NETWORK-LEVEL STUDY OF PROTEIN-PROTEIN INTERACTIONS: ANALYSIS AND PREDICTION

by

Lei Huang

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Spring 2017

NETWORK-LEVEL STUDY OF PROTEIN-PROTEIN INTERACTIONS: ANALYSIS AND PREDICTION

by

Lei Huang

Approved: _____
Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____
Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____
Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Li Liao, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Cathy H. Wu, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Vijay K. Shanker, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chi-Ren Shyu, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

The five-year PhD experience in the Bioinformatics Lab of University of Delaware will be one of my most important and wonderful experience in my life. This dissertation would not have been possible without the support of many people. First and foremost, I would like to sincerely thank my two advisors Dr. Li Liao and Dr. Cathy H. Wu. I am so fortunate to have been your student. You both provided me the perfect balance between guidance and freedom on my research. Li, I really appreciate your unfailing help and support whenever I needed. I admire your keen insights in everything, big and small. I could not achieve my PhD accomplishments without your help. Cathy, I am really grateful for your unfailing support and precious advice to my PhD research; and I am greatly influenced and encouraged by your endless devotion and passion in research and life. I also would like to thank my committee members Dr. Vijay K. Shanker and Dr. Chi-Ren Shyu for their support and help on my projects and dissertation.

I want to thank my labmate and good friend Tianchuan Du for his help and advice on life, research and career. I also want to thank Daniel Tan and Guangjie Ren for their support and mentorship during my internship in IBM Almaden research center. They helped me connect academic and industrial research, and set my career goal. I also want to thank all the friends on the basketball court. Every game on Wednesday night really helped me release pressure and keep me fit.

I would not be where I am today without the constant support and unconditional love from my parents and sister. I also want to especially thank my wife Zhixuan, the love of my life, for her tremendous sacrifice, unconditional love and timely encouragement. She was also very busying with her PhD study, but she could always keep

everything organized for our family. My amazing son Jaden was born during the preparation of this dissertation, and he was cheering me up on day and night. I could not imagine a more joyful distraction than that. The dissertation is dedicated to him.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With continuous efforts in identifying protein-protein interactions (PPIs) through both high-throughput wet-lab experiments and computational methods, an increasing number of new PPIs have been discovered and validated, enabling sizeable (even genome wide) PPI networks to be formed. Therefore, it has become feasible and also imperative to study PPIs, as a whole, at the network level; to gain knowledge about the network topology and evolution; and to leverage the newly gained knowledge to advance the reconstruction of PPI networks, which are still quite sparse in most cases, by inferring de novo PPIs that are difficult to predict without a network context.

In this dissertation, we systematically studied the PPI networks in terms of network evolution analysis and network completion with predicting de novo PPIs, and have proposed and developed a suite of novel methods from selecting evolutionary models to utilizing network evolution and topology, and leveraging multiple heterogeneous data sources for predicting PPIs.

PPI evolution analysis aims at identifying the underlying evolution/growth mechanism of PPI networks, which plays a crucial role for understanding PPIs as a network system and for predicting new interactions. By exploring the state-of-the-art PPI network evolution models, we developed a novel sampling method based on Approximate Bayesian Computation and modified Differential Evolution algorithm to select the most fitting evolution model for different PPI networks. The results from our analysis based on Human and Yeast PPI networks show that different PPI networks may have different evolution/growth models: for Human PPI networks, Duplication-Attachment is the predominant mechanism while Scale-Free is the predominant mechanism for Yeast PPI networks. Equipped with the evolution models for different PPI

networks, we designed a novel PPI prediction method to include the evolution information into the geometric embedding, which consequently improves the PPI prediction performance by about 15%.

Despite of the rapid growth, PPI networks by and large remain incomplete and sparsely disconnected for most organisms, and therefore network completion poses a grand challenge in systems biology. Many traditional network-level PPI prediction methods use only connectivity information of existing edges to predict PPIs. However, from a PPI prediction perspective, what is particularly useful is to incorporate pairwise features for node pairs that are not currently linked by a direct edge but may become linked. In this dissertation, we developed novel PPI network inference methods that can utilize pairwise features for all node pairs, regardless whether they are currently directly connected or not. In particular, our methods can help integrate various heterogeneous feature kernels, e.g. gene co-expression kernel, protein sequence similarity kernel, etc., to build the PPI inference matrix, whose element is interpreted as probability of how likely the two corresponding proteins will interact. Specifically, we adopt two strategies to optimize weights for various feature kernels to build the kernel fusion and eventually the PPI inference matrix. Tested on Yeast PPI data and compared with two control methods, our proposed methods shows a significant improvement in performance as measured by receiver operating characteristic.

Another challenge of PPI prediction is how to train prediction model over extremely sparse and disconnected PPI networks. Many of existing network level methods assume the training network should be connected. However, that assumption greatly affects their predictive power and limits the application area because current golden standard PPI networks are actually very sparse and disconnected. We developed a novel PPI prediction method based on deep learning neural network and regularized Laplacian kernel. We use the neural network to implicitly simulate and guide the evolution process of a PPI network by using rows of an ancient network as inputs and rows of the disconnected training network as labels. After the training step, an evolved PPI network whose rows are outputs of the neural network can be obtained. Then we

predict PPIs by applying the regularized Laplacian kernel to the transition matrix that is built upon the evolved PPI network. The results from cross-validation experiments show that the PPI prediction accuracy for yeast data and human data can be further improved. Meanwhile, the transition matrix based on the evolved PPI network can also be used to leverage complementary information from the disconnected training network and multiple heterogeneous data sources.

In sum, the work in this dissertation contributes to the understanding of PPI networks, especially, those that are large and sparse, by novel methods in selecting network evolutionary models and leveraging network topology and heterogeneous features to improve the prediction performance. We believe methods proposed in this dissertation will be useful tools to help researchers further analyze PPI data and predict PPIs.

Chapter 1

INTRODUCTION

Protein-protein interaction (PPI) plays an essential role in many cellular processes. The study of PPIs can help us to have a better understanding of intracellular signaling pathways, modeling of protein complex structures and elucidating various biochemical processes. In this chapter, firstly we provide a brief background on the biological and computational concepts. Then we discuss recent related works and motivations of our research. After that, we briefly summarize our contributions to this research topic. Meanwhile, we also list some publicly accessible data sources that relevant to or have been adopted in our research.

1.1 DNA, RNA, and Proteins

DNA, RNA and Proteins are three essential macromolecules of life. DNA or otherwise called deoxyribonucleic acid is made up of four nucleotide bases: Adenine (A), Guanine (G), Thymine (T) and Cytosine(C), which is the building block of the life and carries genetic information and consists of thousands of genes. A gene is a segment of DNA that encodes functional RNA and protein products. To a certain extent, RNA is similar to DNA, both of them are nucleic acids of nitrogen-containing bases joined by sugar-phosphate backbone. However, structurally, RNA is a single-stranded where as DNA is double stranded; and DNA has Thymine, while as RNA has Uracil. Moreover, RNA nucleotides include ribose sugar, rather than the Deoxyribose sugar included by DNA. Functionally, DNA maintains the genetic information, whereas RNA uses the information to synthesize the particular protein.

The general process from DNA to protein that consists of transcription, transport, and translation, which is also called central dogma of life. More specifically, the

corresponding genes are transcribed into RNA; then the RNA is first processed so that non-coding parts are removed and is then transported out of the nucleus. After that, the proteins are built based on the code in the RNA.

Proteins are large, complex molecules consists of one or more long chains of amino acid residues, which play many critical roles in the living organisms. Their diverse functions can be providing structures, regulating biological process, transporting materials, building immunization system, etc.

1.2 Protein-Protein Interaction (PPI) and PPI Networks

It is well known that proteins are rarely perform their functions alone, but cooperate with some other function-similar proteins to form protein complex through PPIs. PPIs imply physical contacts between two or more proteins as a result of biochemical events and/or electrostatic forces, which relates almost all the biological processes. Consequently, PPI analysis and prediction remains central task in system biology; and efficient and accurate methods are urgently needed.

Traditionally, many high-throughput experimental methods, such as yeast two-hybrid system, co-Immunoprecipitation (Co-IP) and so on, have been used by biologists to uncover protein interactions. However, it has been demonstrated that these traditional wet-lab methods are prone to having high false-positive rates, besides their high cost. Consequently, Many computational approaches have been developed to predict if any given pair of proteins interact with each other by calculating their similarities based on various biological properties, such as sequence homology, gene co-expression, phylogenetic profiles, etc.

Recently, with continuous efforts in identifying PPIs through both high-throughput wet-lab experiments and computational methods, an increasing number of new PPIs have been discovered and validated, enabling sizeable (even genome wide) PPI networks to be formed, where proteins represent nodes, and interactions represent edges. As a result, researchers have begun to devote more attention to study PPI networks, aiming

at gaining an better understanding about biological function of individual proteins, protein complexes and even larger subcellular machines as a complex system.

