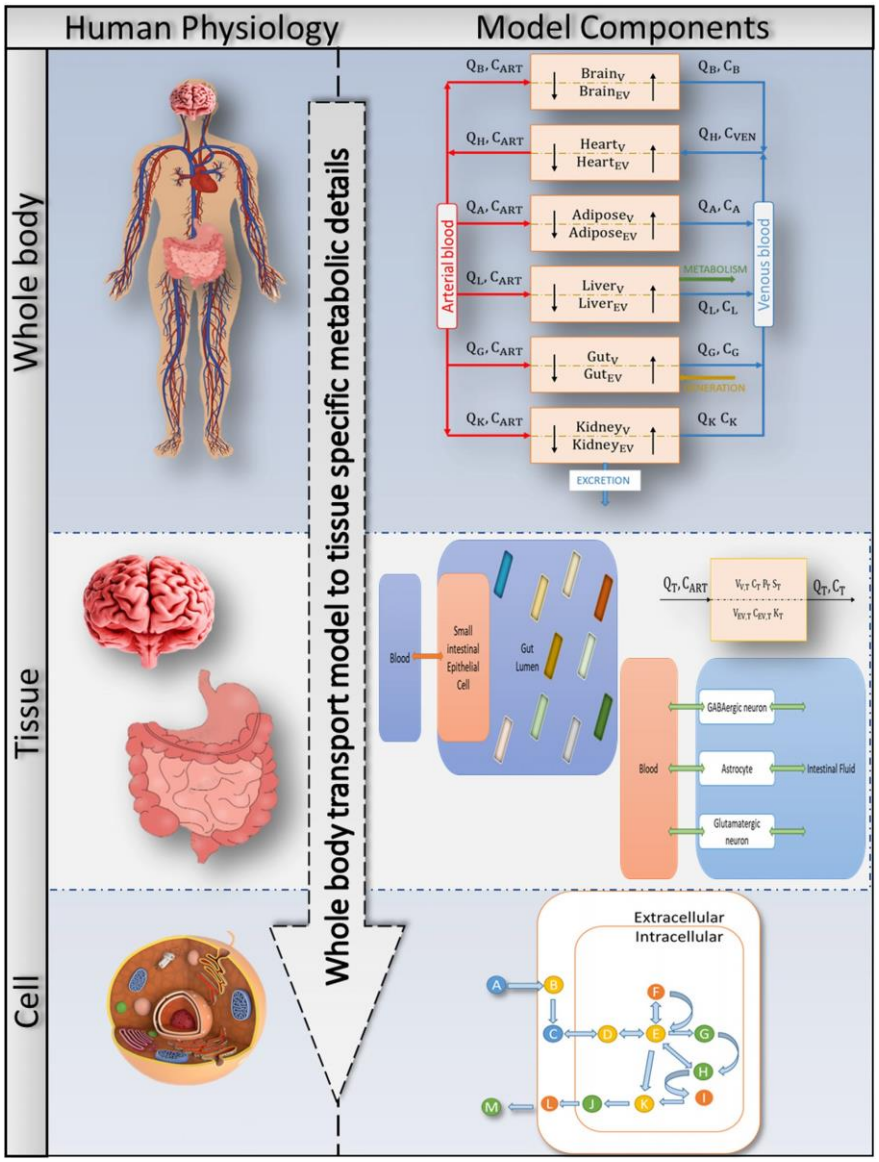


Figure A.2. Bringing together whole-body physiology and model components.

Each tissue of interest is modeled as a two-compartment, permeability-limited, well-stirred tank with vascular and extravascular compartments. The circulatory system connects the various compartments. The cellular biochemistry in the interstitial (extracellular) and intracellular (cytosol, mitochondria, peroxisome, and so on) areas is described using genome-specific COBRA models. A combined PBPK-GSMN model connects tissue-specific metabolic details into a comprehensive representation of a whole-body physiologically based pharmacokinetic (PBPK) model via the metabolism of shared exchange toxins.

Specifically, the presented framework helps us understand and address the following questions:

1. What are the fundamental biochemical factors that drive autism?

2. What is the role of bacteria-derived toxins/metabolites in autism?

3. Can modeling of host-microbe interactions under different dietary conditions hint toward dietary treatment options for autism?

4. How do species added to the microbiome consortia reveal essential metabolic interactions between gut microbes?

A.3. MATERIALS AND METHODS

To investigate dysbiosis-induced metabolic anomalies in the autistic gut vs. the healthy gut, we employed metagenomic data on bacterial abundance to develop an autism gut representation as a dysbiosis feature (**Figure A.3**). To investigate the effects of ROS exposure on diverse organs, including the brain, we integrated tissue-specific metabolic models with a whole-body physiological-based pharmacokinetic (PBPK) model. Mutual exchange of toxins connects the two modeling methodologies, resulting in an integrated COBRA-PBPK model. Furthermore, genome-scale flow patterns characterizing the intercellular and extracellular reactions in the face of toxin exposure are examined for various amounts of food and bacterial intervention, as well as gut bacterial makeup. Genome-scale flow patterns characterizing intercellular and extracellular reactions in the face of toxin exposure are investigated for a variety of gut bacterial compositions and dietary limitations, with each indicating a distinct amount of dietary and bacterial intervention. The outline of the framework is shown in **Figure A.1**.

The structure of the framework stands on three essential components:

1. A genome-scale bacterial community model merged with an intestinal epithelial cell model of the host representing the human gut.

2. Combining neuronal models to represent the human brain.

3. The whole-body PBPK transport model considers six organs interacting through the circulatory system of the body.

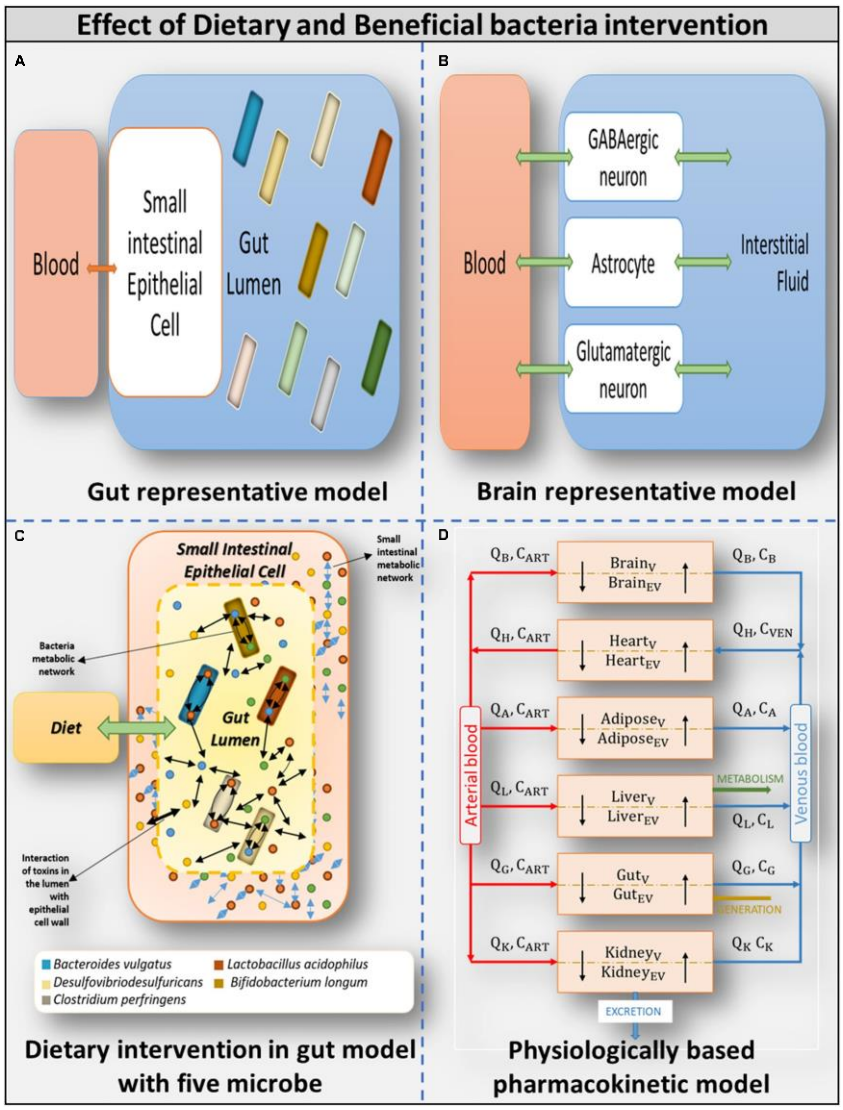


Figure A.3. Overview of models used in the gut-brain axis analysis.

(A) Gut representative model: Overall integrated Gut representative model. **(B)** Brain representative model: Brain model and its components. **(C)** Dietary intervention in gut model with five bacterial models: Gut representative model interacting with diet and host-microbe and microbe-microbe interaction. **(D)** Physiologically based pharmacokinetic model: Whole-body PBPK model.

A.3.1. Model construction

A.3.1.1. Personalized bacterial community combined with host intestinal cells to represent the human gut

The human gut microbiome is a diverse, complex system that responds dynamically to its microenvironment. The microbiome has a significant impact on the well-being of humans. Understanding the inherent ability of the human microbiome to interact in both one-way (commensal) and two-way (mutualistic) manners [199], we built a personalized bacterial community in which the gut microbiome is simplified to the point where the pathophysiological condition associated with ASD can be easily captured [200]. The microbial community is also merged with an intestinal host to represent the typical human gut (**Figure A.3**). The microbial community is designed to crosstalk with each other and interact with the host [201], [202]. The bacterial community is built using the Microbiome Modeling Toolbox [203]. Individual microbial models were combined in this community modeling design setup by adding reactions to each microbial model, *i.e.*, transporting metabolites from the extracellular space to the common lumen compartment. Metabolites in the lumen compartment are constrained by diet or fecal compartment, enabling bilateral interaction with the environment.

One of the most popular biology methods for studying biological cell systems with mechanistic details is constraint-based reconstruction and analysis (COBRA; [198]). Numerous studies have used COBRA tissue modeling to investigate the interaction between gut bacteria and the host gut. The assembly of gut organisms through reconstruction and analysis (AGORA) project [204] presents genome-scale metabolic models derived from human gut metagenomic data that can be used to understand how microbial communities influence human metabolism and health. A collection of 773 metabolic reconstructions/models of the 13 most common human gut microbe species has

been made public [204]–[206]. Thus, combining an individual constraint-based genome-scale metabolic model of human small intestinal enterocyte cell (sIEC; [207]) and gut bacteria implicated in autism will result in a gut metabolic model, as shown in **Table A.1**.

Table A.1. Gut bacterial abundance in the autistic gut when compared to the normal human gut.

Bacteria	Abundance in the autistic gut compared to normal human gut	References
Bacteroides vulgatus ATCC	Increased	Finegold, 2008; Finegold et al., 2010; Ding et al., 2017; Vuong and Hsiao, 2017
Desulfovibrio desulfuricans subsp. desulfuricans	Increased	Finegold, 2008; Finegold et al., 2010; Ding et al., 2017; Vuong and Hsiao, 2017
Clostridium perfringens	Increased	Finegold, 2008; Finegold et al., 2010; Ding et al., 2017; Vuong and Hsiao, 2017
Lactobacillus acidophilus	Decreased	Sandler et al., 2000; Shaaban et al., 2017; Vuong and Hsiao, 2017
Bifidobacterium longum longum	Decreased	Sandler et al., 2000; Shaaban et al., 2017; Vuong and Hsiao, 2017
Akkermansia muciniphila	Decreased	Sandler et al., 2000; Shaaban et al., 2017; Vuong and Hsiao, 2017
Clostridium difficile	Increased	Sandler et al., 2000; Shaaban et al., 2017; Vuong and Hsiao, 2017
Prevotellaruminicola	Decreased	Sandler et al., 2000; Shaaban et al., 2017; Vuong and Hsiao, 2017
Ruminococcus torques	Increased	Finegold, 2008; Finegold et al., 2010; Ding et al., 2017; Vuong and Hsiao, 2017
Shigella flexneri	Increased	Finegold, 2008; Finegold et al., 2010; Ding et al., 2017; Vuong and Hsiao, 2017

A.3.1.2. Gut microbiota representative models

Five bacteria personalized community

A COBRA model of five important bacteria species, *Bacteroides vulgatus*, *Clostridium perfringens*, *Desulfovibrio desulfuricans*, *Lactobacillus acidophilus*, and *Bifidobacterium longum*, shown in Supplementary Files/Additional File 1/**Table S11A**, was coupled with a genome-scale model of the sIEC, representing the human gut with five models of bacteria (specification of combined model shown in Supplementary Files/Additional File 1/**Table S11A**. Furthermore, harmful bacteria in higher abundance (**Table A.1**) were connected to represent purely autistic gut, while beneficial bacteria are linked to represent beneficial gut.

The six separate models listed below were developed to assess the impact of dietary intervention.

1. Gut microbiome: high fiber: bacterial community model constrained with a high-fiber diet.

2. Gut microbiome: Western: bacterial community model constrained with the Western diet.

3. Gut microbiome harmful: high fiber: harmful bacteria (only bacteria present in abundance in the autistic gut) model constrained with a high-fiber diet.

4. Gut microbiome harmful: Western: harmful bacteria (only bacteria present in abundance in the autistic gut) model constrained with the Western diet.

5. Gut microbiome beneficial: high fiber: Beneficial bacteria (Gut bacteria present in very less percentage in the autistic gut) model constrained with High fiber diet.

6. Gut microbiome Beneficial: Western: beneficial bacteria community (gut bacteria present in very less percentage in autistic Gut) model constrained with the Western diet.

The sIEC model, hs_sIEC61 [207], was expanded with new pathways to capture the metabolism of bacteria-derived toxin metabolites. A total of 12 reactions and seven metabolites were added to include the metabolism of propionic acid and essential reactive oxygen species, such as H_2O_2 and SOX (Supplementary Table/Additional File 1/Table S12). The gut bacteria and sIEC interact through the exchange of metabolites *via* the lumen compartment. Thus, to generate the combined model, 665 luminal transport reactions were added for extracellular metabolites of the bacterium models (Supplementary Files/**Table 1**).

For each of the gut representative models, exchange reactions were constrained by intestinal conditions. An example is the aerobic condition of the small intestine, by constraining the lower bound of oxygen exchange reactions in bacterial models to lb = −1 mmol/gDW/hr (Supplementary Files/**Table 1**).

Ten Bacteria Personalized Community

To understand the scalability factor in our personalized gut model, we increased the size of the community to ten bacteria (both harmful and beneficial). Supplementary Files/Additional File 1/**Table S11B** shows the ten selected bacteria in the expanded ten microbial communities and their specifications in individual and combined models. In addition, the entire technique for building the model is detailed in Supplementary Files/**Additional File 1**.

A.3.1.3. Combined neuronal model representing the human brain

Recent investigations have shown that increased levels of oxidative stress in the brain result in an increased level of glutamate and a reduced level of GABA neurotransmitters [208]. We built a brain model by combining previously published GABA and glutamate neuronal models [209]. Each neuronal model comprises two cell types, the

neuron and the supporting astrocyte cell (Supplementary Files/Additional File 1/**Table S11A**). After quality checks and corrections, the individual neuronal models were then combined to form a combined whole-brain model (Supplementary Files/**Table 1**). Furthermore, Supplementary Files/Additional File 1/**Table S11A** shows the published neuronal model specifications and the whole-brain model. Details of the model components are shown in Supplementary Files/**Table 1**.

A.3.1.4. A whole-body physiological-based pharmacokinetic transport model

Constraint-based reconstruction and analysis operate under steady-state conditions [198]. Hence, to capture the dynamics of autism pathogenesis, a whole-body physiological-based pharmacokinetic (PBPK) modeling approach is required [210], precisely to model the transport and effect of gut-derived toxins (H_2O_2 *and* O^{-2}) through the gut-brain-axis in the brain in autism. Therefore, the third most crucial component in building the computational framework is the whole-body transport model. We considered a semi-detailed PBPK model of six organs, as shown in **Figure A.3**.

The PBPK transport model considers each tissue type of interest as a two-compartment, permeability-limited, well-stirred tank consisting of a vascular compartment and an extravascular compartment [211]; shown in **Figure A.3**). The various compartments communicate with one another *via* the circulatory system of the body. Thus, movement and accumulation of the compound as a function of time were quantified in each tissue type through a set of ordinary differential equations (shown in Supplementary Files/Additional File 1/**Table S1**). The steps involved in building a PBPK model are shown in Supplementary Files/Additional File 1/**Table S2**, and details are shown in Supplementary Files/Additional File 1/**Table S2**.

One of the main objectives of this study is to examine the effect of ROS in the brain of individuals with autism; therefore, we included tissues that metabolize ROS, specifically SOX and H_2O_2, or whose function is affected by ROS in the full-body PBPK model. Thus, the minimalistic PBPK model included (i) the heart as a source and sink of the circulatory system, (ii) the gut, and (iii) the brain, as they were the central organs in our study. Since increased ROS levels are implicated in adipocyte dysfunction [212], (iv) adipose tissue was included in the model. Being the central metabolic organ, (v) the liver was included to maintain the generality of the model, and (vi) the kidneys were included to allow for excretion shown in components of the model in **Figure A.3**. To formulate the model equations, tissue-specific characteristics that affect the uptake of the molecule of interest needed to be defined.

As mentioned earlier, elevated levels of ROS in the autistic gut lead to inflammation of the gut epithelial wall, which further results in increased permeability of the gut wall. It was also determined that once toxins pass through the blood-brain barrier, they may cause brain inflammation [176]. Thus, to account for the increased epithelial permeability of the gut wall and brain of individuals with autism vs. controls, a tissue model that accounts for organ permeability in its formulation was used. Thus, permeability-limited, two sub-compartment models were formulated and used for further analysis (**Figure A.3**).