1.3 Related Works

To computationally predict PPIs at network level, the most direct way is to calculate confidence scores (similarities) based on pair-wise topological features or network structure for protein node pairs. On the other hand, some researchers try to indirectly predict PPIs through evolution analysis of PPI networks, aiming to understanding the underlying mechanism of PPI formation. In this section, we plan to have a systematically study about the PPI networks from these two aspects.

1.3.1 Evolution Analysis of PPI Networks

Given there are no chronological PPI networks of a certain species available for evolution analysis, one effective way is to design models that can be used to simulate the evolution or growth process of PPI networks. In the past of few years, many popular PPI evolution models have been proposed. some early studies suggested scale free [11] model may fit PPI network well [15, 30], but there are several statistical challenges for this claim [51, 87]. It is commonly believed that one main mechanism by which PPIs network evolve is gene duplication, subsequently, another mechanism named post-duplication divergence may cause the PPI network further evolve. Briefly, the whole process can be described as follows: given an original PPIs network, any node may be duplicated at a certain probability; and then during the divergence process, some new connections may be formed between the duplicated node and existing nodes, meanwhile, some existing edges may be deleted. Based on the duplication and divergence mechanisms, many evolution models have been proposed [41, 42, 71] among which post-duplication divergence is the main difference. However, the limitation of gene duplication based models is that they may not generate networks with a module structure that resembles the module structure one can find in golden standard PPI networks [26]. To solve that, Bottinelli et al. [14] proposed a novel gene duplication

based evolution model that enables to simultaneously simulate the formation of protein modules during evolution the PPI network. Furthermore, there are many other models tried to uncover the evolution mechanisms from different perspectives, e.g. evolution model learned by data-driven approach [69], network evolution based on parsimony approach [68], network growth and the spectral evolution model [55], and kinetic evolution model [70]. However, there are still many controversies about the underlying evolution mechanisms for PPI networks; especially, PPI network of different species may evolved in totally different ways. Therefore, further indepth studies of evolution model are still needed.

On the other hand, with the increasing number of evolutionary models, it has become urgent to develop accurate analysis methods for evaluating the fitness of evolutionary models. Traditionally, researchers would like to evaluate the difference between simulation network and observed network using metrics on the basis of topological features [100], such as node degree [46, 67], betweenness [48], modularity [77], diameter [92], clustering coefficient [75], assortativity [61, 95] and so on. While these metrics are useful for their specifically designed purposes, it remains difficult to describe the PPIs network as a whole in terms of the summary statistics for the noise and incompleteness. To deal with this problem, most recently, Thorne, T. and Stumpf, M. P. [89] developed Approximate Bayesian computation and sequential Monte Carlo method (ABC-SMC) to do graph spectral analysis, which compares the simulated networks and observed networks comprehensively using graph spectral analysis and enables model selection and parameter estimation over a number of network evolutionary models. It has been demonstrated that the graph spectra based ABC-SMC can capture network data more naturally than the traditional summary statistics. However, it cannot differentiate similar models accurately, especially for these duplication-divergence based models. Moreover, for each time, the sequential Monte Carlo sampling based ABC-SMC needs to choose a proper threshold value ϵ that is used to accept or reject a particle, which highly increase the computational complexity. Thus, developing more efficient and accurate methods for evaluating the fitness of evolutionary models is still a urgent

problem need to be solved.

1.3.2 Network Level PPI Prediction

Generally, current computational methods for PPI prediction can be classified into two categories: (a) pair-wise biological similarity based methods, and (b) network level based methods. For category (a), computational approaches have been developed to predict if any given pair of proteins interact with each other, based on various properties such as sequence homology, gene co-expression and phylogenetic profiles, etc. [78, 21, 29]. Moreover, some previous work also demonstrated that three-dimensional structural information, when available, can be used to predict PPIs with accuracy superior to predictions based on non-structural evidence [102, 80]. However, with no first principles to tell deterministically yet if two given proteins interact or not, the pair-wise biological similarity based on various features and attributes can run out its predictive power, as often the signals may be too weak or noisy. Therefore, recently many researches have been focused on integrating heterogeneous pair-wise features, e.g. genomic features, semantic similarities and etc., in seek of better prediction accuracy [24, 85, 19, 49]. It is biological meaningful if we can disentangle the relations among various pair-wise biological similarities and PPIs, but it is still in early stage for the incomplete and noisy pair-wise similarity kernels.

To circumvent the limitations with using pair-wise biological similarity, efforts have also been made to investigate PPI prediction in the context of networks, which may provide extra information to resolve ambiguities incurred at pairwise level. A network can be constructed from reliable pair-wise PPIs, with nodes representing proteins and edges representing interactions. Topological features, such as the number of neighbors, can be collected for nodes and then are used to measure the similarity for any given node pair to make PPI prediction for the corresponding proteins [18, 59, 57, 73]. Meanwhile, inspired by the PageRank algorithm [63], variants of random walk based methods have been proposed to go beyond these node centric topological features to get the whole network involved: the probability of interaction between given two proteins

is measured in terms of how likely a random walk in the network starting at one node will reach the other node [90, 58, 6]. These methods are suitable for PPI prediction in cases when the task is to find all interacting partners for a particular protein, by using it as the start node for random walks. The computational cost increases from $O(\beta N)$ to $O(\beta N^2)$ for all-against-all PPI prediction for a PPI network with N proteins. To overcome the limitation of single start-node random walk, many kernels on network for link prediction and semi-supervised classification have been systemically studied [28], which can measure the random-walk distance for all node pairs at once. Compared with the random walk methods, kernel methods are obviously more efficient and applicable to various network types. But both the variants of random walk and random walk based kernels cannot differentiate faraway interacting candidates well. Besides, instead of computing proximity measures between nodes from the network structure directly, Kuchaiev et al.and Cannistraci et al. proposed geometric de-noise methods that embed PPI network into a low-dimensional geometric space, in which protein pairs that are closer to each other represent good candidate interactions [53, 16].

Moreover, when the network is represented as adjacent matrix, the prediction problem can be transformed into spectral analysis and matrix completion problem. For example, Symeonidis et al. [86] did link prediction for biological and social networks based on multi-way spectral clustering. Wang et al. [96] and Krishna et al. [62] predicted PPI interactions through matrix factorization based methods. By and large, the prediction task will be reduced to convex optimization problem, and the performance depends on the objective function, which should be carefully designed to ensure fast convergence and avoidance of being stuck in the local optima.

Actually, the two kinds of methods, pair-wise biological similarity based methods and network level based methods, can be mutually beneficial. For example, weights can be assigned to edges in the network using pair-wise biological similarity scores. In Backstrom et al. [6] a supervised learning task is proposed to learn a function that assigns weighted strengths to edges in the network such that a random walker is more likely to visit the nodes to which new links will be created in the future. The matrix

factorization based methods proposed by Wang et al. [96] and Krishna et al. [62] also included multi-modal biological sources to enhance the prediction performance. In these methods, however, only the pair-wise features for the existing edges in the network will be utilized, even though from a PPI prediction perspective what is particularly useful is to incorporate pair-wise features for node pairs that are not currently linked by a direct edge but will if a new edge (PPI) is predicted. Therefore, it would be of great interest if we can infer PPI network directly from multi-modal biological features kernels that involve all node pairs. In Yamanishi et al. [98] a method is developed to infer protein networks from multiple types of genomic data based on a variant of kernel canonical correlation analysis. In that work, all genomic kernels are simply added together, with no weights to regulate these heterogeneous and potentially noisy data sources for their contribution towards PPI prediction.

The last but not the least, almost all the methods discussed above need connected training networks to measure node-pair similarities, although existing PPI networks are usually very sparse and disconnected. Consequently, these traditional methods only keep the maximum connected component of the original PPI network as golden standard data which is divided as a connected training network and testing edges. That is to say, these methods cannot effectively predict interactions for proteins that are not in the maximum connected component. Therefore, it is of great interest and making more sense if we can infer PPI network from a small amount of interaction edges that do not need to form a connected network.

1.4 Our Contributions

In this dissertation, we contribute to finding solutions for challenges that are discussed in the section 1.3, which generally including:

- How to develop effective and reliable evaluation method for various PPI network evolution models?
- How to utilize evolution analysis to help predict de novo PPIs?
- How to infer PPI network from multiple heterogeneous data sources?

- How to accurately and efficiently find optimal weights for those multiple heterogeneous data?
- How to infer de novo PPIs based only on a minimal disconnected training network?