Governing mass balance equations for the permeability-limited model are as follows:

$$\frac{fV_T}{1+f}\frac{dC_T}{dt} = Q_T C_{ART} - Q_T C_T + \frac{P_T S_T C_{EV,T}}{K_T} - P_T S_T C_T \,, \qquad (A.1)$$

$$\frac{V_T}{1+f}\frac{dC_{EV,T}}{dt} = P_T S_T C_T - \frac{P_T S_T C_{EV,T}}{K_T} \,, \qquad (A.2)$$

$$and\ f = \frac{V_T}{V_{EVT}}. \qquad (A.3)$$

The final set of ordinary differential equations used was simply the above two mass balance equations written for each of the six tissue types, *i.e.*, brain, heart, adipose, liver, gut, and kidney (Supplementary Files/Additional File 1/**Table S1**), with additional generation and consumption terms obtained from the COBRA models, as detailed in the supporting information text.

A.3.2. Parameter estimation in the physiological-based pharmacokinetic model

The novelty of the PBPK model lies in its semi-detailed and highly curated nature, wherein the physiological and biochemical parameterization of the model was primarily literature-driven (Supplementary Files/Additional File 1/**Table 2D**). While physiological parameters like blood flow rate through different tissue types, tissue surface area, and tissue volume, and biochemical parameters like enzymatic reaction rate and Michaelis constant were largely obtained from the literature, physicochemical parameters like partition coefficient K_T, renal clearance CL_R, and tissue permeability P_T, were estimated using Quantitative structure-property relationship (QSPR; Supplementary Files/Additional File 1/**Table S3**).

A.3.3. Quantitative structure-property relationship

Quantitative structure-property relationship (QSPR; [213], [214]) models assume a strong correlation between the structure of a compound and its physical and chemical properties like melting point, boiling point, and in our case, tissue-plasma partition coefficient. The fundamental assumption behind QSPR is that molecules with similar structures have similar observable properties and that molecules with different structures

have different observable properties. QSPR models are built with predictors consisting of theoretical molecular descriptors and fragment descriptors of a compound, derived solely from the molecular structure of the compound and not from experimental data [215]. The output variable is the physicochemical property of interest. Further information on the calculation of physicochemical properties is discussed in Supplementary Files/**Additional File 1.**

A.3.4. Integration of tissue-specific constraint-based metabolic model and a whole-body physiological-based pharmacokinetic transport model

Integrated modeling of the gut-brain link in autism pathogenesis combines the following components, *i.e.*, representative gut community model and combined neuronal model with the whole-body dynamic PBPK model (**Figures A.1 and A.2**). Integration is achieved by stepwise discretization into time steps. In each time step, the maximum consumption or production of toxin metabolites by the COBRA model (modeled as input and output exchange reactions) was calculated, and fluxes through these reactions were extracted and converted to a concentration value that is fed in the PBPK model to calculate the constraint for the next time step in the brain and gut specific-genome scale metabolic model. An initial concentration of 0.0001 M was assumed for H_2O_2 and SOX, and a step size of 15 min was used for the integration and simulation ran for 6 h.

In the integrated model, concentrations were changed in every time step not only because of consumption or production of metabolites by the COBRA model (modeled as input and output reactions, respectively) (Supplementary Files/Additional File 1/**Tables S12, S13**) but also because of the rate of metabolite transport. The transport depends on permeability and other parameters taken into consideration by the PBPK model, achieved by computing bounds on the input fluxes to the COBRA model using the concentrations

obtained from the PBPK model in every time instant. To ensure metabolite input into the COBRA model, the reversible exchange reaction was replaced by two irreversible reactions, an input reaction constrained by the PBPK model bound and an output reaction, as shown in Supplementary Files/Additional File 1/**Table S13**.

The COBRA analysis calculated its consumption and generation terms, which were then used as the metabolite's actual input and output exchange fluxes by the PBPK model. Furthermore, the PBPK approach combines concentrations for the next instant. The process was repeated with the obtained concentrations. The entire procedure is described in Algorithm 1 (**Figure A.4**).

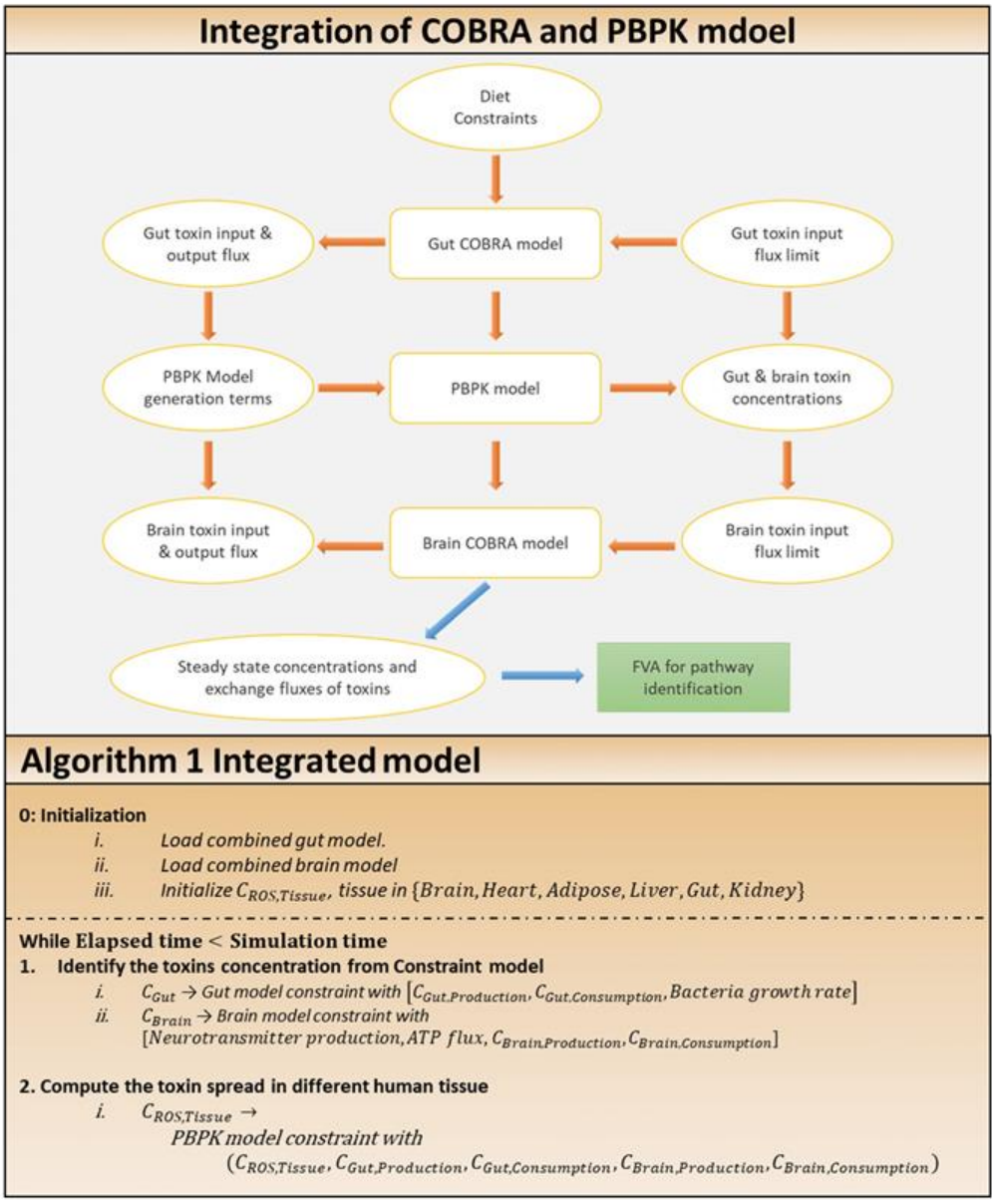


Figure A.4. Approach for the integration of COBRA and PBPK model.

Integration of COBRA and PBPK model: the integrated model, showing the interactions between the gut, brain COBRA models and the transport PBPK model. Simulation of the integrated model yields the steady-state exchange fluxes of toxins used for further analysis through FVA to identify altered pathways.
Algorithm 1: An algorithm for the integration of two models.

The following is a simulation of the representative gut COBRA model for toxin calculation. Under the maximal bounds set for the toxin inputs, Pareto-optimal growth (discussed in the section "Model Analysis" below) of all cell types in the model was computed and set as bounds for the respective biomass reactions. Then, maximal toxin production (output exchange) fluxes were determined by Pareto-optimization and set as bounds to the output exchange constraints. The maximal toxin input fluxes were determined as consumption terms for the PBPK model by Pareto-optimization, and output exchange fluxes as generation terms. The COBRA simulation used the same procedure for the brain in each time step, except that the ATP maintenance reaction was maximized rather than biomass_maintenance, as shown in Supplementary Files/Additional File 1/**Table S14**. Furthermore, neurotransmitter production fluxes (modeled as demand reactions for GABA and glutamate) were computed by Pareto-optimization under the calculated constraints. The steps involved in integrated model analysis are shown in **Figure A.4**.

As shown in Supplementary Files/Additional File 1/**Table S15**, for each time step, the following Pareto-optimization problem is solved depending on the net objective:

$$\max w^T \bar{v} \qquad (A.4)$$

$$where, \quad \bar{v} \in X \ (pareto\ feasible\ space)$$

$$such\ that, \quad S^* v_i = 0,$$

$$and, \quad lb \leq v_i \leq ub$$

Unlike the COBRA model, a flux-based model, the PBPK model requires toxins to be input in concentration. To establish a link between the production and consumption of toxins in two platforms of dynamic and steady-state modeling, the flux from COBRA is appropriately converted to concentration as follows:

$$Flux\left(\frac{nmols}{gram\ weight\ of\ tissue\ hr}\right) = C(M) \times \frac{volume\ of\ extravascular\ region\ of\ tissue\ in\ rat}{gram\ weight\ of\ organ\ in\ rat} \qquad (A.5)$$

The volume of the extravascular region of the brain and the gut was obtained from the PBPK model, and gram weight was obtained from experimental values defined for rats, *i.e.*, the small intestine was 7.26 g [216], and the brain was 2 g [217], [218].

A.3.5. Model analysis

Having constructed a novel computational setup for integrating different model types, we now present the framework for its analysis.

A.3.5.1. Analyzing dietary intervention

An integrated model is developed in this study to understand the complex interactions between gut microbiome composition and the human host under different dietary constraints in the autistic vs. healthy case. The COBRA genome-scale metabolic models of the gut and brain were used to represent the human host. To transport gut-derived metabolites across the COBRA genome-scale metabolic models of the tissues, the PBPK model was used. Two different dietary regimes were used as diet constraints, *i.e.*, a high-fiber and a Western diet (details in Supplementary Files/**Additional File 1**). The combined gut model was constrained with diet by only opening exchanges in the model that correspond to nutrients in the diet for uptake. Six separate models are constructed and analyzed for the effect of dietary intervention.

Furthermore, in constraint-based models, the PBPK model aids in establishing precise bounds for toxin exchange reactions. The integrated approach is essential because, in the absence of a PBPK model, constraint-based models would have been initialized with arbitrary bounds not representing actual physiological conditions. We demonstrated this by choosing the steady-state flux values to be 100 times the actual values obtained from

the PBPK model to simultaneously study the effect of an increase in toxin fluxes (Supplementary Files/**Table 2**).

A.3.5.2. Analyzing the effect of varying gut bacteria compositions on the autistic gut

The abnormal composition of gut bacteria inside the human gut, a source of gut-brain-axis inflammation associated with autism, can be significantly reduced by intervention with beneficial bacteria. To illustrate the intervention with beneficial bacteria, we developed five separate models of the gut microbiome to analyze the effect of varying compositions of gut bacteria, representing their different levels in autism, as shown below.

Five separate models of the gut microbiome to analyze the effect of varying gut bacteria composition, representative of their different levels in autism are listed below:

1. Purely autistic gut (also gut microbiome-harmful): Constructed by combining sIEC and the individual models of harmful bacteria grown at the maximum rate.

2. Typical autistic gut: Constructed by combining the sIEC model with the individual models of all the five bacteria, with the harmful bacteria forming 80% of the total gut bacterial population.

Table A.2. Details of the microbiome community models.

(A) Weights for the biomass reaction at different percentages of beneficial bacteria abundance

Five microbial community model

S. No.	Beneficial bacteria %	% contribution of biomass reaction to the objective in weighted Pareto optimality					
		LA	**BL**	**BV**	**DD**	**CP**	**Biomass SIEC**
1	20	0.0833	0.0833	0.2222	0.2222	0.2222	0.1667
2	40	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667
3	60	0.25	0.25	0.1111	0.1111	0.1111	0.1667
4	80	0.3333	0.3333	0.0556	0.0555	0.0556	0.1667

B) Growth rates of gut microbiome cells (mmol/gDW/hr)

Five microbial community: Individual and combined model specification

Model	**Diet**	**BV**	**DD**	**CP**	**LA**	**BL**	**SIEC**
Gut microbiome	*Western*	1.06472	1.0585	0.3129	0.7336	0.9927	0.00972
	High-Fiber	1.05109	1.04102	0.3057	0.73197	1.0106	0.00933
Gut microbiome – Harmful	*Western*	1.3402	1.2940	0.4529	–	–	0.01601
	High-Fiber	1.2675	1.2220	0.4448	–	–	0.01601
Gut microbiome – Beneficial	*Western*	–	–	–	0.1586	0.1511	0.01984
	High-Fiber	–	–	–	0.3098	0.6589	0.02546

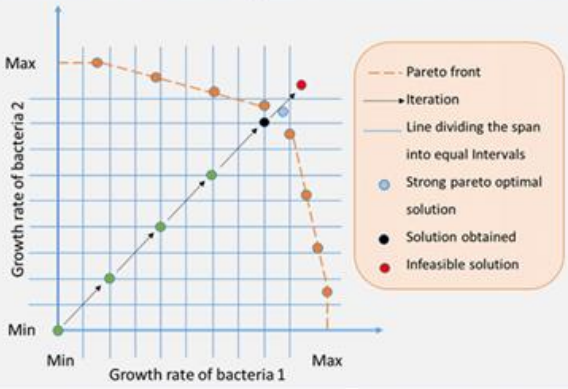
A.3.5.3. Multi-objective optimization of growth rates or Adenosine triphosphate maintenance fluxes of individual cells

Multi-objective optimization techniques are required to determine the growth rates or ATP maintenance fluxes of individual cells in a multicellular model, such as the gut microbiome and brain [219]. Other multi-objective optimization problems on which our research focuses, such as the production and consumption of various toxins and neurotransmitters, were also raised, and each compound was given equal importance. Such situations were encountered while simulating the typical autistic gut and typical healthy gut models.

Pareto-optimization solution to the multi-objective optimization problem

A Pareto-optimal solution is required for multi-objective optimization. Two Pareto-optimization algorithms were developed and implemented, employing linear search (Algorithm 2, **Figure A.5**) for equally weighted Pareto-optimization and binary search for unequally weighted Pareto-optimization. Each objective (e.g., the biomass of each cell) was assigned an equal weighting in a system with an equal number of cells of each type, because it is expected that each cell in the system will try to maximize its growth regardless of external conditions.

Pareto optimization



Algorithm 2 Linear search Un-weighted Pareto-optimization algorithm

0: Initialization

i. *Flux variability analysis FVA for Objective function equation.*
ii. *Split uniformly feasible range of flux value.*
iii. *Initialize with minimum feasible flux point.*

For StepNo = 1: Nstep

1. Find new point

$$New\ point \rightarrow minFlux + \frac{StepNo}{Nstep}(maxFlux - minFlux)$$

2. Check whether new Point is feasible

Yes: $New\ point \rightarrow minFlux + \frac{StepNo}{Nstep}(maxFlux - minFlux)$

No: $New\ Point \rightarrow minFlux + \frac{StepNo-1}{Nstep}(maxFlux - minFlux)$

3. **Compute last feasible point**

Algorithm 3 Binary search weighted Pareto-optimization algorithm

0: Initialization

i. *Flux variability analysis FVA for Objective function equation.*
ii. *Split uniformly feasible range of flux value.*
iii. *Initialize with minimum feasible flux point.*

While Step $\leq$ n

1. Find new point

$$New\ point \rightarrow minFlux + \frac{weight}{weight+1}(maxFlux - minFlux)$$

2. Check whether new Point is feasible

Yes: $New\ point \rightarrow maxFlux$

No: $New\ Point \rightarrow minFlux$

3. **Compute last feasible point**

Figure A.5. Multi-objective optimization in a multi-cellular system.

Pareto optimization: Pareto optimization search technique.
Algorithm 2: Linear search unweighted Pareto optimization algorithm.
Algorithm 3: Binary search weighted Pareto-optimization algorithm.

Linear search

In a system with an equal number of cells of each type, equal weights are awarded to each objective (say, the biomass of each cell), since it is expected that each cell in the system will try to maximize its growth disregarding external conditions (*i.e.*, a cell tries to survive in any environment and does not give more importance to the survival of other cells over its own).

Weighted Pareto-Optimization

Other Pareto-optimal solutions can be obtained by choosing different slopes of lines from the starting point (minimum fluxes in each dimension). Consider the following scenario: in the system, bacterium 1 accounts for 80% of the total microbial composition, while bacterium 2 accounts for the remaining 20%. The algorithm will assign weights to the increment step at a 1:4 ratio. **Figure A.5** depicts a graphical representation of Algorithm 1 in the solution search space. Stopping criteria include the maximum number of iterations or the Euclidean norm of the difference among subsequent iterates falling below a specified tolerance value. In weighted Pareto-optimization, a binary search technique finds other Pareto-optimal solutions by varying the slopes of lines from the starting point (minimum fluxes in each dimension).

A.3.5.4. Crosstalk between the gut microbiome and contribution to secretion products

The gut microbiome model enables the investigation of microbial interaction, which is determined by performing flux variability analysis on the gut microbiome model and identifying active reactions that were revealed. Flux variability analysis (FVA) [220] is helpful to determine the minimum and maximum fluxes for responses in a network while maintaining some network state, such as the rate of biomass production. With the powerful

FVA tool in hand, new questions about the flexibility of biochemical reactions in the network of various organism phenotype states can be addressed.