First of all, we develop a novel method based on Approximate Bayesian Computation and modified Differential Evolution algorithm (ABC-DEP) that is capable of conducting model selection and parameter estimation simultaneously and detecting the underlying evolutionary mechanisms for PPI networks more accurately. Then we propose a method that enables us to incorporate evolutionary information into geometric space to improve PPI prediction, which in turn can be used to select and evaluate various evolutionary models.

We next focus on finding optimal weights for heterogeneous data to build kernel fusion, thereby improving PPI prediction performance. Two methods are developed to solve this problem: (a) The method based on our revised ABC-DEP; (b) The method based on Barker algorithm, random walk and linear programming. Both of those two methods adopt Regularized Laplacian (RL) kernel to make prediction. The former show that the accuracy of PPI prediction measured by AUC is increased by up to 23%, as compared to a baseline without using optimal weights. The latter can learn optimal weights more efficiently while without obvious loss on prediction performance.

Finally, we develop novel PPI prediction method based on deep learning neural network and regularized Laplacian kernel. We use the neural network to implicitly simulate and guide the evolution process of a PPI network. After the training step, an evolved PPI network can be obtained from the extremely sparse and disconnected network. Then we can predict PPIs by applying the regularized Laplacian kernel to the trainsition matrix that is built upon the evolved PPI network. The results show the prediction performance can be effectively improved, although only a really small amount of interaction edges have been used for training. Moreover, the evolved PPI network obtained by the neural network can also help us to leverage complementary

information from the disconnected training network and multiple heterogeneous data sources, which can further improve the prediction performance.

1.5 Protein Datasets

In this section, we list some publicly accessible PPI databases and web-services that related to our research.

- STRING: STRING is a database of known and predicted protein interactions. The interactions include direct (physical) and indirect (functional) associations. http://string-db.org
- DIP: DIP database catalogs experimentally determined interactions between proteins. It combines information from a variety of sources to create a single, consistent set of protein-protein interactions. http://dip.doe-mbi.ucla.edu/dip/
- BioGRID: BioGRID is an interaction repository with data compiled through comprehensive curation efforts.http://thebiogrid.org/
- Ensembl:Ensembl is one of several well known genome browsers for the retrieval of genomic information. http://www.ensembl.org
- GenBank:GenBank is the NIH genetic sequence database, an annotated collection of all publicly available DNA sequences. http://www.ncbi.nlm.nih.gov/genbank
- Gene Ontology (GO) Database: The GO Consortium provides an ontology of defined terms representing gene product properties, including cellular component, molecular function, biological process. http://www.geneontology.org
- Pubmed: PubMed is a search engine accessing primarily the MEDLINE database of references and abstracts of papers on life sciences and biomedical topics. http://www.ncbi.nlm.nih.gov/pubmed
- HPRD: The Human Protein Reference Database (HPRD) includes Human PPI data that has been manually extracted from the literature by expert biologists. http://www.hprd.org/index.html
- PrePPI: PrePPI is a database that combines predicted and experimentally determined PPIs using a Bayesian framework. http://bhapp.c2b2.columbia.edu/PrePPI

Chapter 2

EVOLUTIONARY MODEL SELECTION AND PARAMETER ESTIMATION FOR PROTEIN-PROTEIN INTERACTION NETWORK BASED ON DIFFERENTIAL EVOLUTION ALGORITHM

2.1 Introduction

Revealing the underlying evolutionary mechanism plays an important role in understanding protein interaction networks in the cell. While many evolutionary models have been proposed, the problem about applying these models to real network data, especially for differentiating which model can better describe evolutionary process for the observed network remains a challenge. The traditional way is to use a model with presumed parameters to generate a network, and then evaluate the fitness by summary statistics, which however cannot capture the complete network structures information and estimate parameter distribution.

In this chapter, we develop a novel method based on Approximate Bayesian Computation and modified Differential Evolution algorithm (ABC-DEP) that is capable of conducting model selection and parameter estimation simultaneously and detecting the underlying evolutionary mechanisms for PPI networks more accurately. We test our method for its power in differentiating models and estimating parameters on simulated data and found significant improvement in performance benchmark, as compared with a previous method. We further apply our method to real data of protein interaction networks in human and yeast. Our results show Duplication Attachment model as the predominant evolutionary mechanism for human PPI networks and Scale-Free model as the predominant mechanism for yeast PPI networks.

In the following sections, we firstly give a detailed introduction about our method. Then we demonstrate the accurateness, robustness, and reliability by testing

ABC-DEP based on simulation networks. To show the promising ability of ABC-DEP, we apply it to PPIs network downloaded from PrePPI database [102, 101]. Finally, we conclude by discussing the results and the significance of our method.

2.2 Methods

In this section, we introduce several key parts of our method, and then outline ABC-DEP framework.

2.2.1 Approximate Bayesian Computation (ABC)

Given an observed PPI network, D, and a set of evolutionary models m_i with parameters θ, we develop an efficient method that can carry out model selection and parameter estimation simultaneously to detect the underlying evolutionary mechanism. Being a probabilistic approach, our method is based on the Bayesian analysis to compute the posterior probability of any model m_i, given a network D :

$$p(m_i(\theta)|D) = \frac{p(D|m_i(\theta))p(m_i(\theta))}{p(D)} \tag{2.1}$$

Where $p(D|m_i(\theta))$ is the likelihood, $p(m_i(\theta))$ is the prior, and $p(D)$ is the evidence. The prior $p(m_i(\theta))$ is assumed to be known and is often specified by choosing a particular distribution; here the uniform distribution has been chosen for our method. However, it is often difficult, or even practically impossible to directly evaluate the likelihood $p(D|m_i(\theta))$. In this work, we choose to evaluate the likelihood using approximate Bayesian computation (ABC) instead. ABC based methods estimate the likelihood by simulations whose outputs (simulation network) are compared with the observed network [22]. More specifically, a set of parameters for a certain model is sampled through a presumed prior distribution. The model and its parameters form a so-called "particle" $m_i(\theta)$ with which we can simulate a network D'. The distance between simulation network D' and the observed network D is computed in order to accept or discard this particle. If the distance is smaller than a pre-set threshold, the sampled

particle will be accepted, otherwise, will be discarded. The basic formula can be given by:

$$p(m_i(\theta)|D) \approx p(m_i(\theta)|d(D', D) < \epsilon) \tag{2.2}$$

Where ϵ represents the threshold to judge the distance, and $d(D', D)$ represents the distance between network D' simulated by particle $m_i(\theta)$ and observed network D.

2.2.2 Distance Computing Method for Networks

As described in last subsection, we need to evaluate the distance between the simulated network and the observed network. To begin with, we represent a network by adjacency matrix that is supposed to capture all the structure information of the network, if only implicitly. Given a network with N nodes and E edges, the corresponding adjacency matrix A with $N \times N$ dimension can be given by:

$$a_{i,j} \begin{cases} 1, if(i,j) \in E \\ 0, if(i,j) \notin E \end{cases} \tag{2.3}$$

Where i and j are two nodes in the nodes set N, and (i, j) represents an edge between i and j, $(i, j) \in E$. Suppose the simulation network D' and the observed network D are represented by matrices A and B respectively. In theory, the distance between A and B can be computed as follows:

$$d(A, B) = \sum_i \sum_j (a_{i,j} - b_{i,j})^2 \tag{2.4}$$

where $a_{i,j}$ and $b_{i,j}$ are elements in matrix A and B. While this node-wise comparison of two networks may be as comprehensive as it can be, it will not be very useful if the correspondence between the nodes from the two networks is not fixed – as the distance will likely be different for any different mapping of nodes. Indeed, the PPI networks are usually unlabeled and undirected, which means the actual mapping between the simulation network and observed PPI network is not known or easily attainable. In theory, a sensible remedy is to use either the minimum distance or the average distance among all possible mappings. However, considering that the PPI networks are

with large size while very sparse, for example, the human high-confidence PPI network [102, 101] has 4003 nodes and 6780 edges, so it is extremely expensive to get either the minimum or the average distance between the observed PPI network and the corresponding simulation network by permuting all possible mapping strategies between them.

Therefore, we adopt the theorem of Umeyama [94] by which the approximate lower bound on the edit distance between two networks can be obtained. It has been demonstrated in Wilson and Zhu [97] that the lower bound on edit distance is an excellent approximation of edit distance between two networks, as defined in Eq. (2.4). The measure formula is shown by

$$d(A, B) = \sum_i (\alpha_i - \beta_i)^2 \tag{2.5}$$

Where A and B are Hermitian matrices, and α_i and β_i are their ordered eigenvalues respectively. We will further demonstrate the reliability of 2.5 when we do distance analysis in the later subsection.

2.2.3 Differential Evolution Algorithm

Differential evolution (DE) is a population based, stochastic function optimizer, which is shown to be among the best genetic type of algorithms for solving the real-valued test function suite of the First International Contest on Evolutionary Computation [74]. It has been widely applied to optimization problems of different kinds in various research fields. DE has been adopted as the foundation of our ABC-DEP algorithm for its efficiency, accuracy and reliability. Briefly, the central idea behind DE is a self-organizing scheme for generating trial parameter vectors by mutation and crossover, and then the trial vector will be accepted or rejected based on an objective function. Fig. 2.1 shows the more detailed process of DE algorithm. Given a population of particles $P_{x,g}$ each of which consists of an object function $f()$ and a parameter vector $X_{N_i-1,g}$ $(i = 0, 1...N_p - 1)$, a target vector $X_{N_i-1,g}$ $(i = 0, 1...N_p - 1)$, a randomly chosen base vector $X_{r0,g}$ and other two different random vectors $X_{r1,g}$ and $X_{r2,g}$ are

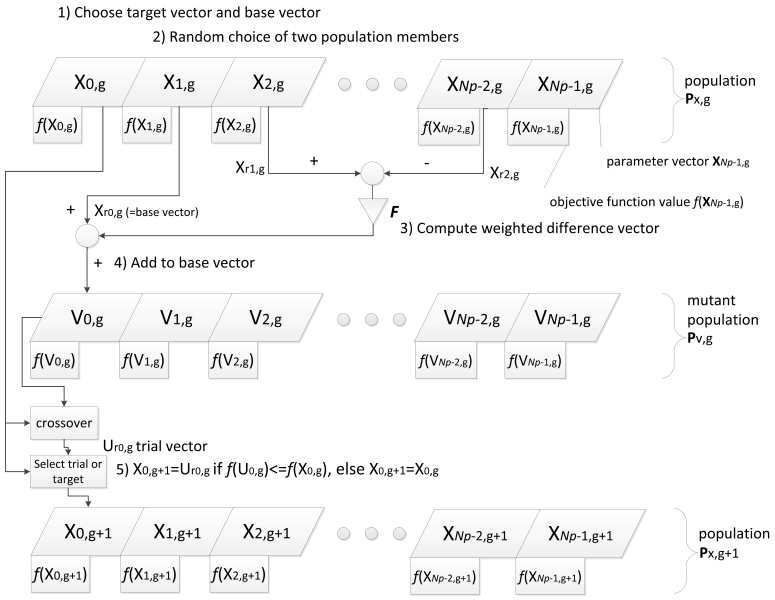


Figure 2.1: Flowchart of DE algorithm [74].

needed to do mutation, which adds the weighted difference between the two random vectors to the base vector. After that, a crossover between the mutant vector $V_{N_i-1,g}$ $(i = 0, 1...N_p - 1)$ and the target vector $X_{N_i-1,g}$ $(i = 0, 1...N_p - 1)$ is used to generate a trial vector $U_{N_i-1,g}$ $(i = 0, 1...N_p - 1)$. Finally, a choice between target vector $X_{N_i-1,g}$ and trial vector $U_{N_i-1,g}$ is made by evaluating their objective function values to get the new particle $X_{N_i-1,g+1}$ for the next generation. Typically, the whole process needs to be repeated multiple times in order to get the optimization output. It should be noted that, inspired by the method of Toni and Stumpf [91] and Thorne and Stumpf [89], the models and parameters are treated analogously, i.e., the different models are encoded as another parameter, which is concatenated with the parameter vector, enabling model selection and parameter estimation simultaneously to be carried out.

2.2.4 ABC-DEP for Model Selection and Parameter Estimation

As mentioned in the previous section, DE is an excellent method for solving the optimization problem. However, the problem we need to solve is model selection and parameter estimation by evaluating the posterior probability, which is based on importance sampling. We make a population of two-tuple particles, each of which

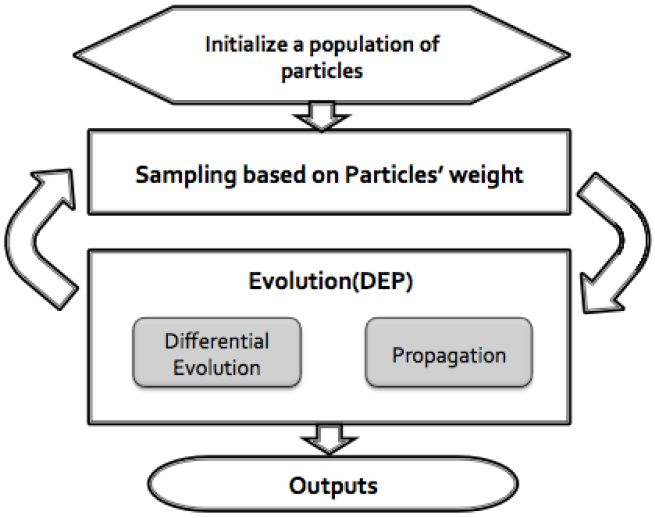


Figure 2.2: ABC-DEP process.

consists of a certain model and its parameter vector. The DE algorithm may help find some good particles, but what we actually need is the posterior distribution of particles. To address this issue, as illustrated in Fig. 2.2, we propose another evolution kernel, propagation, and combine it with DE. Note that the terms evolution, mutate, population, etc., used in the section are in the context of evolutionary computation, and are not in the context of PPI network evolution, for which the method is designed to identify the optimal one among a set of candidate models.

2.2.4.1 Initialization

To do initialization, we randomly choose one out of the six evolutionary models and then randomly assign values from a preset range to the parameters for this model, and make the model and its parameters into a particle, with which we can generate a simulated network D'. Next, we evaluate the distance between the simulation network and the observed network D. If $d(D', D) < \epsilon$, we accept this particle and assign it an initial weight $w = 1$, otherwise, it is discarded. Here ϵ is a threshold used to control the quality of particles in a fixed scale of population. Different from ABC-SMC[19] that highly depends on a set of thresholds ϵ to guarantee the accuracy of final posterior probability, we only need one threshold ϵ for the initialization. The

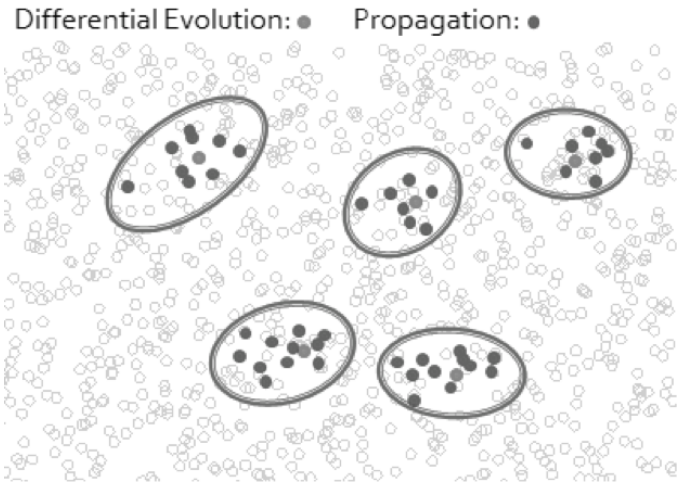


Figure 2.3: Functions of DE and Propagation.

initialization process will be carried out repeatedly until N particles have been accepted into the population.

2.2.4.2 Evolution

Once N particles have been initialized, the procedure goes into the evolution part that includes two kernels: DE and Propagation. As we mentioned in previous sections, we need to randomly select other three different particles for the target particle to do mutation. However, the DE kernel tends to reduce the diversity of the population after several iterations, making it difficult to find other three different particles with the same model but different parameters, which are used to form a mutant to crossover with the target particle. Whenever this happens, ABC-DEP will switch into the propagation kernel: a multivariate Gaussian distribution with zero mean and a proper variance is used to perturb the parameter vector of any target particle for which no three other different particles can be found to form a mutant. As illustrated in Fig. 2.3, while DE can help find several optimized particles, propagation enables us to find some good neighbors around the optimized particles and diversify the population. Each particle in the population will be selected at one time as a target particle to do evolution. Then, we can obtain a trial population that has the same size of the target population. For every pair of particles, one from each of these two populations, we will use them

to generate a simulated network respectively. Suppose network D'_{P^i} is simulated by a target particle P^i, and D'_{Z^i} is simulated by the trial particle Z^i, where Z^i is evolved from P^i. Then ABC-DEP adopts Metropolis-hasting acceptance scheme to determine whether the trial particle Z^i should be accepted to replace the target particle P^i. The probability for acceptance in the Metropolis-hasting scheme can be evaluated as follows.

$$min(1, \frac{d(D, D'_{P^i})t(P^i|Z^i)}{d(D, D'_{Z^i})t(Z^i|P^i)}) \tag{2.6}$$

where the distance $d(D', D)$ is used to evaluate the fitness of simulation network, and $t(b|a)$ represents the transfer probability from a to b. Here we simplify the problem by not differentiating the propagation direction. That is, $t(Z^i|P^i)$and $t(P^i|Z^i)$ are equal; they are given by the following formula.