A.3.5.5. Model comparison for analyzing the metabolic response

Flux variability analysis, *i.e.*, FVA, which computes network variability, is performed to analyze metabolic perturbation in the brain and gut model under varying simulation constraints. We defined two metrics using the fluxes of a reaction in a model:

$$Mean\ shift = \left| \frac{FVA_{Max1} + FVA_{Min1}}{2} - \frac{FVA_{Max2} + FVA_{Min2}}{2} \right| \qquad (A.6)$$

$$Range\ change = (FVA_{Max1} - FVA_{Min1}) - (FVA_{Max2} - FVA_{Min2}) \qquad (A.7)$$

Before changing conditions, FVA_{Max1} and FVA_{Min1} were the maximum and minimum fluxes of a reaction in a model. FVA_{Max2} and FVA_{Min2} are the maximum and minimum fluxes after the conditions are changed.

These metrics are used to compare the responses of a model to the effect of change in conditions. Both these metrics were computed for each of the reactions of interest in the model, and the reactions were ordered in decreasing order of the mean shift. In case of a tie, the decreasing order of range change was used to determine the order. Thus, an order was determined for the effect of modifications in a model on all the reactions of interest.

Moreover, a topological analysis, called pathway analysis, module measures metabolic subsystems in a metabolic network that have a significant shift in the flux of reactions associated with the subsystems. To identify a statistically significant subsystem having significant pathway shifts, we define a pathway score as below.

$$Pathway\ score = \frac{No.of\ shifted\ reactions}{Total\ no.of\ reactions\ in\ subsystem} \qquad (A.8)$$

Subsystems with high pathway scores represent a high degree of changes in the reactions.

A.3.5.6. Metabolic pathways response due to increased oxidative stress

The effect of oxidative stress was assessed by comparing the toxic and non-toxic models (detailed in the section "Results") for both the brain and Gut by FVA and was determined by setting the lower bounds of toxin input and output exchanges to the steady-state flux values obtained from the integrated model. The non-toxic model allowed for lower exchanges of toxins. Implications of the leaky-gut hypothesis and increased permeability of the blood-brain barrier were examined through simulation of this phenomenon on the brain model by allowing toxins into the brain model.

Furthermore, the effect of toxins on the brain was studied by comparing a toxic with a non-toxic model of the brain. The integrated model's steady-state values for brain toxin exchanges were used to set the toxin exchange bounds for the brain model. Likewise, for the gut model, FVA was performed to compare the models in both analyses.

A.4. RESULTS

A.4.1. Dietary influence on gut bacterial growth

As mentioned earlier, one of the major objectives of this study was to determine the impact of a potential dietary intervention strategy on autism. A Western diet and a high-fiber diet were selected as two typical dietary patterns that promote the growth of beneficial bacteria. Each of the aforementioned gut representative models, both autistic (gut microbiome-harmful) and healthy (gut microbiome), was subjected to dietary and beneficial bacteria interactions. We observed that the addition of the Western diet to the

purely autistic gut, *i.e.*, gut microbiome-harmful, decreased the growth rates of each of the harmful bacteria compared to the gut microbiome model, despite having lesser quantity than the harmful bacteria, as shown in **Figure A.6**; growth rates are shown in **Table A.2B**. Since the growth rate of healthy bacteria increased (when compared to the Western diet) to a larger extent compared to that of harmful bacteria in the high-fiber diet, especially with a higher percentage of healthy bacteria in the gut, we can infer that the high-fiber diet is better than the Western diet for growth of beneficial bacteria, and consequently, alleviation of autism symptoms, as discussed below.

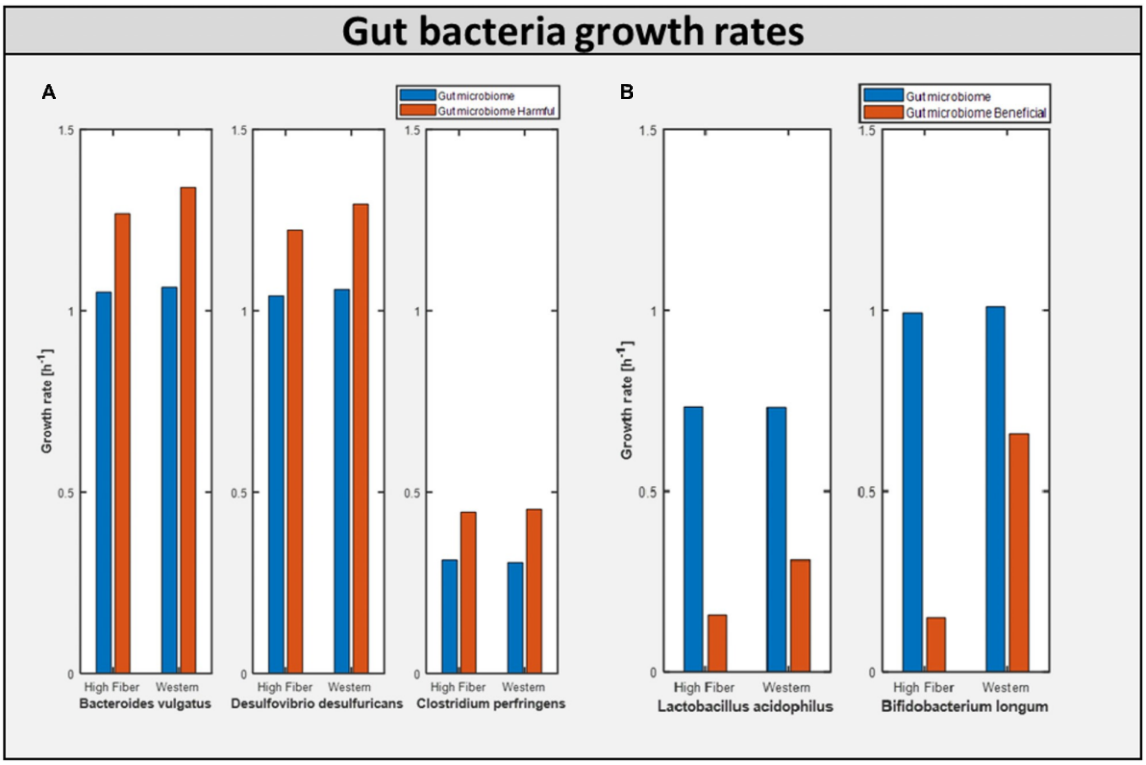


Figure A.6. Bacterial growth rate in differential model.

(A) Growth of harmful bacteria as compared in the two model Gut microbiome and Gut microbiome harmful. **(B)** Growth of beneficial bacteria as compared in the two model Gut microbiome and Gut microbiome beneficial.

A.4.2. Toxins derived in the gut microbiome model

Calculating the toxins produced inside the gut is another approach of assessing the influence of diet on improving gut health. **Figure A.7** shows the production of toxins inside the typical gut subjected to a different diet. **Figure A.7A** shows the production of toxins inside the gut microbiome-harmful model, *i.e.*, gut with harmful bacteria inside. On the other hand, **Figure A.7B** shows that after adding beneficial bacteria into the microbial consortia, the level of toxins was significantly reduced in the case of the high-fiber diet. This suggests the need for both beneficial and high fiber diets to improve autism symptoms inside the gut.

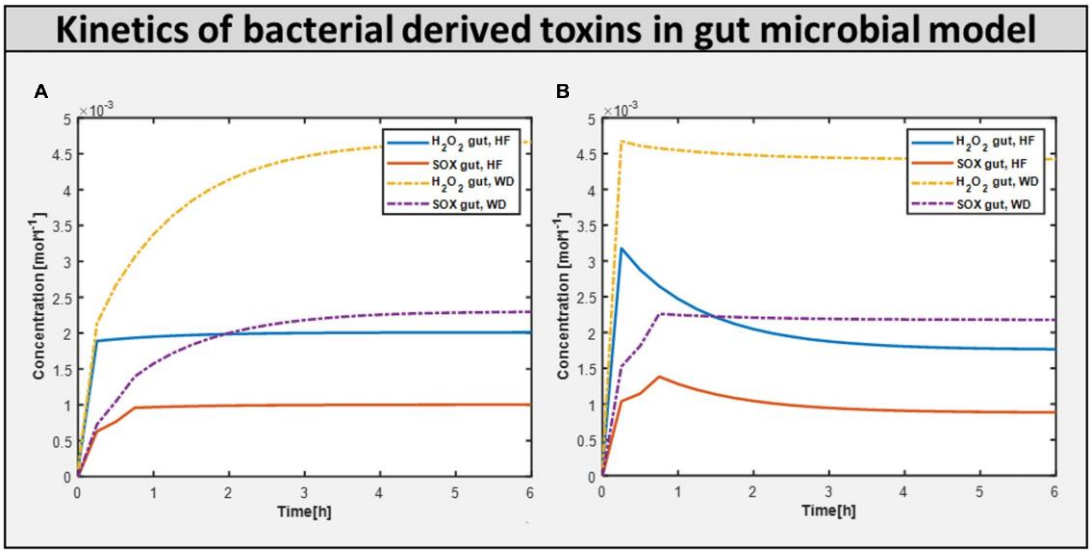


Figure A.7. Bacteria-derived toxins.

(A) Toxins produced H_2O_2 (hydrogen peroxide) and SOX (superoxide) in the Gut microbiome harmful model constrained with differential diet. **(B)** Toxins produced H_2O_2 and SOX in Gut microbiome beneficial model constrained with differential diet High fiber and Western.

A.4.3. Microbiome and diet influence intestinal and metabolic functions in autism

Now, to better understand how the addition of beneficial bacteria to microbiome consortia improves autism symptoms in the gut, we explored another way of intervention: adding beneficial bacteria at various levels of intervention. To examine the effect of helpful bacteria, we introduced them to the purely autistic gut model, *i.e.*, gut microbiome-bad bacteria, to create a model with varying levels of beneficial bacteria intervention, as shown above in model analysis, and found that the beneficial bacteria grew at a faster rate. **Figure A.6** depicts the growth rate of several bacteria in the gut model. Contrastingly, the addition of beneficial bacteria led to reduced flux in reactions linked to oxidative stress metabolism. Therefore, the concentration of ROS toxin is significantly reduced, as shown in **Figure A.7B**.

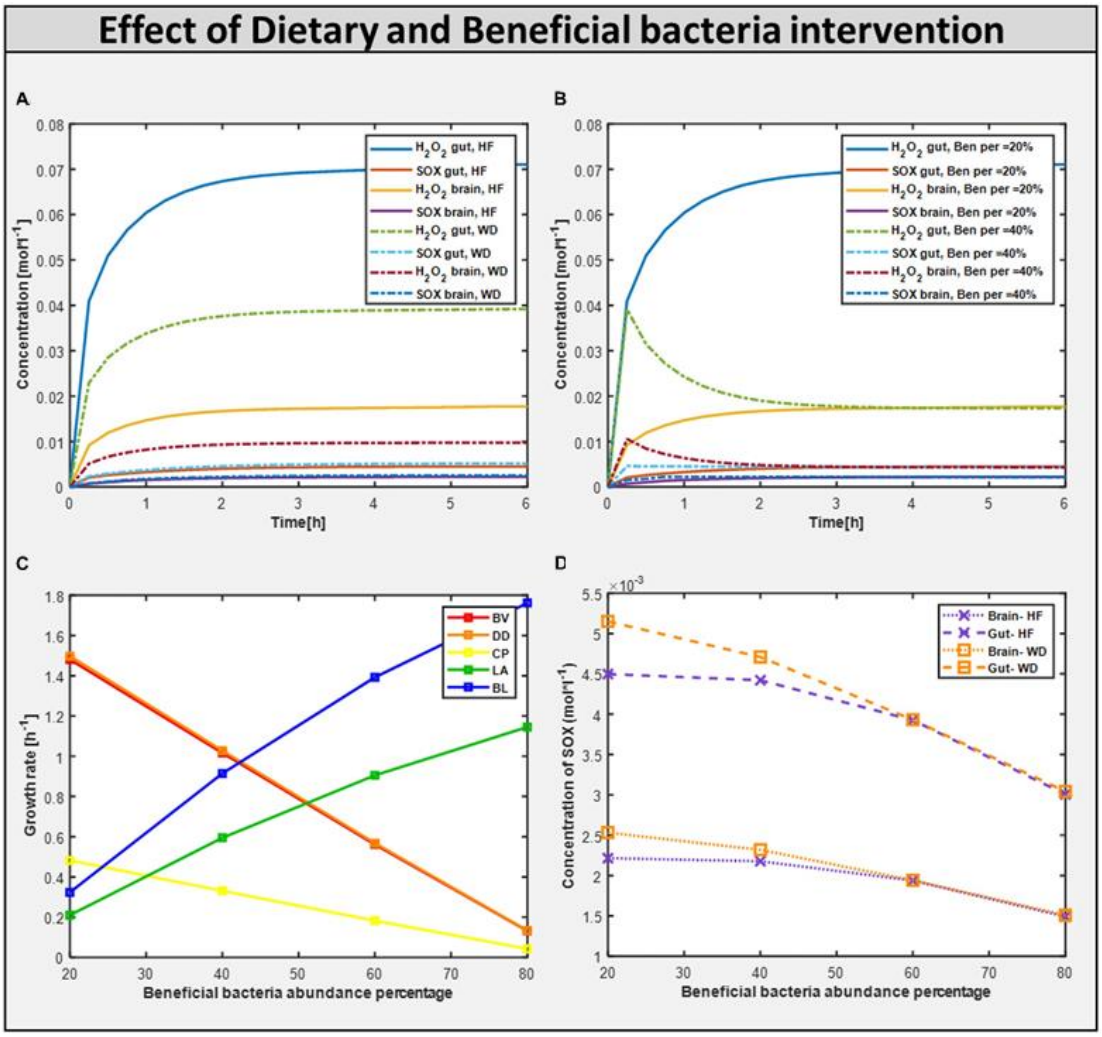


Figure A.8. Effect of dietary and probiotic intervention.

(A) Toxins profile in 20% abundance of beneficial bacteria in the community with change in diet from HF to WD. (B) Toxins profile with varying beneficial bacteria abundance in the community from 20% to 40% with fixed diet HF. (C) The growth rate of different bacterial biomass at differential abundance percentages. (D) Reducing the build-up of ROS toxin *i.e.* Superoxide (SOX) with the help of increasing beneficial percentage along with the correct diet.

A.4.4. Changing gut microbial composition with beneficial bacteria improved autistic characteristics

By quantifying the level of toxins reaching the brain that are produced in the gut, the involvement of bacteria-derived toxins/metabolites in autism was identified. The toxins produced/consumed are allowed to reach a steady-state value before being fed to PBPK for whole-body toxin distribution in each time instant. The concentration profile of toxins in the gut and brain models fed with a high-fiber diet and 20% abundance of beneficial bacteria in the gut model are shown in **Figures A.8A and B**, respectively. By increasing the fraction of beneficial bacteria in the gut, we observed a monotonic decrease in the growth rate of all harmful bacteria under both diets, as well as a monotonic decrease in the level of oxidative stress (**Figure A.8**) in both the brain and the gut, which is equivalent to administering probiotics to an individual with autism.

A.4.5. Neurotransmitters in the integrated brain model

Studies and assessments on elevated levels of reactive oxygen species in neural tissues reveal that proteins involved in proper brain function are susceptible to oxidative changes, which alter their action. This is measured by determining the amount of GABA neurons in the brain that have been damaged by oxidative toxins.

Maintenance flux for each of the neuronal cell types in the brain model was obtained (Supplementary Files/Additional File 1/**Table S16**). Additionally, the model indicated that the flux through synthesis reactions of GABA, an inhibitory neurotransmitter, is increased upon intake of beneficial bacteria, as shown in **Figure A.9**. This demonstrates that by increasing the level of probiotic intervention, the level of GABA required for normal brain activity may be retained, supporting studies on GABA dysfunction in autism (detailed in Supplementary Files/**Additional File 1**).

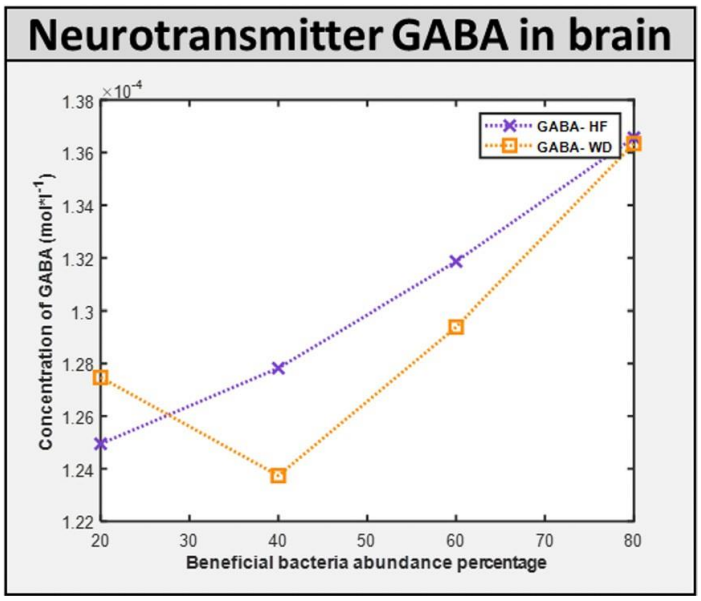


Figure A.9. ROS induced level of neurotransmitter in the brain.

GABA concentration in the brain with different diets, HF and WD, and increased level of probiotic intervention in the corresponding gut model.

A.4.6. Bacterial secretion products

The importance of core biochemical mechanisms driving autism must be understood, and metabolic biomarkers or bacterial secretion products in the gut can be used to assess them. The importance of an autistic-specific metabotype, *i.e.*, secretion profile of a group of patients with autism group was realized *via* recent metabolomics studies [186], [221] and other screening tests [166]. We predicted a total of 230 possible bacterial secretion products (Supplementary Files/**Table 3**). The top 15 secretion products, *i.e.*, the exchange reactions with the highest maximum fluxes in the FVA are given in Supplementary Files/Additional File/**Table S17**, ranked from high to low. The biomarker predictive capacity of the integrated model is moderate, as it captures some important bacterial secretion products implicated in autism, like propionic acid [193], [194], lactate, and pyruvate, which contribute to conditions linked to autism, such as mitochondrial dysfunction [222] and oxidative stress [222], as well as ammonia [171]. Although experimental validation of the remaining predicted bacterial secretion metabolites would be helpful, this is beyond the scope of this study.