$$\begin{aligned} &t(Z^i|P^i) = \\ &t(P^i|Z^i) = CR\sum_j \sum_k \sum_f P^j.\theta + F(P^k.\theta - P^f.\theta) \end{aligned} \tag{2.7}$$

CR is the crossover probability and F is the weighting factor in DE. On the basis of (6) and (7), therefore, the Metropolis-hasting acceptance probability is simplified as follows.

$$min(1, \frac{d(D, D'_{P^i})}{d(D, D'_{Z^i})}) \tag{2.8}$$

The trial particle may be accepted with the probability given in Eq. (2.8) to replace the target particle; otherwise, the target particle will be kept for the next generation. Next, ABC-DEP updates particle's weight by the method shown in Algorithm DEP. The importance of updating weight by multiplying $\frac{\epsilon}{D',D}$ is to incorporate the fitness of the particle, namely, the better (i.e., with a smaller d) the simulated network is, the higher the weight of that particle is. Unlike the ABC-SMC [89] which may inefficiently try hundreds or thousands times to get a satisfying particle for the continuously strict acceptance threshold ϵ and which is a problem that becomes especially serious during the last few iterations, our method only need one trial to select a particle based on Eq. (2.8) for the next generation.

2.2.4.3 Sampling

Before sampling, the weights of the N_p particles through evolution are normalized first. Then a model's intermediate posterior probability can be obtained by adding the weights of its particles. For instance, there are n particles that belongs to model i, $n < N_p$ then the model i's posterior probability is given by

$$Pro_{model[i]} = \sum_{j=1}^{n} w^j \tag{2.9}$$

During sampling, a model is selected based on its intermediate posterior probability, and then for the selected model a specific particle is chosen based on the particle's weight, i.e., the particle with higher weight and belongs to a model with higher posterior probability will get more chance to survive for the next generation. Besides, in order to prevent a certain model from being extinct during the model selection, there is a probability p to stay at any selected model, and a probability $1 - p$ to jump randomly to other models. Therefore, we assign the new sampled particle with a weight by

$$w_{particles \in model[i]} = \\ p \times Pro_{model[i]} + (1-p) \sum_{j \neq i} Pro_{model[j]} \tag{2.10}$$

where $(1 - p)$ represents the transfer probability. This sampling procedure should be continued until N_p particles have been sampled. Together, the evolution and sampling steps are repeated without decreasing the acceptance threshold ϵ until converged model posterior probabilities are obtained.

2.3 Results and Discussion

We test our method extensively on simulated data first, as they, unlike the real PPI networks, offer ground truth and controllability for reliable assessment of performance. To compare with the work reported in T. Thorne and M. P. H. Stumpf [89], the same six evolution models are included in our experiments: Duplication Attachment (DA), Duplication Attachment with Complementarity (DAC), Linear Preferential Attachment (LPA) [11], the general Scale Free (SF) [25], Combination model of DAC

Algorithm 1 ABC-DEP

Input: $Model_i \leftarrow evolution\ models$
$M \leftarrow iterations\ times$
$N \leftarrow particles$
while $t \leq M$ **do**
if $t = 1$ **then**
Initialize N particles satisfy $d(D, D') < \epsilon$
$P_t, W_t \leftarrow \{P^i, W^i\}_{i=1}^N$ // P^i is a particle, W^i is the weight of P^i. P_t, W_t represents the t^{th} generation of particles and weights.
else
$\{P_t, W_t\}_{i=1}^N \leftarrow Sampling((P_{t-1}, W_{t-1}))$
end if
$(P_{t+1}, W_{t+1}) \leftarrow DEP(P_t, W_t)$
$t \leftarrow t + 1$
end while
$Normalize((P_M, W_M))$
$PosteriorPro(Model_i, (P_M, W_M))$ // Cacluating posterior probability for each model.

and LPA (DACL) and DAC model with random edges addition(DACR). The experiments based on simulated data aim to evaluate how accurately ABC-DEP can detect the underlying model that is used to simulate the testing networks. Finally, we apply ABC-DEP to analyze the possible evolutionary mechanism for real PPI networks.

2.3.1 Results based on the Simulated Data

DACL and DACR are the most similar pair among the six models included in our experiment, making them the hardest to differentiate from each other. In Thorne and Stump [89], networks generated by DACR were used as the target to see if they could be detected correctly or would be mistaken as DACL. Here we did the same test for comparison. In addition, we also did the test in the reversed direction, namely, using the networks generated by DACL as the target to see how well ABC-DEP can detect and differentiate them from DACR and other models.

Algorithm 2 DEP

Input: Population with size N_p

for $i = 1$ to N_p **do**

Randomly select P^f, P^j, P^k where $i \neq j \neq k \neq f$ and $P^i.m = P^j.m = P^k.m = P^f.m$

// P^i is the target particle, P^f, P^j and P^k are three randomly selected particles. $P^i.m$, $P^j.m$, $P^k.m$ and $P^f.m$ represent particles' models. $P^i.\theta$, $P^j.\theta$, $P^k.\theta$ and $P^f.\theta$ represent particles' parameter vectors.

if $P^i.\theta = P^j.\theta = P^k.\theta = P^f.\theta$ **then**

$Z^i = Propagation(P^i)$

else

$Z^i = DifferentialEvolution(P^i, P^j, P^k, P^f)$

end if

end for

for $i = 1$ to N_p **do**

Simulate D'_{Z^i} by particle Z^i and D'_{P^i} by particle P^i // D is the observed network, D'_{Z^i} and D'_{P^i} are simulated networks. $d(,)$ is the distance.

if $rand(0,1) \leq min(1, \frac{d(D,D'_{P^i})}{d(D,D'_{Z^i})})$ **then**

if $\frac{d(D,D'_{P^i})}{d(D,D'_{Z^i})} \geq 1$ **then**

$W^{P^i} \leftarrow W^{P^i} \times \frac{\epsilon}{d(D,D'_{Z^i})}$

else

$W^{P^i} \leftarrow W^{P^i} \times \frac{d(D,D'_{P^i})}{d(D,D'_{Z^i})} \times \frac{\epsilon}{d(D,D'_{Z^i})}$

end if

$P^i \leftarrow Z^i$

else

$P^i \leftarrow P^i \quad W^{P^i} \leftarrow W^{P^i} \times \frac{\epsilon}{d(D,D'_{P^i})}$

end if

end for

$Normalize(P^i, W^{P^i})$

$Prob_{model}(P^i, W^{P^i})$ // Cacluating intermediary probability for each model.

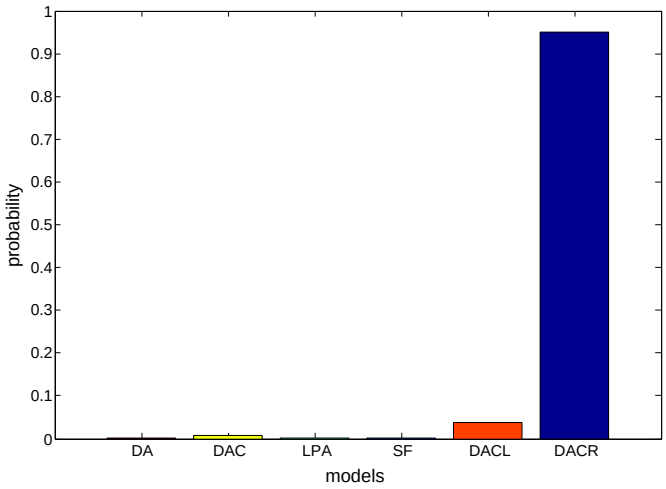
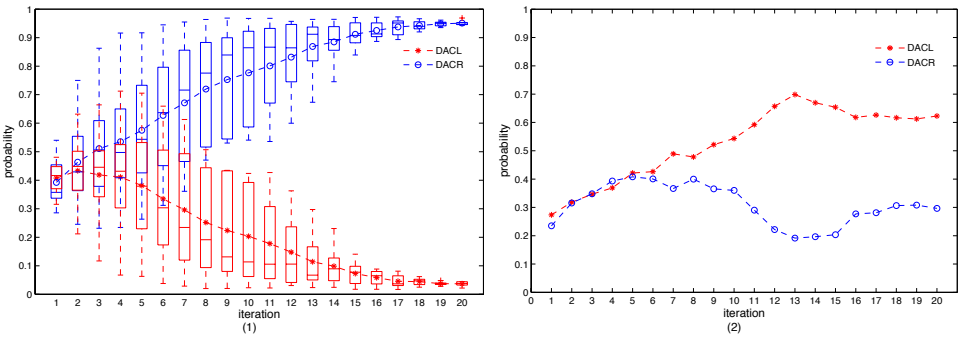


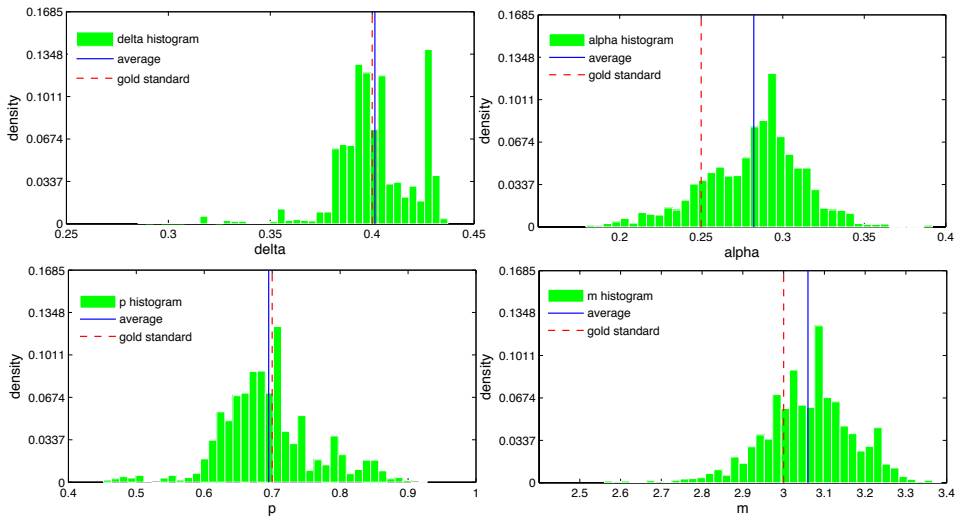
Figure 2.4: Posterior probabilities based on test data simulated by DACR.

2.3.1.1 Data Simulated by DACR

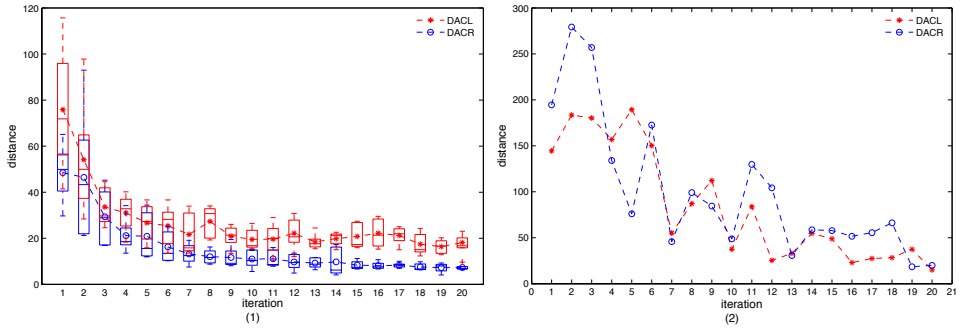
The first test data is simulated by DACR, with the parameters being set at the following values: $\delta = 0.4$, $\alpha = 0.25$, $p = 0.7$ and $m = 3$. The network sizes are controlled at 5000 nodes with 25009 edges. The posterior model probabilities illustrated in Fig. 2.4 show that the DACR has the highest average probability, which means ABC-DEP can accurately detect the model that is used to generate the test data. From the boxplot in Fig. 2.5(a)(1), we see that DACR's posterior probabilities converge rapidly and smoothly towards their expected values respectively. Contrast to the traditional method ABC-SMC [89] which mistakenly predicted DACL as the underlying model for networks generated by DACR, our method is obviously more accurate and detects the target correctly. Moreover, for the parameter estimation of DACR shown by Fig. 2.5(b), where the vertical line means the average value of our estimation and the vertical dashed line means the gold standard value used to generate the test network, the parameter distribution, although not very smooth, is almost centered around the correct values. The standard deviations for the estimation of δ, α, p and m are 0.0196, 0.0300, 0.1114 and 0.0700 respectively. And the range of parameters' distribution of our method is narrower than that of ABC-SMC [89]. Our experiments suggest that smoother distribution can be attained by increasing the number of particles, which can



(a) Comparison of posterior probability converging process between our method(1) and ABC-SMC(2) [89].



(b) Parameter estimation based on our method.



(c) Comparison of minimum distance converging process between our method(1) and ABC-SMC(2) [89].

Figure 2.5: Results based on testing data simulated by DACR.

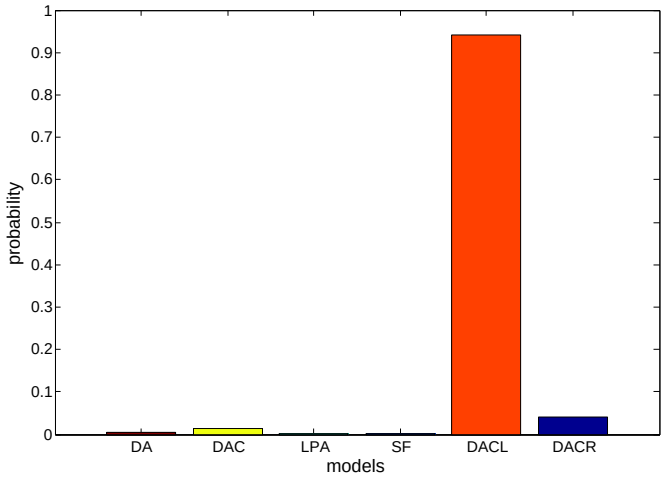
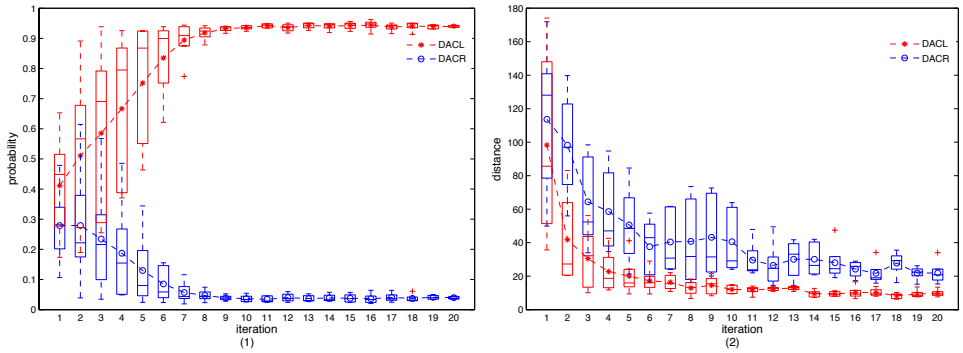
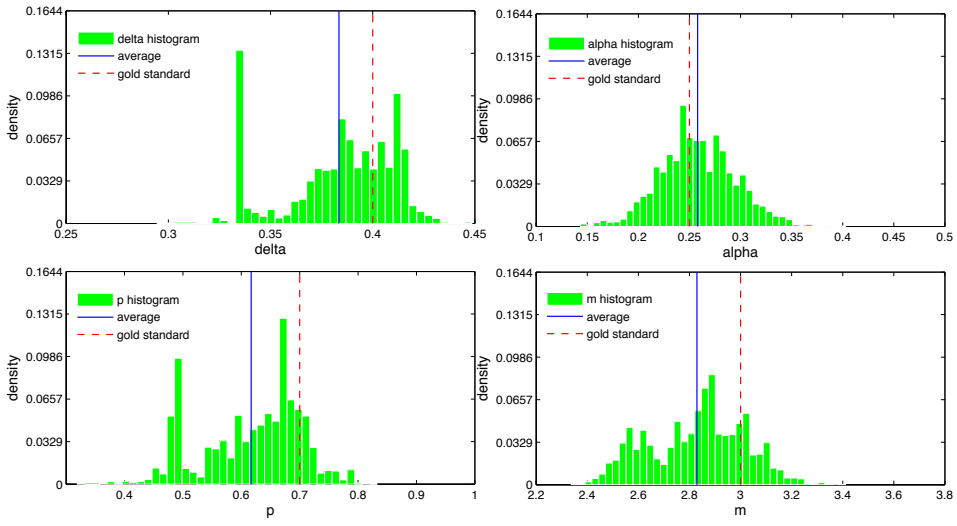


Figure 2.6: Posterior probabilities based on test data simulated by DACL.



(a) Coverging process of porsterior probability(1) and minimum distance(2).



(b) Parameter estimation based on our method.

Figure 2.7: Results based on testing data simulated by DACL.

result in a sample mean closer to the actual value, though at additional computational cost. We therefore strike a balance between the accuracy and computational complexity. Besides, we have reproduced the traditional method ABC-SMC [89] to make further comparison. Note that the posterior probability from ABC-SMC incorrectly converges towards DACL for test networks generated by DACR as illustrated in Fig. 2.5(a)(2). In additional to posterior probability, we also analyzed the minimum distance, as measured by Eq. (2.5), between the simulated network and the test network, as shown in Fig. 2.5(c). Our results show a better convergence; and also the minimum distance obtained by our method can reach a smaller value.