We then analyzed the effect of the addition of probiotics on these potential biomarkers. The flux through harmful bacteria secretion is reduced as the fraction of beneficial bacteria is increased, as shown in **Table A.3** and Supplementary Files/**Table 4**. Typical examples include reduced flux through ammonia excretion. Interestingly, elevated levels of ammonia have been reported to be present in children with autism [171], [223].

Table A.3. Summary of the effect of beneficial bacteria and diet on secretion products of the harmful bacteria.

Rank	(a) Harmful bacteria products	(b) Effect of probiotic addition	(c) Change	(d) Effect of diet change
1	Formate	Formate	Significant decrease	Hydrogen
2	Hydrogen	Ammonium	Decrease	Carbon dioxide
3	Acetate	Acetate	Decrease	Hydrogen ions
4	Ammonium	Hydrogen ions	Increase	Acetate
5	Guanine	Guanine	Decrease	Succinate
6	Carbon dioxide	Carbon dioxide	Increase	Pyruvate
7	Hydrogen ions	Hydrogen	Increase	Amino acetaldehyde
8	Adenine	Xanthine	Increase	Lactate
9	Succinate	AMP	Output to input	Formate
10	Propionate	Glycerol 3-phosphate	Output to input	Malate
11	Lactate	Hydrogen phosphate	Output to input	Fumarate
12	Ethanol	Pyruvate	Increase	Ammonia
13	Aminoacetaldehyde	L-alanine	Decrease	Propionate
14	Pyruvate	Amino acetaldehyde	Increase	2-Oxobutanoate
15	Malate	Adenine	Decrease	Ethanol

Only the top 15 bacterial secretion products are shown. (a) Harmful bacterial secretion products, (b) Harmful bacterial secretion products that are affected by the addition of beneficial bacteria, (c) change observed upon the addition of corresponding beneficial bacteria, (d) increased secretion products after changing the diet from Western to high-fiber.

Furthermore, given these potential biomarkers, we analyzed their modulation *via* different diets. With the Western diet, the beneficial bacteria secreted 20 metabolites, 65% of which were foreign. With the high-fiber diet, beneficial bacteria secretion increased to 25 metabolites, 68% of which were foreign (Supplementary Files/**Table 4**). This is indicative of probiotics being relatively more effective in the high-fiber diet than in the Western diet.

With the Western diet, the harmful bacteria secreted 45 metabolites, 76% of which were foreign to the gut lumen. The beneficial bacteria secreted 20 metabolites, 65% of which were foreign. With the high-fiber diet, the harmful bacteria secreted 48 metabolites, 79% of which were foreign to the gut lumen. The beneficial bacteria secreted 25 metabolites, 68% of which were foreign. These are indicative of probiotics being relatively more effective in the high-fiber diet than in the Western diet. Since beneficial and harmful bacteria secrete common metabolites (Supplementary Files/**Table 3**), net bacterial secretion products in the gut microbiome model, with the Western diet, were found to be 45, 78% of which were foreign. With the high-fiber diet, net secretion was found to be 48 metabolites, 79% of which were foreign. These also show that bacterial secretion products themselves are dependent on the diet and not just their reaction fluxes. The top 15 bacterial secretion products for the harmful bacteria, *i.e.*, exchange reactions with the highest maximum fluxes in the FVA, are given in Supplementary Files/Additional File/**Table S9**, ranked from high to low. The entire list, along with their maximum secretion fluxes used to determine the ranking, is given in Supplementary Files/**Table 5**.

A.4.7. Effect of gut microbiome and diet on the gut metabolism

The effect of the addition of harmful gut bacteria on gut metabolism is determined by comparing the fluxes of sIEC reactions of the models gut microbiome–harmful and

sIEC by FVA as before. Similarly, to determine the effect of the addition of beneficial bacteria to this system, we compare the gut microbiome-harmful and gut microbiome models. We compare the gut microbiome using Western and high-fiber diets to analyze the effect of diet. The reactions, ordered as per the extent to which they are affected, are given in Supplementary Files/**Table 6**.

The number of reactions affected in each of the comparisons is summarized in Supplementary Files/Additional File 1/**Table S18**.

A.4.8. Effect of gut bacteria, diet, and reactive species metabolites on the metabolism of gut and brain: Metabolic Pathway Analysis

We investigated if oxidative stress is reduced with the administration of probiotics and observed that steady-state SOX levels in the brain and gut are reduced with both diets, with the reduction in SOX larger with the high-fiber diet than with the Western diet, confirming our hypothesis that the Western diet is more harmful to autism pathogenesis, as shown in **Figure A.10** and Supplementary Files/Additional File/**Table S6**. The effect of varying percentages of beneficial bacteria on ROS levels is presented in Supplementary Files/**Table 7**. Consistent with the previously observed effect of the addition of probiotics, we observed that increasing the percentage of beneficial bacteria decreases the steady-state concentration of superoxide in the gut and brain. As indicated in Supplementary Files/Additional File 1/**Table S18**, several pathways affected by the addition of beneficial bacteria, *i.e.*, pathways that are most affected by gut imbalance seen in autism, overlap with pathways affected by oxidative stress.

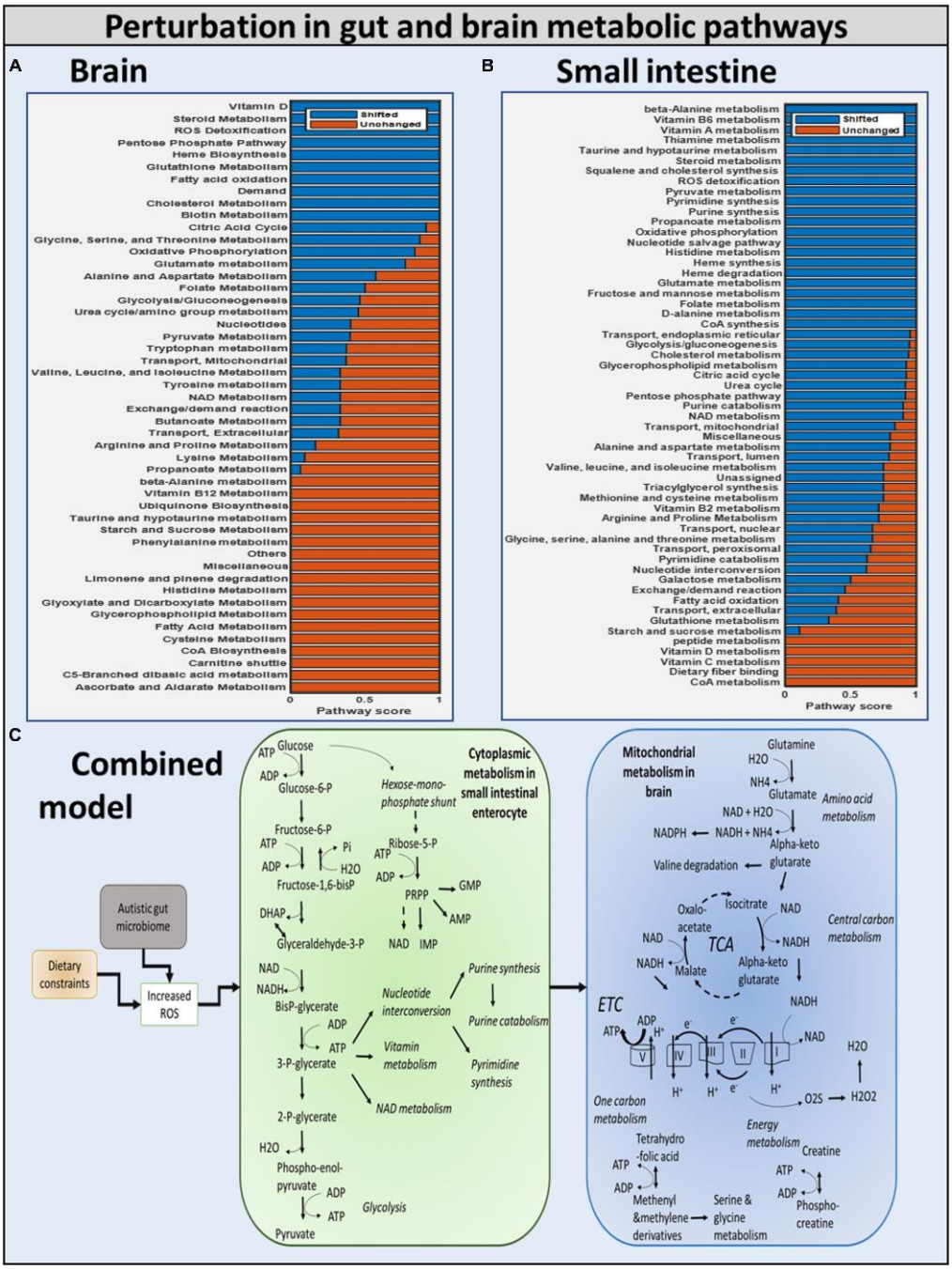


Figure A.10. Summary of the modulation of metabolic pathways in autism.

(A) Brain model: brain metabolic pathways perturbed due to dietary intervention. (B) Small intestinal model: SIEC metabolic pathways perturbed. (C) Combined model: the overall shift in metabolism of the brain and SIEC host.

Purine catabolism and nucleotide interconversion are among the most affected pathways in autism pathogenesis (Supplementary Files/Additional File 1/**Table S18**). Consequently, several experimental studies have indicated the association of nucleotide metabolism with autism. Additionally, upon inclusion of beneficial bacteria in the purely autistic gut model, higher flux through glycolysis was observed, hinting toward the potential of this being one of the protective mechanisms against oxidative stress.

To determine the effect of oxidative stress on metabolic pathways affected in autism, we developed and compared the "toxic model" and the "non-toxic model" for both the gut and the brain, with the former allowing high exchanges of oxidative stress-related toxins across membranes because of increased permeability observed in autistic conditions [224], [225], and the latter corresponding to low toxin exchange. The pathways affected by oxidative stress are given in Supplementary Files/**Table 7**.

A.5. DISCUSSION

This is the first time that a hybrid modeling technique for ASD combining steady-state and dynamic modeling techniques has been presented. The integrated model closely resembles known metabolic abnormalities in the gut and brain of people with autism. A novel dietary regimen for the treatment of autism symptoms is also recommended. The basic problem with merging constraint-based and PBPK models is that the former predicts reaction fluxes (rates), while the latter calculates reaction concentrations of the metabolites of interest. This was resolved by employing the static optimization strategy of dynamic Flux balance analysis dFBA [226], which assumes that a reaction consumes the whole concentration of a metabolite in each time step; consequently, the maximum rate served as input reaction boundaries. Simulating multicellular systems is difficult for a variety of reasons. As a result, Pareto-optimization was devised and used to find a solution with non-

zero biomass fluxes for all cells (where possible), allowing the system to grow collectively. Important findings on the impact of diet and gut bacteria on autism include: (i) a high-fiber diet reduces the growth of harmful bacteria, lowering the amount of toxins produced in the gut; and (ii) probiotic intervention with beneficial gut bacteria lowers the level of toxins in the autistic gut and brain.

Metabolic interactions drive microbial communities. Modeling and understanding microbial metabolism can disclose vital knowledge of how microorganisms form communities, interact, and can be employed in biotechnological processes as a result. Several methods for modeling microbial metabolism have been developed in recent decades, and some of them are being extended to represent microbial communities [227]. While there are alternative approaches to modeling microbial metabolism, the constraint-based modeling paradigm is, by far, the most used. Constraint-based modeling is a method for simulating any biological system, be it a single organism or a collection of species [228].

We chose five and ten bacteria that have been linked to the autistic gut in previous studies, and a genome-scale metabolic reconstruction is also provided [168], [171], [178]–[181], [183], [185], [186]. To this end, we used the metabolic network of Desulfovibrio desulfuricans subsp. desulfuricans, Clostridium perfringens ATCC 13124, Lactobacillus acidophilus ATCC 4796, Bifidobacterium longum longum JDM301, Akkermansia muciniphila ATCC BAA 835, Clostridium difficile NAP07, Prevotella ruminicola 23, Ruminococcus torques ATCC 27756, and Shigella flexneri 2002017.

In systems biology, it is common to put together a consortium of disease-causing bacteria to better understand how they interact with one another and with the host. However, the growing demand for metabolic models of various bacteria necessitates the use of the methodology provided with a consortium of 776 gut microorganisms to make

the forecast more realistic [204], [205]. However, those enormous-scale models with large-size models (millions of metabolic reactions from different microorganisms and hosts) will necessitate the most up-to-date big data approaches to handle the large amounts of data and extract insights from the results.

A.5.1. Importance of the high-fiber diet in reducing autism symptoms of patients with autism spectrum disorder compared to the western diet

Because gut bacteria metabolize most of the fiber in the body, the proliferation of all bacteria was shown to be generally higher with the high-fiber diet than with the Western diet (**Table A.2B**; [229]). As a result, the high-fiber diet encourages the growth of healthy gut bacteria. When compared to the Western diet, the high-fiber diet causes more bacterial secretion products to be produced (**Table A.3**). As previously stated, gut bacteria are known to digest the majority of fiber in our diets, promoting bacterial development and resulting in more bacterial secretion products. We can deduce that the high-fiber diet is more effective than the Western diet in reducing autism symptoms.

A.5.2. Significance of beneficial bacteria intervention

Upon increasing the fraction of beneficial bacteria in the gut, which is equivalent to administering probiotics to an individual with autism, we observed a monotonic decrease in the growth rate of all harmful bacteria with both diets, and a monotonic reduction in the levels of SOX in both the brain and the gut (**Figure A.7**). However, the production of H_2O_2 is reduced, where the combination of a high-fiber diet and probiotic works significantly well in reducing the level of H_2O_2 until the percentage abundance of beneficial bacteria reaches 40%. Increasing further the abundance of beneficial bacteria increases the level of H_2O_2 produced, the reason being some of the metabolic pathways synthesizing SOX *to* H_2O_2 carry an increased flux in the model at a higher percentage of probiotic intervention,

and one of the important enzymatic reactions synthesizing SOX is superoxide dismutase, which is also reported to be significantly lower in children with autism [218]. Superoxide dismutase is a potent protective enzyme that can selectively scavenge the radical O−12 by catalyzing its dismutation to H_2O_2:

$$2\ H + 2\ O_2S \rightarrow\ O_2 +\ H_2O_2 \qquad (A.9)$$

The flux through the superoxide dismutase in the host SIEC model increases with an increase in probiotic composition, thereby increasing the level of H_2O_2 toxin in the host model. This is carefully analyzed and documented in the five-microbe_community/Supplementary Files/**Table 7.xlsx**. Furthermore, the flux through other pathways associated with ASD condition, such as thiobarbituric acid-reactive substances (TBARS), superoxide dismutase (SOD), catalase (CAT) xanthine oxidase (XO), and adenosine deaminase (ADA) affected in the model with the change in the probiotics composition in the community model can be followed in Supplementary Files/**Additional_file_8.xlsx**.

A.5.3. Brain Adenosine triphosphate and neurotransmitter levels

Altered neurotransmissions in GABAergic neurons, including loss of GABA interneurons, have been identified in a subset of autism cases [162], [180], [208]. Hence, we next simulated the effect of probiotics on brain metabolism. Maintenance flux for each of the neuronal cell types in the brain model was obtained (Supplementary Files/Additional File/**Table S16**). Additionally, the model indicated that the flux through the synthesis reactions of GABA, an inhibitory neurotransmitter, was increased upon intake of beneficial bacteria (Supplementary Files/**Table 5**). Hence, we propose that probiotics may improve autism, majorly *via* increasing the availability of GABA in the brain.

They investigated the evidence related to the hypothesis that ASD is characterized by dysfunction in the GABA signaling mechanism. Furthermore, they attempted to explain how the abnormality theoretically manifests in autism symptoms by performing genetic and *in vivo* studies. The results produced by the integrated model also show marked deviation in GABA levels in the brain of individuals with "purely autistic" gut when compared with the "corrected" gut, thus concurring with the study of Coghlan [208].

A.5.4. Modeling oxidative stress is important for understanding fundamental metabolic mechanisms that cause autism

Typically, the production of ROS, H_2O_2, and SOX [192] is associated with autism [194]. To determine the effect of oxidative stress on metabolic pathways affected in autism, we developed and compared the "toxic" model and the "non-toxic" model for both the gut and the brain, with the former allowing for high exchanges of oxidative stress-related toxins across membranes because of increased permeability observed under autistic conditions [186], [230], [231], and the latter corresponding to low toxin exchange.

The toxin concentration gradually increased during the simulation until it reached saturation (**Figure A.8**). This occurred because the simulation starts with initial conditions chosen for an ideal (or desired) case of very low toxin concentrations in the body, which would then naturally increase with the activation of metabolic pathways, producing ROS in the gut and the brain, reaching saturation levels where the net intake and production of ROS from the body balanced the net consumption and excretion of ROS from the body. The H_2O_2 toxin profile in organs is reduced to a 40% beneficial percentage in the presence of a high fiber diet. The H_2O_2 toxin increases steadily with the Western diet. This highlights that a high-fiber diet and an optimum number of probiotics are important to reduce the toxins produced in autism.

First, upon increasing the fraction of beneficial bacteria in the gut, which is equivalent to administering probiotics to an individual with autism, we observed a monotonic decrease in the growth rate of all harmful bacteria with both diets, and a monotonic reduction in the levels of oxidative stress in both the brain and the gut (**Figure A.11**). The correlation between these two phenomena is further highlighted by the presence of glutamate metabolism and ROS detoxification among the most affected pathways in the brain [232]; **Table A.4**).

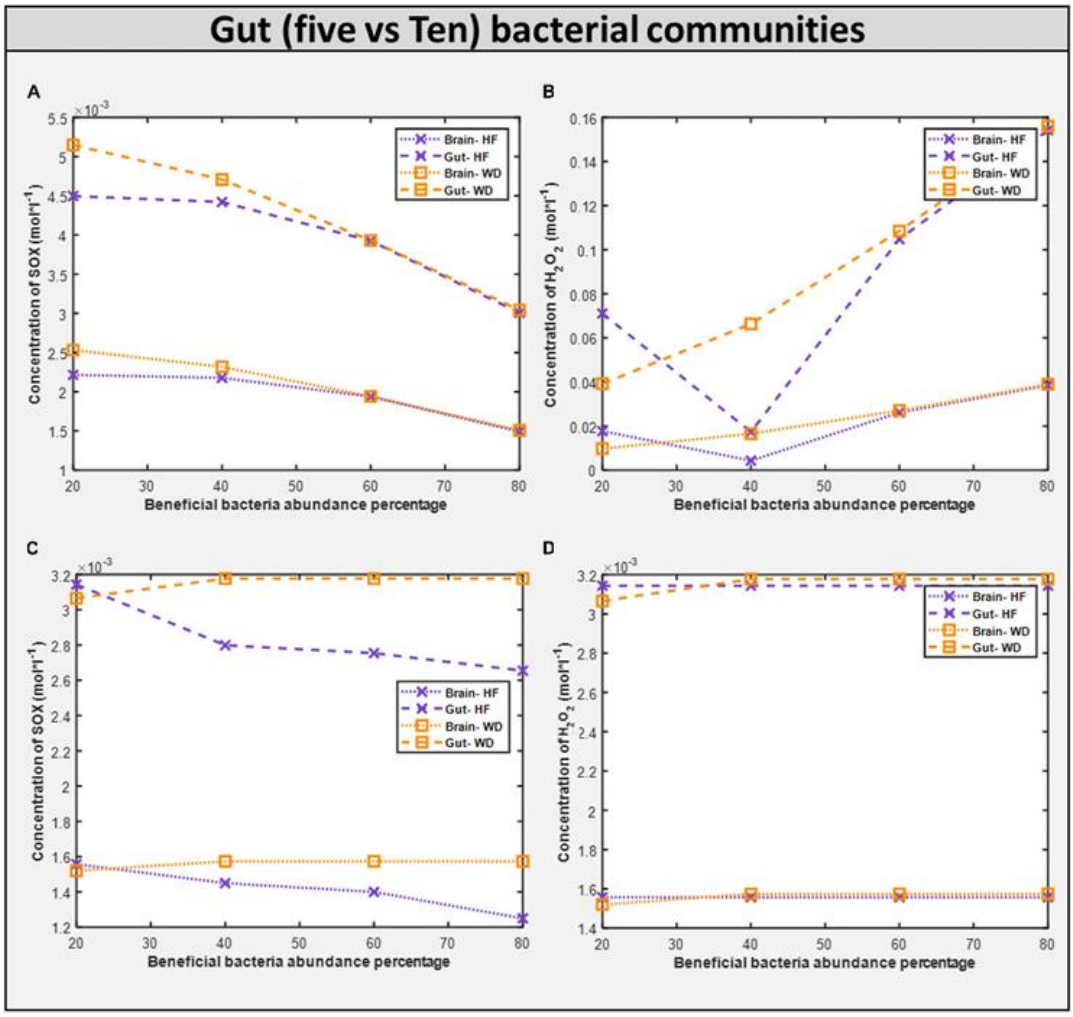


Figure A.11. Comparison of toxins (H_2O_2 and O_2S) in five vs ten bacterial gut representative model and toxins in corresponding brain model.

(A) SOX concentration generated in the five microbial community. (B) H_2O_2 concentration generated in the five microbial community. (C) SOX concentration generated in the ten microbial community. (D) H_2O_2 concentration generated in the five microbial community.

Table A.4. Top 10 gut metabolism pathways affected by harmful bacteria subsequent addition of beneficial bacteria, change in diet, and effect of oxidative stress on the gut and brain.

S. No.	Pathways affected by the addition of beneficial bacteria to "purely" autistic gut	Pathways affected by the change in diet	Pathways affected in the gut	Pathways affected in the brain
1	Purine catabolism	Purine catabolism	Purine catabolism	ROS detoxification
2	Nucleotide interconversion	Glycolysis/ gluconeogenesis	Citric acid cycle	Oxidative phosphorylation
3	Glycolysis/ gluconeogenesis	Fructose and mannose metabolism	Alanine and aspartate metabolism	Pyruvate metabolism
4	Citric acid cycle	Citric acid cycle	Purine synthesis	Glutamate metabolism
5	Valine, leucine, and isoleucine metabolism	Pyruvate metabolism	Vitamin B6 metabolism	NAD metabolism
6	Propanoate metabolism	Nucleotide interconversion	NAD metabolism	Citric acid cycle
7	Pyruvate metabolism	NAD metabolism	Pyrimidine catabolism	Urea cycle/amino group metabolism
8	Vitamin B6 metabolism	Purine synthesis	Glycolysis/ gluconeogenesis	Alanine and aspartate metabolism
9	NAD metabolism	Vitamin B6 metabolism	Arginine and proline metabolism	Valine, leucine, and isoleucine metabolism
10	Purine synthesis	Pyrimidine catabolism	Nucleotide interconversion	Folate metabolism

A.5.5. Autistic biomarkers are revealed by metabolic perturbations in gut and brain models

Diet and gut microorganisms have been found to have a significant impact on host metabolism, primarily through central carbon, nucleotide, and vitamin metabolism in the gut, and mitochondrial energy, amino acid, and metabolism in the brain (**Figure A.10**). While cytoplasmic pathways were discovered to be severely impacted in the small intestine enterocyte model, mitochondrial metabolism was found to be disrupted in the brain. Because ATP is involved in glycolysis, nucleotide, and vitamin metabolism, these pathways were discovered to be not only interconnected but also to be the primary players in mediating small intestinal metabolism in the autistic instance. In the brain, glutamate and glutamine metabolism was discovered to be interconnected with downstream processes, such as energy generation *via* the TCA, ETC, and phosphocreatine pathways, as well as one-carbon metabolism. As a result, our model revealed a greater flux through glutamate metabolic reactions in the brain. Autism has been linked to high glutamate levels and a malfunctioning mitochondrial glutamate transporter [171], [221]. **Figure A.10** shows a summary of the numerous metabolic impacts. Finally, the model was utilized to demonstrate that by including probiotics in one's diet, altered intestine and brain metabolism can be managed to alleviate autism symptoms. Reduced ROS levels in the gut and increased GABA levels in the brain led to this conclusion.

Studies have associated abnormal amino acid, fatty acid, energy, and reactive species metabolism [168], [169] with autism. Herein, we report that the purine catabolism and nucleotide interconversion pathways are among the most affected pathways in autism pathogenesis when simulated with the Western diet (Supplementary Files/Additional File/**Table S10**). Consequently, elevated purine catabolism *via* xanthine oxidase (GeneID: 7498, *XDH*, E.C. 1.17.3.2.) was reported in patients with autism [218]. Further, increased

uric acid levels in children with autism, and anti-purinergic therapy has been shown to reduce autistic features in mice [233].

Pyruvate and lactate, both known biomarkers of mitochondrial dysfunction, are present at abnormally high levels in individuals with autism [222]. The integrated model was also in consonance with this evidence from the literature, wherein metabolic pathways of pyruvate and lactate metabolism were among the most affected in models of autism (Supplementary Files/Additional File/**Tables S14, S15**). An interesting observation shown in Supplementary Files/Additional File/**Table S17** is that among the top three secretion products, the Bacteroides vulgatus model contributed the maximum amount to the total formate and acetate generated in the autistic gut (Supplementary Files/Additional File/**Table S17**). This makes sense since Bacteroides vulgatus decomposes complex sugars and produce short-chain fatty acids [234]. On the other hand, the Clostridium perfringens model generated the maximum amount of H_2. Thus, *Bacteroides vulgatus* and *Clostridium perfringens* can be the most severe drivers of autism among all the implicated harmful bacteria, and the significant reduction in the levels of formate in the gut upon addition of beneficial bacteria (Supplementary Files/Additional File/**Table S17**) further strengthen this theory. We propose that the development of novel drugs and dietary formulations focusing on reducing the quantities of these two bacteria can show a huge improvement in autism symptoms. Simultaneously, we propose novel biomarkers for autism (Supplementary Files/Additional File/**Table S17**), warranting clinical investigations.

A.5.6. The benefit of increasing gut microbiota community to ten bacteria

The large-scale dynamics of the microbiome can be used to better understand population ecology and microbial community characteristics [162]. Increasing the size of the model allows us to better understand the behavior of our new framework and

demonstrate its durability in the face of a more complicated and extended flux-based model. We used two different-sized microbial community models. An extended model was utilized to see if the conclusions from the five-microbe community model on metabolic pathways, oxidative stress, secretory profile, and treatment alternatives are still valid.

The community is carefully observed, and it was found that the microbe *Desulfovibrio desulfuricans* changed its phenotype behavior because of changes in its interacting partners. Interestingly, an abundance of *Desulfovibrio desulfuricans* in the stool of patients with autism has been reported, and the change in abundance is positive [235], *i.e.*, patients with autism have an increased abundance of *Desulfovibrio desulfuricans* in the gut, which means that it should not be taken as beneficial bacteria but instead as harmful bacteria. Therefore, when increasing the beneficial bacteria, we are also increasing *Desulfovibrio* abundance. Hence, the effect from the beneficial bacteria and *Desulfovibrio* nullifies the total beneficial effect. This demonstrates the resilience of the model and the importance of selecting the proper probiotic.

Dietary intervention in autism is crucial for both the GI and autistic systems to improve [236]. Steady-state SOX concentrations were observed to be lower with the high-fiber diet than with the Western diet according to the five-microbe community model. In the presence of a high fiber diet, the H_2O_2 toxin profile in organs is lowered to a 40 percent favorable percentage. Like in the five-microbe community model, steady-state SOX concentrations were found to be lower for the high-fiber diet than for the Western diet. This emphasizes the importance of a high-fiber diet, as well as proper dosage and selection of probiotics, in reducing the toxins produced in autism.

The growth rate of bacteria in the ten gut-microbial community is higher with the high-fiber diet than with the Western diet (Supplementary Files/Additional File/**Table S4**), confirming our findings from five-microbe community studies [229]. Upon increasing the

fraction of beneficial bacteria in the gut, which is equivalent to administering probiotics to an individual with autism, we observed a monotonic decrease in the growth rate of all harmful bacteria with both diets like in the five-microbe community, and a monotonic increase in the levels of *SOX* in both the brain and the gut unlike in the case of the five-microbe community. Furthermore, the flux through which other pathways associated with the ASD condition, such as thiobarbituric acid-reactive substances (TBARS), superoxide dismutase (SOD), catalase (CAT), xanthine oxidase (XO), and adenosine deaminase (ADA), affected the model with change in probiotic composition in the community model can be found in Supplementary Files/**Table 7**. In addition to the purine catabolism and nucleotide interconversion pathways that were found to be affected in the five-microbe community model, we also found glutathione metabolism to be among the most affected pathways in autism pathogenesis [237](Supplementary Files/Additional File/**Table S8**).

A.6. CONCLUSION

The novel *in silico* approach used in this study demonstrates how toxins produced by gut dysbiosis alter the normal function of the brain in ASD. Integrating the steady-state and dynamic techniques (*i.e.*, COBRA and PBPK, respectively) has enabled the development of the novel hybrid model, a whole-body transport system with tissue-specific metabolic details. Tissue-specific models revealed metabolic disruption in both the gut and the brain. Most importantly, we showed the integral connection between diet and gut microbes that majorly influences host metabolism, typically *via* central carbon, nucleotide, and vitamin metabolism in the gut, and mitochondrial energy, amino acid, and metabolism in the brain. This approach helped in gaining better insights into the gut-brain axis in ASD. By leveraging the varying compositions of gut bacteria models, *i.e.*, five-bacteria gut model and ten-bacteria gut model, we found that *Bacteroides vulgatus* and *Clostridium*

perfringens contribute maximally to total formate and acetate generated in the autistic gut. Hence, these were identified to be the severe drivers of autism among all the implicated harmful bacteria. In response to dietary intervention, a high-fiber diet combined with beneficial bacteria, *i.e.*, *Lactobacillus acidophilus* and *Bifidobacterium longum*, is proposed to help alleviate autism symptoms. The unique paradigm we provide here can, thus, be used for metabolic and metabolomic investigations on diseases characterized by disturbed microbial makeup in the gut.

A.7. DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

A.8. AUTHOR CONTRIBUTIONS

FM, MP, and SSh built the models, analyzed the data, and interpreted the results. FM and MP developed the algorithms and analysis tools. RR monitored the PBPK modeling. SS was responsible for the COBRA modeling parts, conceived and designed the study, and planned the experiments. All authors wrote the manuscript.

A.9. FUNDING

This work was made possible by an INSPIRE faculty award from the Department of Science and Technology of India (DST/INSPIRE/04/2015/000036) to SS.

A.10. SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphys.2022.760753/full#supplementary-material

A.11. ABBREVIATIONS

- COBRA: constraint-based regression and analysis;
- PBPK: physiologically based pharmacokinetic model;
- ASD: autism spectrum disorder;
- GABA: gamma-aminobutyric acid;
- FVA: flux variability analysis;
- SIECs: small intestinal enterocyte cells;
- ROS: reactive oxygen species;
- *SOX*: superoxide;
- H_2O_2: Hydrogen peroxide;
- AGORA: assembly of the gut organism through reconstruction and analysis;
- QSPR: quantitative structure-property relationship;
- ATP: adenosine triphosphate;
- BV: *Bacteroides vulgatus* ATCC8482;
- DD: *Desulfovibrio desulfuricans subsp. desulfuricans* DSM642;
- CL: *Clostridium perfringens* ATCC13124;
- LA: *Lactobacillus acidophilus* ATCC4796;
- BL: *Bifidobacterium longum longum* JDM301;
- AKK: *Akkermansia muciniphila* ATCCBAA835;
- CL: *Clostridium difficile* NAP07;

- PR: *Prevotella ruminicola* 23;
- RT: *Ruminococcus torques* ATCC27756;
- SH: *Shigella flexneri* 2002017.

References

[1] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, Feb. 2010, doi: 10.1016/j.physrep.2009.11.002.

[2] Y. Qi, F. Balem, C. Faloutsos, J. Klein-Seetharaman, and Z. Bar-Joseph, "Protein complex identification by supervised graph local clustering," *Bioinformatics*, vol. 24, no. 13, pp. i250–i268, Jul. 2008, doi: 10.1093/bioinformatics/btn164.

[3] C. Lee, F. Reid, A. McDaid, and N. Hurley, "Detecting highly overlapping community structure by greedy clique expansion," *arXiv:1002.1827 [physics]*, Jun. 2010, Accessed: Oct. 28, 2020. [Online]. Available: http://arxiv.org/abs/1002.1827

[4] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek, "CFinder: locating cliques and overlapping modules in biological networks," *Bioinformatics*, vol. 22, no. 8, pp. 1021–1023, Apr. 2006, doi: 10.1093/bioinformatics/btl039.

[5] G. D. Bader and C. W. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, no. 1, p. 2, Jan. 2003, doi: 10.1186/1471-2105-4-2.

[6] S. Dongen, "Graph Clustering by Flow Simulation," *PhD thesis, Center for Math and Computer Science (CWI)*, May 2000.

[7] G. Liu, L. Wong, and H. N. Chua, "Complex discovery from weighted PPI networks," *Bioinformatics*, vol. 25, no. 15, pp. 1891–1897, Aug. 2009, doi: 10.1093/bioinformatics/btp311.

[8] M. Mete, F. Tang, X. Xu, and N. Yuruk, "A structural approach for finding functional modules from large biological networks," *BMC Bioinformatics*, vol. 9, no. Suppl 9, p. S19, Aug. 2008, doi: 10.1186/1471-2105-9-S9-S19.

[9] T. Nepusz, H. Yu, and A. Paccanaro, "Detecting overlapping protein complexes in protein-protein interaction networks," *Nat. Methods*, vol. 9, no. 5, pp. 471–472, Mar. 2012, doi: 10.1038/nmeth.1938.

[10] J. Wu and M. Lin, "Protein Complex Detection Based on Semi-Supervised Matrix Factorization," in *2018 37th Chinese Control Conference (CCC)*, Wuhan, Jul. 2018, pp. 8205–8208. doi: 10.23919/ChiCC.2018.8484055.

[11] "Biochemistry, 4th Edition | Wiley," *Wiley.com*. https://www.wiley.com/en-us/Biochemistry%2C+4th+Edition-p-9780470570951 (accessed Nov. 13, 2020).

[12] "PDB101: Goodsell Gallery: Escherichia coli," *RCSB: PDB-101*. https://pdb101.rcsb.org/sci-art/goodsell-gallery/escherichia-coli (accessed Dec. 10, 2020).

[13] A. E. Medlock *et al.*, "Identification of the Mitochondrial Heme Metabolism Complex," *PLOS ONE*, vol. 10, no. 8, p. e0135896, Aug. 2015, doi: 10.1371/journal.pone.0135896.

[14] L. Shi, X. Lei, and A. Zhang, "Protein complex detection with semi-supervised learning in protein interaction networks," *Proteome Sci*, vol. 9, no. Suppl 1, p. S5, 2011, doi: 10.1186/1477-5956-9-S1-S5.

[15] F. Yu, Z. Yang, N. Tang, H. Lin, J. Wang, and Z. Yang, "Predicting protein complex in protein interaction network - a supervised learning based method," *BMC Syst Biol*, vol. 8, no. Suppl 3, p. S4, 2014, doi: 10.1186/1752-0509-8-S3-S4.