2.3.1.2 Data Simulated by DACL

To further demonstrate the robustness and accuracy of our method, we test our method by an additional simulated test data. This second test data is simulated by DACL, also with parameters $\delta = 0.4$, $\alpha = 0.25$, $p = 0.7$ and $m = 3$, and grown to 5000 nodes with 25009 edges. Fig. 2.6 illustrates the posterior probabilities of six models, and Fig. 2.7(a)(1) and (2) illustrate the converging processes of posterior probabilities and minimum distances respectively. And the histogram of parameter estimation of DACL shown by Fig. 2.7(b) is quite smoothly distributed and centered around the correct values, with standard deviations 0.0270, 0.0364, 0.1866 and 0.0861 for the estimation of δ, α, p and m respectively.

2.3.2 Results based on Protein Interaction Data

We then apply ABC-DEP to high-confidence human PPI network and high-confidence yeast PPI network that were downloaded from PrePPI database [102, 101], where high-confidence means the interactions in the datasets are at least supported by two publications. As a quick summary, the high-confidence human PPI network has 4003 nodes and 6781 edges, and the high-confidence yeast PPI network has 3236 nodes and 11381 edges. The results illustrated in Fig. 2.8 shows the DA is the predominant evolutionary mechanism for the high-confidence human PPI network, while SF is the

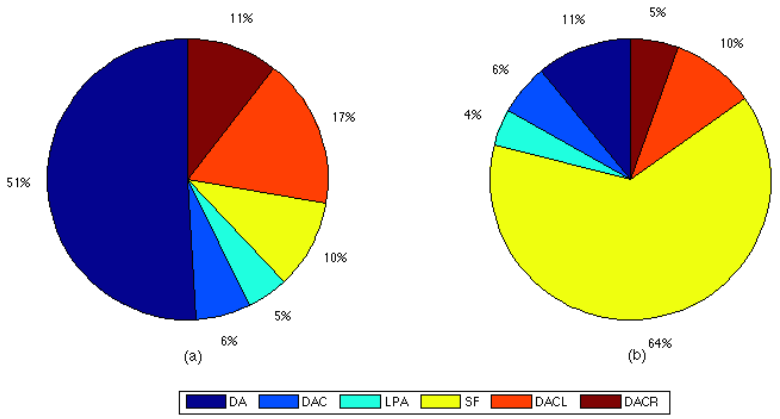
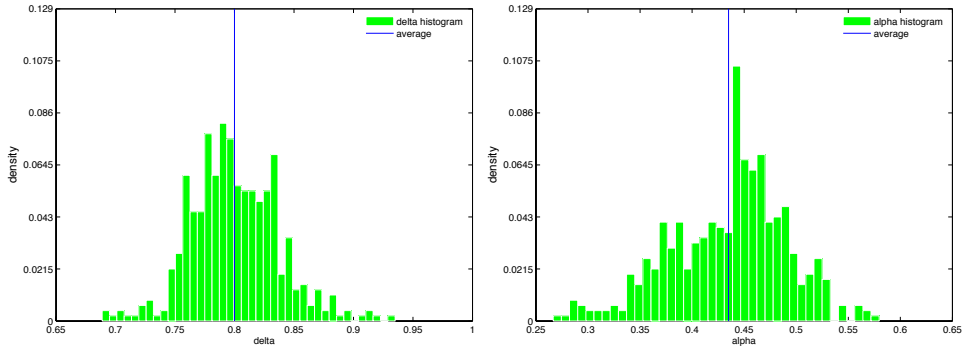
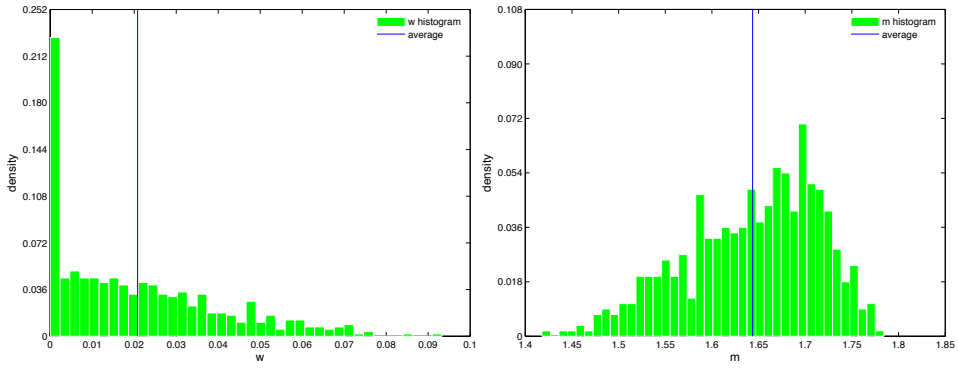


Figure 2.8: Evolutionary model prediction for real PPI networks.(a) is for Human, (b) is for Yeast.



(a) Parameter estimation for Human based on DA.



(b) Parameter estimation for Yeast based on SF.

Figure 2.9: Parameters estimation for Human and Yeast based on their preferred models.

predominant evolutionary mechanism for the high-confidence yeast PPI network. For these two models, the parameter estimations are shown by Fig. 2.9(a) and Fig. 2.9(b) respectively. The parameter estimation of w shown by Fig. 2.9(b) is presumed $>$ 0, that is why the distribution looks a little different. Generally, the center-around parameter distribution can help researchers to choose most suitable parameters for the evolutionary models.

While scale-free is an important topology property of PPI network [46, 8], not all scale-free PPI networks are born equal [36]. Considering that our graph spectra based method can capture many aspects of network structure, so the simulation networks generated by different evolutionary models may be somewhat scale-free overall, and yet different in other topological properties such as betweenness, modularity, clustering coefficient and so on, a phenomenon reported in recent literature [48]. In this regard, our finding of DA as the predominant mechanism for high-confidence human PPI network is consistent with other recent studies [47]. There is currently no consensus with respect to topological characteristics of yeast PPI networks; while some reported scale-free [3], others did not [31].

As duplication is a main mechanism via which genomes evolve and gain bigger sizes, it is plausible that association can be detected between genome size and PPI network growth involving duplication-divergence (DA), and that such association may be more pronounced for genomes of bigger sizes, like human genome, although an affirmative answer requires more detailed investigation and is beyond the scope of the current paper. The framework of our approach is whole network comparison, as formulated in Eq. (2.4) and approximated in Eq. (2.5), other than based on summary statistics, such as size, density and other topological features. There are pros and cons for these two types of approaches, i.e., whole-network vs summary statistics. While the whole-network approach is capable of comparing the simulated networks against the observed network comprehensively instead of only against a selected set of features thus avoiding potential biases due to selection of particular topological features, it then does not provide an easy interpretation of the results in terms of specific network

features, such as size and density.

Given a real PPI network, although there is no guarantee to find the true underlying evolution model – as it may not be one of the candidate models, our method offers an effective way to find the relatively better one among the candidate models in matching the data, as shown in pie charts in Fig. 2.8. In reality, it is plausible that the true underlying evolutionary mechanism is either too complex to be captured in a single model thus an ensemble of models would be more appropriate to provide an equivalent representation, or is indeed just a mixture of multiple sub mechanisms. Either way, we believe these pie charts in Fig. 2.8 should provide useful insights. Note that the existence of a predominant model, as suggested by the pie chart, tend to suggest that we are capturing something real, in contrast to a scenario where all models get about the equal share, which may be more of a sign for random data. Also, the method is inclusive. A new proposed evolution model can be easily included with our encoding scheme to participate in simulation and be compared with other models.

The time cost of simulation process mainly depends on the number of particles (i.e., simulated networks), and the number of iterations, and the size of the observed network. Generating a simulated network is linear time with regard to the network size. The main cost come from calculating eigenvalues for adjacency matrices, which is in $O(n^3)$, where n is the number of nodes in the network. In our experiments, we have used the CULAtools GPU linear algebra library (www.culatools.com) to perform the matrix calculations. Compared to a conventional CPU implementation, its parallel architecture greatly improved the efficiency. The parameter space for a given evolution model can also significantly affect the computational complexity. But in practice, the parameter spaces are usually kept at the range of $2 \sim 5$. Moreover, to follow the principle of Occams razor, it is also better to use as fewer parameters as possible when we design a new model. When there are many candidate models, we can deploy the divide-and-conquer strategy or tournament scheme, i.e., we divide them into several groups and do model selection to get a representative model for each group, and identify the best model through the tournament. The whole process is highly parallelizable.