[16] Y. Dong, Y. Sun, and C. Qin, “Predicting protein complexes using a supervised learning method combined with local structural information,” *PLoS ONE*, vol. 13, no. 3, p. e0194124, Mar. 2018, doi: 10.1371/journal.pone.0194124.

[17] A. Sikandar *et al.*, “Decision Tree Based Approaches for Detecting Protein Complex in Protein Protein Interaction Network (PPI) via Link and Sequence Analysis,” *IEEE Access*, vol. 6, pp. 22108–22120, 2018, doi: 10.1109/ACCESS.2018.2807811.

[18] Y. Qi, F. Balem, C. Faloutsos, J. Klein-Seetharaman, and Z. Bar-Joseph, “Protein complex identification by supervised graph local clustering,” *Bioinformatics*, vol. 24, no. 13, pp. i250–i268, Jul. 2008, doi: 10.1093/bioinformatics/btn164.

[19] B. C. Borgeson, “All-by-all discovery of conserved protein complexes by deep proteome fractionation,” Thesis, 2016. doi: 10.15781/T2CZ3299G.

[20] R. M. Karp, “An algorithm to solve the m × n assignment problem in expected time O(mn log n),” *Networks*, vol. 10, no. 2, pp. 143–152, 1980, doi: https://doi.org/10.1002/net.3230100205.

[21] S. Brohée and J. van Helden, “Evaluation of clustering algorithms for protein-protein interaction networks,” *BMC Bioinformatics*, vol. 7, no. 1, p. 488, Dec. 2006, doi: 10.1186/1471-2105-7-488.

[22] K. Drew *et al.*, “Integration of over 9,000 mass spectrometry experiments builds a global map of human protein complexes,” *Mol. Syst. Biol.*, vol. 13, no. 6, p. 932, 08 2017, doi: 10.15252/msb.20167490.

[23] M. Wu, X. Li, C.-K. Kwoh, and S.-K. Ng, “A core-attachment based method to detect protein complexes in PPI networks,” *BMC Bioinformatics*, vol. 10, no. 1, p. 169, Jun. 2009, doi: 10.1186/1471-2105-10-169.

[24] Randy Olson *et al.*, *EpistasisLab/tpot: v0.10.1 minor release*. Zenodo, 2019. doi: 10.5281/zenodo.2647523.

[25] M. Palukuri and E. Marcotte, “Supervised Community Detection in Protein-interaction Networks,” *TACCSTER 2019 Proceedings*, 2019, doi: 10.26153/tsw/6852.

[26] L. A. Wilson and J. M. Fonner, “Launcher: A Shell-based Framework for Rapid Development of Parallel Parametric Studies,” in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, New York, NY, USA, Jul. 2014, pp. 1–8. doi: 10.1145/2616498.2616534.

[27] V. A. Traag, L. Waltman, and N. J. van Eck, “From Louvain to Leiden: guaranteeing well-connected communities,” *Scientific Reports*, vol. 9, no. 1, Art. no. 1, Mar. 2019, doi: 10.1038/s41598-019-41695-z.

[28] M. Giurgiu *et al.*, “CORUM: the comprehensive resource of mammalian protein complexes-2019,” *Nucleic Acids Res.*, vol. 47, no. D1, pp. D559–D563, Jan. 2019, doi: 10.1093/nar/gky973.

[29] I. Xenarios, L. Salwínski, X. J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg, “DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions,” *Nucleic Acids Res.*, vol. 30, no. 1, pp. 303–305, Jan. 2002, doi: 10.1093/nar/30.1.303.

[30] H. W. Mewes *et al.*, “MIPS: analysis and annotation of proteins from whole genomes,” *Nucleic Acids Res.*, vol. 32, no. Database issue, pp. D41-44, Jan. 2004, doi: 10.1093/nar/gkh092.
[31] A.-C. Gavin *et al.*, “Proteome survey reveals modularity of the yeast cell machinery,” *Nature*, vol. 440, no. 7084, pp. 631–636, Mar. 2006, doi: 10.1038/nature04532.
[32] D. E. Gordon *et al.*, “A SARS-CoV-2 protein interaction map reveals targets for drug repurposing,” *Nature*, vol. 583, no. 7816, Art. no. 7816, Jul. 2020, doi: 10.1038/s41586-020-2286-9.
[33] The UniProt Consortium, “UniProt: the universal protein knowledgebase in 2021,” *Nucleic Acids Research*, vol. 49, no. D1, pp. D480–D489, Jan. 2021, doi: 10.1093/nar/gkaa1100.
[34] P. J. Thul *et al.*, “A subcellular map of the human proteome,” *Science*, vol. 356, no. 6340, May 2017, doi: 10.1126/science.aal3321.
[35] M. Wainberg *et al.*, “A genome-wide atlas of co-essential modules assigns function to uncharacterized genes,” *Nature Genetics*, pp. 1–12, Apr. 2021, doi: 10.1038/s41588-021-00840-z.
[36] H. Li *et al.*, “Identifying gene function and module connections by the integration of multispecies expression compendia,” *Genome Res.*, vol. 29, no. 12, pp. 2034–2045, Dec. 2019, doi: 10.1101/gr.251983.119.
[37] J. Song and M. Singh, “How and when should interactome-derived clusters be used to predict functional modules and protein function?,” *Bioinformatics*, vol. 25, no. 23, pp. 3143–3150, Dec. 2009, doi: 10.1093/bioinformatics/btp551.
[38] T. T. Le, W. Fu, and J. H. Moore, “Scaling tree-based automated machine learning to biomedical big data with a feature set selector,” *Bioinformatics*, vol. 36, no. 1, pp. 250–256, Jan. 2020, doi: 10.1093/bioinformatics/btz470.
[39] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, “Automating biomedical data science through tree-based pipeline optimization.” arXiv, Jan. 28, 2016. doi: 10.48550/arXiv.1601.07925.
[40] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, “Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science,” in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, New York, NY, USA, Jul. 2016, pp. 485–492. doi: 10.1145/2908812.2908918.
[41] “Proceedings of the Python in Science Conference (SciPy): Exploring Network Structure, Dynamics, and Function using NetworkX.” https://conference.scipy.org/proceedings/SciPy2008/paper_2/ (accessed Jun. 01, 2022).
[42] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.
[43] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.” arXiv, Mar. 16, 2016. doi: 10.48550/arXiv.1603.04467.

[44] A. Ruepp *et al.*, "CORUM: the comprehensive resource of mammalian protein complexes," *Nucleic Acids Res*, vol. 36, no. Database issue, pp. D646–D650, Jan. 2008, doi: 10.1093/nar/gkm936.

[45] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv:1609.02907 [cs, stat]*, Feb. 2017, Accessed: Nov. 28, 2020. [Online]. Available: http://arxiv.org/abs/1609.02907

[46] A. Elnaggar *et al.*, "ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing." arXiv, May 04, 2021. doi: 10.48550/arXiv.2007.06225.

[47] F. Morcos *et al.*, "Direct-coupling analysis of residue coevolution captures native contacts across many protein families," *PNAS*, vol. 108, no. 49, pp. E1293–E1301, Dec. 2011, doi: 10.1073/pnas.1111471108.

[48] T. Rolland *et al.*, "A Proteome-Scale Map of the Human Interactome Network," *Cell*, vol. 159, no. 5, pp. 1212–1226, Nov. 2014, doi: 10.1016/j.cell.2014.10.050.

[49] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning Distributed Representations of Graphs," *arXiv:1707.05005 [cs]*, Jul. 2017, Accessed: Nov. 28, 2020. [Online]. Available: http://arxiv.org/abs/1707.05005

[50] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," *arXiv:1607.00653 [cs, stat]*, Jul. 2016, Accessed: Nov. 28, 2020. [Online]. Available: http://arxiv.org/abs/1607.00653

[51] S. Mei and H. Zhu, "A novel one-class SVM based negative data sampling method for reconstructing proteome-wide HTLV-human protein interaction networks," *Scientific Reports*, vol. 5, no. 1, Art. no. 1, Jan. 2015, doi: 10.1038/srep08034.

[52] A. L. Richards, M. Eckhardt, and N. J. Krogan, "Mass spectrometry-based protein–protein interaction networks for the study of human diseases," *Mol Syst Biol*, vol. 17, no. 1, p. e8792, Jan. 2021, doi: 10.15252/msb.20188792.

[53] K. Titeca, I. Lemmens, J. Tavernier, and S. Eyckerman, "Discovering cellular protein-protein interactions: Technological strategies and opportunities," *Mass Spectrom Rev*, vol. 38, no. 1, pp. 79–111, Jan. 2019, doi: 10.1002/mas.21574.

[54] A. H. Smits and M. Vermeulen, "Characterizing Protein-Protein Interactions Using Mass Spectrometry: Challenges and Opportunities," *Trends Biotechnol*, vol. 34, no. 10, pp. 825–834, Oct. 2016, doi: 10.1016/j.tibtech.2016.02.014.

[55] J. Snider, M. Kotlyar, P. Saraon, Z. Yao, I. Jurisica, and I. Stagljar, "Fundamentals of protein interaction network mapping," *Mol Syst Biol*, vol. 11, no. 12, p. 848, Dec. 2015, doi: 10.15252/msb.20156351.

[56] T. M. Cafarelli, A. Desbuleux, Y. Wang, S. G. Choi, D. De Ridder, and M. Vidal, "Mapping, modeling, and characterization of protein-protein interactions on a proteomic scale," *Curr Opin Struct Biol*, vol. 44, pp. 201–210, Jun. 2017, doi: 10.1016/j.sbi.2017.05.003.

[57] K. Drew, J. B. Wallingford, and E. M. Marcotte, "hu.MAP 2.0: integration of over 15,000 proteomic experiments builds a global compendium of human multiprotein

assemblies," *Mol Syst Biol*, vol. 17, no. 5, p. e10016, May 2021, doi: 10.15252/msb.202010016.
[58] A. Malovannaya *et al.*, "Analysis of the Human Endogenous Coregulator Complexome," *Cell*, vol. 145, no. 5, pp. 787–799, May 2011, doi: 10.1016/j.cell.2011.05.006.
[59] M. Y. Hein *et al.*, "A Human Interactome in Three Quantitative Dimensions Organized by Stoichiometries and Abundances," *Cell*, vol. 163, no. 3, pp. 712–723, Oct. 2015, doi: 10.1016/j.cell.2015.09.053.
[60] E. L. Huttlin *et al.*, "The BioPlex Network: A Systematic Exploration of the Human Interactome," *Cell*, vol. 162, no. 2, pp. 425–440, Jul. 2015, doi: 10.1016/j.cell.2015.06.043.
[61] E. L. Huttlin *et al.*, "Architecture of the human interactome defines protein communities and disease networks," *Nature*, vol. 545, no. 7655, Art. no. 7655, May 2017, doi: 10.1038/nature22366.
[62] C. Wan *et al.*, "Panorama of ancient metazoan macromolecular complexes," *Nature*, vol. 525, no. 7569, Art. no. 7569, Sep. 2015, doi: 10.1038/nature14877.
[63] K. J. Kirkwood, Y. Ahmad, M. Larance, and A. I. Lamond, "Characterization of Native Protein Complexes and Protein Isoform Variation Using Size-fractionation-based Quantitative Proteomics *," *Molecular & Cellular Proteomics*, vol. 12, no. 12, pp. 3851–3873, Dec. 2013, doi: 10.1074/mcp.M113.032367.
[64] A. R. Kristensen, J. Gsponer, and L. J. Foster, "A high-throughput approach for measuring temporal changes in the interactome," *Nat Methods*, vol. 9, no. 9, Art. no. 9, Sep. 2012, doi: 10.1038/nmeth.2131.
[65] P. C. Havugimana *et al.*, "A Census of Human Soluble Protein Complexes," *Cell*, vol. 150, no. 5, pp. 1068–1081, Aug. 2012, doi: 10.1016/j.cell.2012.08.011.
[66] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig, "Community detection in networks: A multidisciplinary review," *Journal of Network and Computer Applications*, vol. 108, pp. 87–111, Apr. 2018, doi: 10.1016/j.jnca.2018.02.011.
[67] G. D. Bader and C. W. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, no. 1, p. 2, Jan. 2003, doi: 10.1186/1471-2105-4-2.
[68] G. Liu, L. Wong, and H. N. Chua, "Complex discovery from weighted PPI networks," *Bioinformatics*, vol. 25, no. 15, pp. 1891–1897, Aug. 2009, doi: 10.1093/bioinformatics/btp311.
[69] M. Wu, X. Li, C.-K. Kwoh, and S.-K. Ng, "A core-attachment based method to detect protein complexes in PPI networks," *BMC Bioinformatics*, vol. 10, no. 1, p. 169, Jun. 2009, doi: 10.1186/1471-2105-10-169.
[70] T. Nepusz, H. Yu, and A. Paccanaro, "Detecting overlapping protein complexes in protein-protein interaction networks," *Nat Methods*, vol. 9, no. 5, Art. no. 5, May 2012, doi: 10.1038/nmeth.1938.
[71] C. Lee, F. Reid, A. McDaid, and N. Hurley, "Detecting highly overlapping community structure by greedy clique expansion," arXiv, arXiv:1002.1827, Jun. 2010. doi: 10.48550/arXiv.1002.1827.

[72] Y. Dong, Y. Sun, and C. Qin, “Predicting protein complexes using a supervised learning method combined with local structural information,” *PLOS ONE*, vol. 13, no. 3, p. e0194124, Mar. 2018, doi: 10.1371/journal.pone.0194124.
[73] M. V. Palukuri and E. M. Marcotte, “Super.Complex: A supervised machine learning pipeline for molecular complex detection in protein-interaction networks,” *PLOS ONE*, vol. 16, no. 12, p. e0262056, Dec. 2021, doi: 10.1371/journal.pone.0262056.
[74] E. C. Paim, A. L. C. Bazzan, and C. Chira, “Detecting Communities in Networks: a Decentralized Approach Based on Multiagent Reinforcement Learning,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2020, pp. 2225–2232. doi: 10.1109/SSCI47803.2020.9308197.
[75] J. D. Arroyo *et al.*, “A Genome-wide CRISPR Death Screen Identifies Genes Essential for Oxidative Phosphorylation,” *Cell Metabolism*, vol. 24, no. 6, pp. 875–885, Dec. 2016, doi: 10.1016/j.cmet.2016.08.017.
[76] R. L. Wolfson *et al.*, “KICSTOR recruits GATOR1 to the lysosome and is necessary for nutrients to regulate mTORC1,” *Nature*, vol. 543, no. 7645, Art. no. 7645, Mar. 2017, doi: 10.1038/nature21423.
[77] S. Suetsugu, H. Miki, and T. Takenawa, “Identification of two human WAVE/SCAR homologues as general actin regulatory molecules which associate with the Arp2/3 complex,” *Biochem Biophys Res Commun*, vol. 260, no. 1, pp. 296–302, Jun. 1999, doi: 10.1006/bbrc.1999.0894.
[78] O. D. Weiner *et al.*, “Hem-1 complexes are essential for Rac activation, actin polymerization, and myosin regulation during neutrophil chemotaxis,” *PLoS Biol*, vol. 4, no. 2, p. e38, Feb. 2006, doi: 10.1371/journal.pbio.0040038.
[79] N. H. Cho *et al.*, “OpenCell: Endogenous tagging for the cartography of human cellular organization,” *Science*, vol. 375, no. 6585, p. eabi6983, Mar. 2022, doi: 10.1126/science.abi6983.
[80] D. Wichtowska, T. W. Turowski, and M. Boguta, “An interplay between transcription, processing, and degradation determines tRNA levels in yeast,” *Wiley Interdiscip Rev RNA*, vol. 4, no. 6, pp. 709–722, Dec. 2013, doi: 10.1002/wrna.1190.
[81] G. Kustatscher *et al.*, “Understudied proteins: opportunities and challenges for functional proteomics,” *Nat Methods*, pp. 1–6, May 2022, doi: 10.1038/s41592-022-01454-x.
[82] “C4orf19 expression in human.” https://bgee.org/gene/ENSG00000154274 (accessed May 20, 2022).
[83] “Tissue expression of C4orf19 - Summary - The Human Protein Atlas.” https://www.proteinatlas.org/ENSG00000154274-C4orf19/tissue (accessed Jun. 16, 2022).
[84] W. Wang *et al.*, “Down-regulated C4orf19 confers poor prognosis in colon adenocarcinoma identified by gene co-expression network,” *J Cancer*, vol. 13, no. 4, pp. 1145–1159, Jan. 2022, doi: 10.7150/jca.63635.
[85] X. Zheng *et al.*, “CCM3 signaling through sterile 20-like kinases plays an essential role during zebrafish cardiovascular development and cerebral cavernous

malformations," *J Clin Invest*, vol. 120, no. 8, pp. 2795–2804, Aug. 2010, doi: 10.1172/JCI39679.

[86] M. Goudreault *et al.*, "A PP2A phosphatase high density interaction network identifies a novel striatin-interacting phosphatase and kinase complex linked to the cerebral cavernous malformation 3 (CCM3) protein," *Mol Cell Proteomics*, vol. 8, no. 1, pp. 157–171, Jan. 2009, doi: 10.1074/mcp.M800266-MCP200.

[87] R. Wang *et al.*, "Pdcd10-Stk24/25 complex controls kidney water reabsorption by regulating Aqp2 membrane targeting," *JCI Insight*, vol. 6, no. 12, p. 142838, Jun. 2021, doi: 10.1172/jci.insight.142838.

[88] M. Xiong *et al.*, "KIF20A promotes cellular malignant behavior and enhances resistance to chemotherapy in colorectal cancer through regulation of the JAK/STAT3 signaling pathway," *Aging*, vol. 11, no. 24, pp. 11905–11921, Dec. 2019, doi: 10.18632/aging.102505.

[89] D. Stangel *et al.*, "Kif20a inhibition reduces migration and invasion of pancreatic cancer cells," *J Surg Res*, vol. 197, no. 1, pp. 91–100, Jul. 2015, doi: 10.1016/j.jss.2015.03.070.

[90] "PDCD10 programmed cell death 10 [Homo sapiens (human)] - Gene - NCBI." https://www.ncbi.nlm.nih.gov/gene/11235 (accessed May 31, 2022).

[91] H.-P. Hsu, C.-Y. Wang, P.-Y. Hsieh, J.-H. Fang, and Y.-L. Chen, "Knockdown of serine/threonine-protein kinase 24 promotes tumorigenesis and myeloid-derived suppressor cell expansion in an orthotopic immunocompetent gastric cancer animal model," *J Cancer*, vol. 11, no. 1, pp. 213–228, Jan. 2020, doi: 10.7150/jca.35821.

[92] L. Liang, V. Chen, K. Zhu, X. Fan, X. Lu, and S. Lu, "Integrating data and knowledge to identify functional modules of genes: a multilayer approach," *BMC Bioinformatics*, vol. 20, no. 1, p. 225, Dec. 2019, doi: 10.1186/s12859-019-2800-y.

[93] M. Shroff, A. Knebel, R. Toth, and J. Rouse, "A complex comprising C15ORF41 and Codanin-1: the products of two genes mutated in congenital dyserythropoietic anaemia type I (CDA-I)," *Biochemical Journal*, vol. 477, no. 10, pp. 1893–1905, May 2020, doi: 10.1042/BCJ20190944.

[94] R. Russo *et al.*, "Characterization of Two Cases of Congenital Dyserythropoietic Anemia Type I Shed Light on the Uncharacterized C15orf41 Protein," *Frontiers in Physiology*, vol. 10, 2019, Accessed: Jun. 08, 2022. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphys.2019.00621

[95] Y. Tang *et al.*, "Structure of a human ASF1a/HIRA complex and insights into specificity of histone chaperone complex assembly," *Nat Struct Mol Biol*, vol. 13, no. 10, pp. 921–929, Oct. 2006, doi: 10.1038/nsmb1147.

[96] T. S. Rai *et al.*, "Human CABIN1 Is a Functional Member of the Human HIRA/UBN1/ASF1a Histone H3.3 Chaperone Complex ▿ ," *Mol Cell Biol*, vol. 31, no. 19, pp. 4107–4118, Oct. 2011, doi: 10.1128/MCB.05546-11.

[97] G. Swickley *et al.*, "Characterization of the interactions between Codanin-1 and C15Orf41, two proteins implicated in congenital dyserythropoietic anemia type I disease," *BMC Molecular and Cell Biology*, vol. 21, no. 1, p. 18, Mar. 2020, doi: 10.1186/s12860-020-00258-1.

[98] R. Evans *et al.*, “Protein complex prediction with AlphaFold-Multimer.” bioRxiv, p. 2021.10.04.463034, Mar. 10, 2022. doi: 10.1101/2021.10.04.463034.
[99] M. Mirdita, K. Schütze, Y. Moriwaki, L. Heo, S. Ovchinnikov, and M. Steinegger, “ColabFold: making protein folding accessible to all,” *Nat Methods*, vol. 19, no. 6, pp. 679–682, Jun. 2022, doi: 10.1038/s41592-022-01488-1.
[100] M. Shroff, A. Knebel, R. Toth, and J. Rouse, “A complex comprising C15ORF41 and Codanin-1: the products of two genes mutated in congenital dyserythropoietic anaemia type I (CDA-I),” *Biochem J*, vol. 477, no. 10, pp. 1893–1905, May 2020, doi: 10.1042/BCJ20190944.
[101] Y. Tang *et al.*, “Structure of a human ASF1a-HIRA complex and insights into specificity of histone chaperone complex assembly,” *Nat Struct Mol Biol*, vol. 13, no. 10, pp. 921–929, Oct. 2006, doi: 10.1038/nsmb1147.
[102] T. Wassmer *et al.*, “The retromer coat complex coordinates endosomal sorting and dynein-mediated transport, with carrier recognition by the trans-Golgi network,” *Dev Cell*, vol. 17, no. 1, pp. 110–122, Jul. 2009, doi: 10.1016/j.devcel.2009.04.016.
[103] “Subcellular - C11orf42 - The Human Protein Atlas.” https://www.proteinatlas.org/ENSG00000180878-C11orf42/subcellular (accessed Jun. 16, 2022).
[104] “Subcellular - SNX5 - The Human Protein Atlas.” https://www.proteinatlas.org/ENSG00000089006-SNX5/subcellular (accessed Jun. 16, 2022).
[105] “Subcellular - VPS29 - The Human Protein Atlas.” https://www.proteinatlas.org/ENSG00000111237-VPS29/subcellular (accessed Jun. 16, 2022).
[106] “Subcellular - SNX2 - The Human Protein Atlas.” https://www.proteinatlas.org/ENSG00000205302-SNX2/subcellular (accessed Jun. 16, 2022).
[107] “Subcellular - SNX1 - The Human Protein Atlas.” https://www.proteinatlas.org/ENSG00000028528-SNX1/subcellular (accessed Jun. 16, 2022).
[108] F. L. Kyrilis, A. Meister, and P. L. Kastritis, “Integrative biology of native cell extracts: a new era for structural characterization of life processes,” *Biological Chemistry*, vol. 400, no. 7, pp. 831–846, Jul. 2019, doi: 10.1515/hsz-2018-0445.
[109] E. Callaway, “Revolutionary cryo-EM is taking over structural biology,” *Nature*, vol. 578, no. 7794, pp. 201–201, Feb. 2020, doi: 10.1038/d41586-020-00341-9.
[110] D. Wrapp *et al.*, “Cryo-EM structure of the 2019-nCoV spike in the prefusion conformation,” *Science*, vol. 367, no. 6483, pp. 1260–1263, Mar. 2020, doi: 10.1126/science.abb2507.
[111] P. L. Kastritis *et al.*, “Capturing protein communities by structural proteomics in a thermophilic eukaryote,” *Mol Syst Biol*, vol. 13, no. 7, p. 936, Jul. 2017, doi: 10.15252/msb.20167412.
[112] E. J. Verbeke, A. L. Mallam, K. Drew, E. M. Marcotte, and D. W. Taylor, “Classification of Single Particles from Human Cell Extract Reveals Distinct

Structures," *Cell Rep*, vol. 24, no. 1, pp. 259-268.e3, Jul. 2018, doi: 10.1016/j.celrep.2018.06.022.
[113] Y. Aizenbud and Y. Shkolnisky, "A max-cut approach to heterogeneity in cryo-electron microscopy," *Journal of Mathematical Analysis and Applications*, vol. 479, no. 1, pp. 1004–1029, Nov. 2019, doi: 10.1016/j.jmaa.2019.06.064.
[114] G. T. Herman and M. Kalinowski, "Classification of heterogeneous electron microscopic projections into homogeneous subsets," *Ultramicroscopy*, vol. 108, no. 4, pp. 327–338, Mar. 2008, doi: 10.1016/j.ultramic.2007.05.005.
[115] M. Shatsky, R. J. Hall, E. Nogales, J. Malik, and S. E. Brenner, "Automated multi-model reconstruction from single-particle electron microscopy data," *J Struct Biol*, vol. 170, no. 1, pp. 98–108, Apr. 2010, doi: 10.1016/j.jsb.2010.01.007.
[116] E. J. Verbeke, Y. Zhou, A. P. Horton, A. L. Mallam, D. W. Taylor, and E. M. Marcotte, "Separating distinct structures of multiple macromolecular assemblies from cryo-EM projections," *Journal of Structural Biology*, vol. 209, no. 1, p. 107416, Jan. 2020, doi: 10.1016/j.jsb.2019.107416.
[117] M. Van Heel, "Angular reconstitution: A posteriori assignment of projection directions for 3D reconstruction," *Ultramicroscopy*, vol. 21, no. 2, pp. 111–123, Jan. 1987, doi: 10.1016/0304-3991(87)90078-7.
[118] Y. Zhuang *et al.*, "Unsupervised learning approaches to characterize heterogeneous samples using X-ray single particle imaging." arXiv, Sep. 13, 2021. doi: 10.48550/arXiv.2109.06179.
[119] S. Pan, C. Zhu, X.-M. Zhao, and L. P. Coelho, "A deep siamese neural network improves metagenome-assembled genomes in microbiome datasets across different environments," *Nat Commun*, vol. 13, no. 1, Art. no. 1, Apr. 2022, doi: 10.1038/s41467-022-29843-y.
[120] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 25. Accessed: May 31, 2022. [Online]. Available: https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html
[121] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv, arXiv:1409.1556, Apr. 2015. doi: 10.48550/arXiv.1409.1556.
[122] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.
[123] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv, arXiv:1512.03385, Dec. 2015. doi: 10.48550/arXiv.1512.03385.
[124] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," arXiv, arXiv:1905.11946, Sep. 2020. doi: 10.48550/arXiv.1905.11946.

[125] C. Safka, *Image 2 Vec with PyTorch*. 2022. Accessed: Jun. 01, 2022. [Online]. Available: https://github.com/christiansafka/img2vec
[126] D. Chicco, "Siamese Neural Networks: An Overview," in *Artificial Neural Networks*, H. Cartwright, Ed. New York, NY: Springer US, 2021, pp. 73–94. doi: 10.1007/978-1-0716-0826-5_3.
[127] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
[128] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 815–823. doi: 10.1109/CVPR.2015.7298682.
[129] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable Representation Learning for Heterogeneous Networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, Aug. 2017, pp. 135–144. doi: 10.1145/3097983.3098036.
[130] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning Structural Node Embeddings Via Diffusion Wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018, pp. 1320–1329. doi: 10.1145/3219819.3220025.
[131] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. Alemi, "Watch Your Step: Learning Node Embeddings via Graph Attention," arXiv, arXiv:1710.09599, Sep. 2018. doi: 10.48550/arXiv.1710.09599.
[132] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Attributed network embedding via subspace discovery," *Data Min Knowl Disc*, vol. 33, no. 6, pp. 1953–1980, Nov. 2019, doi: 10.1007/s10618-019-00650-2.
[133] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," arXiv, arXiv:1706.02216, Sep. 2018. doi: 10.48550/arXiv.1706.02216.
[134] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep Graph Infomax," arXiv, arXiv:1809.10341, Dec. 2018. doi: 10.48550/arXiv.1809.10341.
[135] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," Oct. 2017, doi: 10.48550/arXiv.1710.10903.
[136] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank," arXiv, arXiv:1810.05997, Apr. 2022. doi: 10.48550/arXiv.1810.05997.
[137] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2019, pp. 257–266. doi: 10.1145/3292500.3330925.

[138] CSIRO's Data61, *StellarGraph Machine Learning Library*. S T E L L A R, 2018. Accessed: May 31, 2022. [Online]. Available: https://github.com/stellargraph/stellargraph
[139] M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analysers," p. 30.
[140] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," arXiv, arXiv:0909.4061, Dec. 2010. doi: 10.48550/arXiv.0909.4061.
[141] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, Aug. 1996, pp. 226–231.
[142] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," *SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, Jun. 1999, doi: 10.1145/304181.304187.
[143] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," *SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996, doi: 10.1145/235968.233324.
[144] B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007, doi: 10.1126/science.1136800.
[145] M. V. Palukuri and E. M. Marcotte, "Super.Complex: A supervised machine learning pipeline for molecular complex detection in protein-interaction networks," *PLOS ONE*, vol. 16, no. 12, p. e0262056, Dec. 2021, doi: 10.1371/journal.pone.0262056.
[146] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987, doi: 10.1016/0377-0427(87)90125-7.
[147] P. Pons and M. Latapy, "Computing communities in large networks using random walks (long version)," arXiv, arXiv:physics/0512106, Dec. 2005. doi: 10.48550/arXiv.physics/0512106.
[148] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, Jun. 2002, doi: 10.1073/pnas.122653799.
[149] G. Csardi and T. Nepusz, "The Igraph Software Package for Complex Network Research," *InterJournal*, vol. Complex Systems, p. 1695, Nov. 2005.
[150] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, p. 066111, Dec. 2004, doi: 10.1103/PhysRevE.70.066111.
[151] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, Art. no. 7043, Jun. 2005, doi: 10.1038/nature03607.
[152] G. Cordasco and L. Gargano, "Community Detection via Semi-Synchronous Label Propagation Algorithms," Mar. 2011. doi: 10.1504/..045103.

[153] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Phys. Rev. E*, vol. 76, no. 3, p. 036106, Sep. 2007, doi: 10.1103/PhysRevE.76.036106.
[154] M. V. Palukuri, R. S. Patil, and E. M. Marcotte, “Molecular complex detection in protein interaction networks through reinforcement learning.” bioRxiv, p. 2022.06.20.496772, Jun. 21, 2022. doi: 10.1101/2022.06.20.496772.
[155] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, Jan. 1974, doi: 10.1080/03610927408827101.
[156] D. L. Davies and D. W. Bouldin, “A Cluster Separation Measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979, doi: 10.1109/TPAMI.1979.4766909.
[157] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.
[158] E. D. Zhong, T. Bepler, B. Berger, and J. H. Davis, “CryoDRGN: reconstruction of heterogeneous cryo-EM structures using neural networks,” *Nat Methods*, vol. 18, no. 2, Art. no. 2, Feb. 2021, doi: 10.1038/s41592-020-01049-4.
[159] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, “Perspective transformer nets: learning single-view 3D object reconstruction without 3D supervision,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2016, pp. 1704–1712.
[160] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.
[161] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs.” arXiv, Feb. 28, 2017. doi: 10.48550/arXiv.1702.08734.
[162] J. A. Chen, O. Peñagarikano, T. G. Belgard, V. Swarup, and D. H. Geschwind, “The emerging picture of autism spectrum disorder: genetics and pathology,” *Annu Rev Pathol*, vol. 10, pp. 111–144, 2015, doi: 10.1146/annurev-pathol-012414-040405.
[163] T. Chen, Z. Xie, and Q. Ouyang, “Expanded flux variability analysis on metabolic network of Escherichia coli,” *Chin. Sci. Bull.*, vol. 54, no. 15, pp. 2610–2619, Aug. 2009, doi: 10.1007/s11434-009-0341-x.
[164] R. K. Murray, D. K. Granner, and V. W. Rodwell, *Harper’s Illustrated Biochemistry*. 2010. Accessed: Jul. 04, 2022. [Online]. Available: http://localhost:8080/xmlui/handle/123456789/1754
[165] P. Chaste and M. Leboyer, “Autism risk factors: genes, environment, and gene-environment interactions,” *Dialogues Clin Neurosci*, vol. 14, no. 3, pp. 281–292, Sep. 2012.
[166] R. E. Frye, S. Melnyk, and D. F. Macfabe, “Unique acyl-carnitine profiles are potential biomarkers for acquired mitochondrial disease in autism spectrum disorder,” *Transl Psychiatry*, vol. 3, p. e220, Jan. 2013, doi: 10.1038/tp.2012.143.

[167] J. B. Adams *et al.*, "Nutritional and metabolic status of children with autism vs. neurotypical children, and the association with autism severity," *Nutr Metab (Lond)*, vol. 8, no. 1, p. 34, Jun. 2011, doi: 10.1186/1743-7075-8-34.
[168] R. E. Frye, S. Rose, J. Slattery, and D. F. MacFabe, "Gastrointestinal dysfunction in autism spectrum disorder: the role of the mitochondria and the enteric microbiome," *Microb Ecol Health Dis*, vol. 26, p. 27458, 2015, doi: 10.3402/mehd.v26.27458.
[169] N. Cheng, J. M. Rho, and S. A. Masino, "Metabolic Dysfunction Underlying Autism Spectrum Disorder and Potential Treatment Approaches," *Front Mol Neurosci*, vol. 10, p. 34, 2017, doi: 10.3389/fnmol.2017.00034.
[170] L. Wang, M. T. Angley, J. P. Gerber, and M. J. Sorich, "A review of candidate urinary biomarkers for autism spectrum disorder," *Biomarkers*, vol. 16, no. 7, pp. 537–552, Nov. 2011, doi: 10.3109/1354750X.2011.598564.
[171] H. T. Ding, Y. Taur, and J. T. Walkup, "Gut Microbiota and Autism: Key Concepts and Findings," *J Autism Dev Disord*, vol. 47, no. 2, pp. 480–489, Feb. 2017, doi: 10.1007/s10803-016-2960-9.
[172] I. Cho and M. J. Blaser, "The human microbiome: at the interface of health and disease," *Nat Rev Genet*, vol. 13, no. 4, pp. 260–270, Mar. 2012, doi: 10.1038/nrg3182.
[173] B. K. Huang and H. D. Sikes, "Quantifying intracellular hydrogen peroxide perturbations in terms of concentration," *Redox Biol*, vol. 2, pp. 955–962, 2014, doi: 10.1016/j.redox.2014.08.001.
[174] S. Adamberg *et al.*, "Degradation of Fructans and Production of Propionic Acid by Bacteroides thetaiotaomicron are Enhanced by the Shortage of Amino Acids," *Front Nutr*, vol. 1, p. 21, 2014, doi: 10.3389/fnut.2014.00021.
[175] J. Wang, X. Gu, J. Yang, Y. Wei, and Y. Zhao, "Gut Microbiota Dysbiosis and Increased Plasma LPS and TMAO Levels in Patients With Preeclampsia," *Front Cell Infect Microbiol*, vol. 9, p. 409, 2019, doi: 10.3389/fcimb.2019.00409.
[176] C. G. M. de Theije *et al.*, "Pathways underlying the gut-to-brain connection in autism spectrum disorders as future targets for disease management," *Eur J Pharmacol*, vol. 668 Suppl 1, pp. S70-80, Sep. 2011, doi: 10.1016/j.ejphar.2011.07.013.
[177] Q. Li, Y. Han, A. B. C. Dy, and R. J. Hagerman, "The Gut Microbiota and Autism Spectrum Disorders," *Front Cell Neurosci*, vol. 11, p. 120, 2017, doi: 10.3389/fncel.2017.00120.
[178] S. M. Finegold, "Therapy and epidemiology of autism--clostridial spores as key elements," *Med Hypotheses*, vol. 70, no. 3, pp. 508–511, 2008, doi: 10.1016/j.mehy.2007.07.019.
[179] S. M. Finegold *et al.*, "Pyrosequencing study of fecal microflora of autistic and control children," *Anaerobe*, vol. 16, no. 4, pp. 444–453, Aug. 2010, doi: 10.1016/j.anaerobe.2010.06.008.
[180] H. E. Vuong and E. Y. Hsiao, "Emerging Roles for the Gut Microbiome in Autism Spectrum Disorder," *Biol Psychiatry*, vol. 81, no. 5, pp. 411–423, Mar. 2017, doi: 10.1016/j.biopsych.2016.08.024.

[181] R. H. Sandler *et al.*, "Short-term benefit from oral vancomycin treatment of regressive-onset autism," *J Child Neurol*, vol. 15, no. 7, pp. 429–435, Jul. 2000, doi: 10.1177/088307380001500701.
[182] M. A. Oberhardt, J. B. Goldberg, M. Hogardt, and J. A. Papin, "Metabolic network analysis of Pseudomonas aeruginosa during chronic cystic fibrosis lung infection," *J Bacteriol*, vol. 192, no. 20, pp. 5534–5548, Oct. 2010, doi: 10.1128/JB.00900-10.
[183] S. Y. Shaaban *et al.*, "The role of probiotics in children with autism spectrum disorder: A prospective, open-label study," *Nutr Neurosci*, vol. 21, no. 9, pp. 676–681, Nov. 2018, doi: 10.1080/1028415X.2017.1347746.
[184] A. Chauhan and V. Chauhan, "Oxidative stress in autism," *Pathophysiology*, vol. 13, no. 3, pp. 171–181, Aug. 2006, doi: 10.1016/j.pathophys.2006.05.007.
[185] R. E. Frye and D. A. Rossignol, "Mitochondrial dysfunction can connect the diverse medical symptoms associated with autism spectrum disorders," *Pediatr Res*, vol. 69, no. 5 Pt 2, pp. 41R–7R, May 2011, doi: 10.1203/PDR.0b013e318212f16b.
[186] X. Ming, T. P. Stein, V. Barnes, N. Rhodes, and L. Guo, "Metabolic perturbance in autism spectrum disorders: a metabolomics study," *J Proteome Res*, vol. 11, no. 12, pp. 5856–5862, Dec. 2012, doi: 10.1021/pr300910n.
[187] D. C. Rojas, "The role of glutamate and its receptors in autism and the use of glutamate receptor antagonists in treatment," *J Neural Transm (Vienna)*, vol. 121, no. 8, pp. 891–905, Aug. 2014, doi: 10.1007/s00702-014-1216-0.
[188] J. Wasilewska and M. Klukowski, "Gastrointestinal symptoms and autism spectrum disorder: links and risks - a possible new overlap syndrome," *Pediatric Health Med Ther*, vol. 6, pp. 153–166, 2015, doi: 10.2147/PHMT.S85717.
[189] E. Thursby and N. Juge, "Introduction to the human gut microbiota," *Biochem J*, vol. 474, no. 11, pp. 1823–1836, May 2017, doi: 10.1042/BCJ20160510.
[190] J. K. Kern and A. M. Jones, "Evidence of toxicity, oxidative stress, and neuronal insult in autism," *J Toxicol Environ Health B Crit Rev*, vol. 9, no. 6, pp. 485–499, Dec. 2006, doi: 10.1080/10937400600882079.
[191] A. El-Ansary and L. Al-Ayadhi, "GABAergic/glutamatergic imbalance relative to excessive neuroinflammation in autism spectrum disorders," *J Neuroinflammation*, vol. 11, p. 189, Nov. 2014, doi: 10.1186/s12974-014-0189-0.
[192] G. Bjørklund *et al.*, "Oxidative Stress in Autism Spectrum Disorder," *Mol Neurobiol*, vol. 57, no. 5, pp. 2314–2332, May 2020, doi: 10.1007/s12035-019-01742-2.
[193] D. F. MacFabe *et al.*, "Neurobiological effects of intraventricular propionic acid in rats: possible role of short chain fatty acids on the pathogenesis and characteristics of autism spectrum disorders," *Behav Brain Res*, vol. 176, no. 1, pp. 149–169, Jan. 2007, doi: 10.1016/j.bbr.2006.07.025.
[194] D. F. Macfabe, "Short-chain fatty acid fermentation products of the gut microbiome: implications in autism spectrum disorders," *Microb Ecol Health Dis*, vol. 23, 2012, doi: 10.3402/mehd.v23i0.19260.

[195] P. Srikantha and M. H. Mohajeri, “The Possible Role of the Microbiota-Gut-Brain-Axis in Autism Spectrum Disorder,” *Int J Mol Sci*, vol. 20, no. 9, p. E2115, Apr. 2019, doi: 10.3390/ijms20092115.
[196] D. B. Menzel, “ES Series: Cancer Risk Assessment. 2. Physiological pharmacokinetic modeling,” *Environ Sci Technol*, vol. 21, no. 10, pp. 944–950, Oct. 1987, doi: 10.1021/es50001a004.
[197] X. Zhuang and C. Lu, “PBPK modeling and simulation in drug research and development,” *Acta Pharm Sin B*, vol. 6, no. 5, pp. 430–440, Sep. 2016, doi: 10.1016/j.apsb.2016.04.004.
[198] L. Heirendt *et al.*, “Creation and analysis of biochemical constraint-based models using the COBRA Toolbox v.3.0,” *Nat Protoc*, vol. 14, no. 3, pp. 639–702, Mar. 2019, doi: 10.1038/s41596-018-0098-2.
[199] D. Zheng, T. Liwinski, and E. Elinav, “Interaction between microbiota and immunity in health and disease,” *Cell Res*, vol. 30, no. 6, pp. 492–506, Jun. 2020, doi: 10.1038/s41422-020-0332-7.
[200] J. Elzinga, J. van der Oost, W. M. de Vos, and H. Smidt, “The Use of Defined Microbial Communities To Model Host-Microbe Interactions in the Human Gut,” *Microbiol Mol Biol Rev*, vol. 83, no. 2, pp. e00054-18, May 2019, doi: 10.1128/MMBR.00054-18.
[201] A. Heinken and I. Thiele, “Systems biology of host-microbe metabolomics,” *Wiley Interdiscip Rev Syst Biol Med*, vol. 7, no. 4, pp. 195–219, Aug. 2015, doi: 10.1002/wsbm.1301.
[202] S. Magnúsdóttir and I. Thiele, “Modeling metabolism of the human gut microbiome,” *Curr Opin Biotechnol*, vol. 51, pp. 90–96, Jun. 2018, doi: 10.1016/j.copbio.2017.12.005.
[203] F. Baldini, A. Heinken, L. Heirendt, S. Magnusdottir, R. M. T. Fleming, and I. Thiele, “The Microbiome Modeling Toolbox: from microbial interactions to personalized microbial communities,” *Bioinformatics*, vol. 35, no. 13, pp. 2332–2334, Jul. 2019, doi: 10.1093/bioinformatics/bty941.
[204] S. Magnúsdóttir *et al.*, “Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota,” *Nat Biotechnol*, vol. 35, no. 1, pp. 81–89, Jan. 2017, doi: 10.1038/nbt.3703.
[205] E. Bauer and I. Thiele, “From Network Analysis to Functional Metabolic Modeling of the Human Gut Microbiota,” *mSystems*, vol. 3, no. 3, pp. e00209-17, Jun. 2018, doi: 10.1128/mSystems.00209-17.
[206] A. Noronha *et al.*, “The Virtual Metabolic Human database: integrating human and gut microbiome metabolism with nutrition and disease,” *Nucleic Acids Res*, vol. 47, no. D1, pp. D614–D624, Jan. 2019, doi: 10.1093/nar/gky992.
[207] S. Sahoo and I. Thiele, “Predicting the impact of diet and enzymopathies on human small intestinal epithelial cells,” *Hum Mol Genet*, vol. 22, no. 13, pp. 2705–2722, Jul. 2013, doi: 10.1093/hmg/ddt119.
[208] S. Coghlan, J. Horder, B. Inkster, M. A. Mendez, D. G. Murphy, and D. J. Nutt, “GABA system dysfunction in autism and related disorders: From synapse to

symptoms," *Neuroscience & Biobehavioral Reviews*, vol. 36, no. 9, pp. 2044–2055, Oct. 2012, doi: 10.1016/j.neubiorev.2012.07.005.
[209] N. E. Lewis *et al.*, "Large-scale in silico modeling of metabolic interactions between cell types in the human brain," *Nat Biotechnol*, vol. 28, no. 12, pp. 1279–1285, Dec. 2010, doi: 10.1038/nbt.1711.
[210] F. Y. Bois, "Physiologically based modelling and prediction of drug interactions," *Basic Clin Pharmacol Toxicol*, vol. 106, no. 3, pp. 154–161, Mar. 2010, doi: 10.1111/j.1742-7843.2009.00488.x.
[211] M. D. Thompson and D. A. Beard, "Physiologically based pharmacokinetic tissue compartment model selection in drug development and risk assessment," *J Pharm Sci*, vol. 101, no. 1, pp. 424–435, Jan. 2012, doi: 10.1002/jps.22768.
[212] J. P. Castro, T. Grune, and B. Speckmann, "The two faces of reactive oxygen species (ROS) in adipocyte function and dysfunction," *Biol Chem*, vol. 397, no. 8, pp. 709–724, Aug. 2016, doi: 10.1515/hsz-2015-0305.
[213] A. R. Katritzky *et al.*, "QSAR modeling of blood:air and tissue:air partition coefficients using theoretical descriptors," *Bioorg Med Chem*, vol. 13, no. 23, pp. 6450–6463, Dec. 2005, doi: 10.1016/j.bmc.2005.06.066.
[214] K. Roy, S. Kar, and rudra n das, *A Primer on QSAR/QSPR Modeling*. Fundamental Concepts. Cham: Springer International Publishing. Accessed: Jul. 04, 2022. [Online]. Available: doi: 10.1007/978-3-319-17281-1
[215] F. Gui, Y. Ma, F. Zhang, M. Liu, R. Yin, and W. Shen, "A Stable and Distributed Community Detection Algorithm Based on Maximal Cliques," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Kowloon, Oct. 2015, pp. 1346–1350. doi: 10.1109/SMC.2015.239.
[216] A. L. Sheldon and M. B. Robinson, "The role of glutamate transporters in neurodegenerative diseases and potential opportunities for intervention," *Neurochem Int*, vol. 51, no. 6–7, pp. 333–355, Dec. 2007, doi: 10.1016/j.neuint.2007.03.012.
[217] D. L. Miller, "Rat small intestine: development, composition and effects of perfusion," *Am J Dig Dis*, vol. 16, no. 3, pp. 247–254, Mar. 1971, doi: 10.1007/BF02235247.
[218] S. S. Zoroglu *et al.*, "Increased oxidative stress and altered activities of erythrocyte free radical scavenging enzymes in autism," *Eur Arch Psychiatry Clin Neurosci*, vol. 254, no. 3, pp. 143–147, Jun. 2004, doi: 10.1007/s00406-004-0456-7.
[219] D. Nagrath, M. Avila-Elchiver, F. Berthiaume, A. W. Tilles, A. Messac, and M. L. Yarmush, "Integrated energy and flux balance based multiobjective framework for large-scale metabolic networks," *Ann Biomed Eng*, vol. 35, no. 6, pp. 863–885, Jun. 2007, doi: 10.1007/s10439-007-9283-0.
[220] S. Gudmundsson and I. Thiele, "Computationally efficient flux variability analysis," *BMC Bioinformatics*, vol. 11, p. 489, Sep. 2010, doi: 10.1186/1471-2105-11-489.
[221] I. K. S. Yap, M. Angley, K. A. Veselkov, E. Holmes, J. C. Lindon, and J. K. Nicholson, "Urinary metabolic phenotyping differentiates children with autism from

their unaffected siblings and age-matched controls," *J Proteome Res*, vol. 9, no. 6, pp. 2996–3004, Jun. 2010, doi: 10.1021/pr901188e.
[222] D. A. Rossignol and R. E. Frye, "Mitochondrial dysfunction in autism spectrum disorders: a systematic review and meta-analysis," *Mol Psychiatry*, vol. 17, no. 3, pp. 290–314, Mar. 2012, doi: 10.1038/mp.2010.136.
[223] I. Görker and U. Tüzün, "Autistic-like findings associated with a urea cycle disorder in a 4-year-old girl," *J Psychiatry Neurosci*, vol. 30, no. 2, pp. 133–135, Mar. 2005.
[224] L. de Magistris *et al.*, "Alterations of the intestinal barrier in patients with autism spectrum disorders and in their first-degree relatives," *J Pediatr Gastroenterol Nutr*, vol. 51, no. 4, pp. 418–424, Oct. 2010, doi: 10.1097/MPG.0b013e3181dcc4a5.
[225] M. Fiorentino *et al.*, "Blood-brain barrier and intestinal epithelial barrier alterations in autism spectrum disorders," *Mol Autism*, vol. 7, p. 49, 2016, doi: 10.1186/s13229-016-0110-z.
[226] R. Mahadevan, J. S. Edwards, and F. J. Doyle, "Dynamic Flux Balance Analysis of Diauxic Growth in Escherichia coli," *Biophysical Journal*, vol. 83, no. 3, pp. 1331–1340, Sep. 2002, doi: 10.1016/S0006-3495(02)73903-9.
[227] M. Ibrahim, L. Raajaraam, and K. Raman, "Modelling microbial communities: Harnessing consortia for biotechnological applications," *Comput Struct Biotechnol J*, vol. 19, pp. 3892–3907, 2021, doi: 10.1016/j.csbj.2021.06.048.
[228] J. Schellenberger *et al.*, "Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0," *Nat Protoc*, vol. 6, no. 9, pp. 1290–1307, Aug. 2011, doi: 10.1038/nprot.2011.308.
[229] J. Slavin, "Fiber and prebiotics: mechanisms and health benefits," *Nutrients*, vol. 5, no. 4, pp. 1417–1435, Apr. 2013, doi: 10.3390/nu5041417.
[230] H. Kondoh, M. E. Lleonart, D. Bernard, and J. Gil, "Protection from oxidative stress by enhanced glycolysis; a possible mechanism of cellular immortalization," *Histol Histopathol*, vol. 22, no. 1, pp. 85–90, Jan. 2007, doi: 10.14670/HH-22.85.
[231] R. Downs, J. Perna, A. Vitelli, D. Cook, and P. Dhurjati, "Model-based hypothesis of gut microbe populations and gut/brain barrier permeabilities in the development of regressive autism," *Med Hypotheses*, vol. 83, no. 6, pp. 649–655, Dec. 2014, doi: 10.1016/j.mehy.2014.09.005.
[232] W. R. McGinnis, "Oxidative stress in autism," *Altern Ther Health Med*, vol. 10, no. 6, pp. 22–36; quiz 37, 92, Dec. 2004.
[233] J. C. Naviaux *et al.*, "Reversal of autism-like behaviors and metabolism in adult mice with single-dose antipurinergic therapy," *Transl Psychiatry*, vol. 4, p. e400, Jun. 2014, doi: 10.1038/tp.2014.33.
[234] H. M. Wexler, "Bacteroides: the good, the bad, and the nitty-gritty," *Clin Microbiol Rev*, vol. 20, no. 4, pp. 593–621, Oct. 2007, doi: 10.1128/CMR.00008-07.
[235] S. M. Finegold, "Desulfovibrio species are potentially important in regressive autism," *Med Hypotheses*, vol. 77, no. 2, pp. 270–274, Aug. 2011, doi: 10.1016/j.mehy.2011.04.032.

[236] D.-W. Kang *et al.*, “Microbiota Transfer Therapy alters gut ecosystem and improves gastrointestinal and autism symptoms: an open-label study,” *Microbiome*, vol. 5, no. 1, p. 10, Jan. 2017, doi: 10.1186/s40168-016-0225-7.
[237] I. Kurochkin *et al.*, “Metabolome signature of autism in the human prefrontal cortex,” *Commun Biol*, vol. 2, p. 234, 2019, doi: 10.1038/s42003-019-0485-4.

Vita

I was born a few streets away from the Bay of Bengal, in the beautiful city of 'Vizag', Visakhapatnam, in India, and have not lived in the same house for more than 2 years; moving to Singapore, Hyderabad, Chennai and finally Austin. I received my undergraduate and Master's from IIT Madras, a campus set in the Guindy national park in Chennai, and decided I love the campus life enough to spend 5 more years in a university. I could not have chosen a more beautiful place to do so, living by the river Colorado in Austin, TX, courtesy of UT's graduate apartments.

Inspired by my father, I learned to code in my high school years and developed a love for solving problems programmatically; and have been seeking out computational projects ever since. An interest in chemistry led me to study chemical engineering, where I was introduced to computational biochemistry during my Masters' project. This coincided with my younger sister choosing to be a vet, leading her to develop and share her love for biology with me, motivating my foray into computational biology. A love for reading, learning and exploring that I learned from my mother from a young age, led me towards research and a Ph.D. in a field that intrigued me.

Sometime during the 4th year of my undergraduate, I watched the movie Good Will Hunting, where smart Matt Damon solves a hard graph-theoretic problem (without any formal training in the field). I was impressed and wanted to be smart like him, however, unlike him, I decided to train a bit more formally in the field, and took some classes on the applications of graph theory. Ever since, I have been working with graphs, from microfluidic networks to metabolic networks to finally in my Ph.D., protein interaction networks, networks linking projections of images, and transportation networks during my curricular practical training at Amazon. Community detection as a concept appealed to me fundamentally, for, humans are social beings who live in communities, and finding

communities for different things turns out to be useful, just like it does for humans. Machine learning was another natural choice to explore for me, as learning how to learn from others can help humanity as a huge community. Hence my Ph.D. thesis, titled, Machine learning methods for community detection methods using known information. I hope to work in this general space for the rest of my life as well, and hope to contribute in one way or the other to the world.

Permanent address (or email): Meghana.palukuri@utexas.edu

This dissertation was typed by Meghana Venkata Palukuri.