The importance of our method is to help researchers analyze PPI network comprehensively from the evolutionary perspective, and provide a reference for them to find a possible evolutionary model given existing PPI networks.

2.4 Conclusions

We have developed a novel model selection and parameter estimation method, ABC-DEP, based on Approximate Bayesian Computation and modified Differential Evolution. The results based on simulated data illustrate the efficacy of ABC-DEP. Detailed comparisons between our method and T. Thorne and M. P. H. Stumpf [89] have been made, which shows ABC-DEP has competitive advantages in accuracy and efficiency. Furthermore, we applied our method to real PPI networks data from human and yeast. The results show that DA model is the predominant evolutionary mechanism for the high-confidence human PPI network, and SF model as the predominant evolutionary mechanism for the high-confidence yeast PPI network. Given the strong performance on the simulated data, we believe that our method provides a useful tool for researchers to select and develop PPI evolutionary models and may also help resolve controversy regarding topological characteristics of PPI networks from the evolutionary perspective.

Chapter 3

EVOLUTIONARY ANALYSIS AND INTERACTION PREDICTION FOR PROTEIN-PROTEIN INTERACTION NETWORK IN GEOMETRIC SPACE

3.1 Introduction

One important forefront in PPI network study is to understand how the PPI networks evolve over time. In order to reveal the underlying evolutionary mechanism of PPI network, many evolutionary models, such as Duplication-Divergence model [41, 42, 1, 71], Scale Free model [10] etc., have been proposed to simulate the evolutionary processes of PPI networks. For these different evolutionary models, there are still some controversies about fitting models to different species [47, 4, 32]. The uncertainty of evolutionary models is attributed to several factors, including the incompleteness of the current PPI networks, ineffective evaluation methods and limited applications. While recent efforts have led to development of more effective evaluation methods, such as using an Approximate Bayesian computation and Differential Evolution based method [40] that we introduced in the Chapter 2 , the evaluation is essentially restricted to using simulated data, primarily due to the lacking of "ground truth" – the ancestral networks – for real PPI network data. On the other hand, although evolutionary models are mainly used to explain how the networks evolve from an ancient version to what they currently are, by going back in time, namely "removing" edges from the current networks, it would be also useful, probably even more so, to let us "add" edges, i.e., to make prediction of de novo interactions.

Meanwhile, some attempts at de novo interaction prediction based on PPI network topology have been made, using techniques that can be loosely categorized as

"node neighborhood" [99, 16, 100]: assign a likelihood score to any candidate interaction – a pair of non-connected proteins in a PPI network – based on the topological properties of neighboring nodes. It is known that these methods perform poorly when the networks are sparse and noisy [16]. Higham et al. [33] suggest that PPI networks may have a geometric random structure. And most recently, Zhu et al. [103] further support the hypothesis of geometric random structure through recovering the PPI network by geometric embedding. Moreover, geometric embedding based methods [53, 17, 16] are demonstrated to be helpful for de-noising PPI networks. Although the embedding based methods are rooted in the notion that a PPI network lies on a low dimensional manifold shaped in a high-dimensional space whose topology is determined by the constraints imposed on the protein interactions through biological evolution, no evolutionary models are explicitly invoked.

In this chapter, we develop a novel method that combines evolutionary models and geometric embedding to make prediction of de novo protein interactions in a PPI network. The key idea of our method is to apply an evolutionary model onto a partial PPI network (with some edges reserved for testing) and embed the evolved network into a geometric space, then to predict new interactions based on Euclidean distance in that geometric space. The cross-validation using human PPI network and yeast PPI network data indicates that the invocation of appropriate evolutionary model can increase the accuracy of PPI prediction measured by ROC score by up to 14.6%, as compared to a baseline without using evolutionary information. Our approach also provides means for assessing evolutionary models on real PPI network data. The results show that our modified evolutionary model - combining a gene duplication/neofunctionalization model and scale-free model - has a better fitness and prediction efficacy for these two PPI networks.

3.2 Methods and data

3.2.1 Overview

A PPI network can be conveniently represented as a graph $G = (V, E)$ with V nodes (proteins) and E edges (interactions). G is defined by the adjacency matrix A with $V \times V$ dimension:

$$a_{i,j} = \begin{cases} 1, if(i,j) \in E \\ 0, if(i,j) \notin E \end{cases} \tag{3.1}$$

where i and j are two nodes in the nodes set V, and (i, j) represents an edge between i and j, $(i, j) \in E$. The graph is called *connected* if there is a path from any nodes to any other nodes in this graph. The real PPI networks are usually disconnected due to current incompleteness or biological features, but a maximum connected component that includes most of the nodes and edges can be found. For example, the yeast PPI dataset obtained from DIP (Release 20140117) [79] has 5120 proteins among which 5055 proteins belong to the maximum component. In order to avoid the problem of spatial overlap that is caused by embedding disconnected components, this paper only takes into consideration the maximum connected component.

The flow of our algorithm is generally illustrated in Figure 3.1. Given a PPI network, we first find the maximum connected component and its minimum spanning tree (MST) by Prim algorithm [72]. The MST is not unique given all the edges' weights are equal to 1. Then MST is used as the training sub-network. The rest of edges that are not in the MST will be used as a testing set to evaluate the performance of the predictions. Based on the training sub-network, the distances between all pair of nodes can be obtained to form a distance matrix. In typical embedding methods, the distance is the shortest path between node pairs in the minimum spanning tree, which is also called "Minimum Curvilinearity" that proposed by the work of Cannistraci et al. [17, 16]. In this work, we propose a new "distance" called *evolutionary distance*, as we hypothesize that, through evolutionary analysis we can get a more complete training sub-network based on the minimum spanning tree. After applying the evolutionary model, the evolved training sub-network is embeded into a geometric space, where

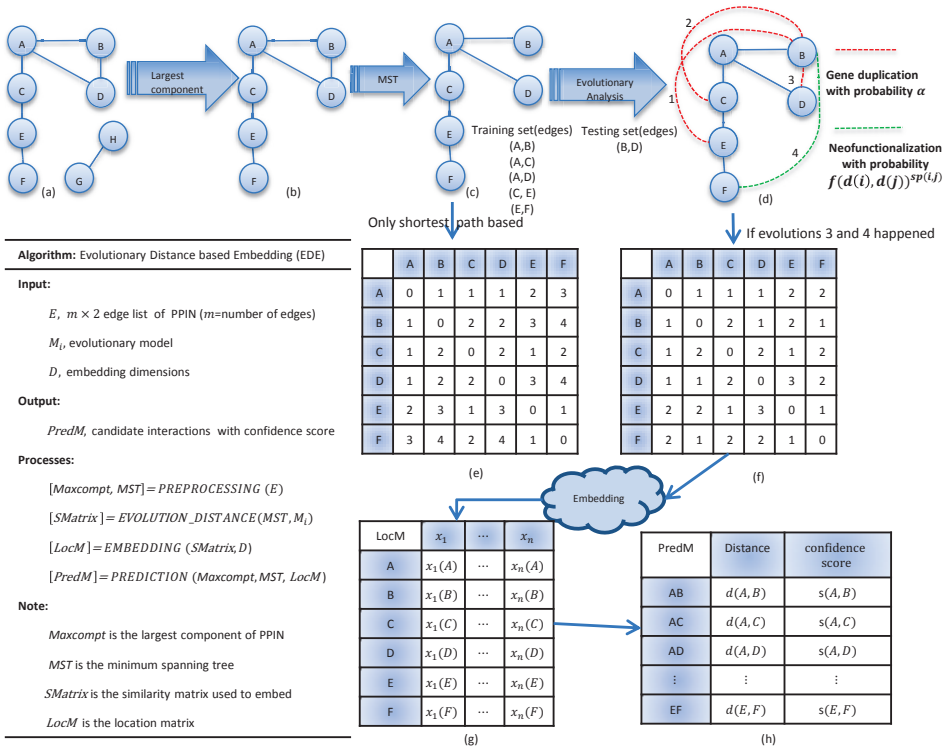


Figure 3.1: An example to show the principle of our algorithm.(a) Ground-truth network; (b) Maximum connected component; (c) Minimum spanning tree (training set); (d) Evolved network (dash lines are elementary predictions based on evolutionary analysis); (e) Distance matrix based on shortest path in MST; (f) Distance matrix based on evolutionary distance; (g) Coordinates in geometric space; (h) Euclidean distances in geometric space and corresponding confidence scores.

all of nodes are assigned with coordinates and the spatial distance between any node pairs can be computed. Confidence scores are derived from various spatial distances. Lastly, interaction prediction is made for candidate pairs based the confidences scores, and is evaluated by comparing against the testing set. More formally, given a PPI network with E edges, edges in the training sub-network can be represented by set $TrnE \in MST$, and testing edge set is $TstE = E - TrnE, TstE \cap TrnE = \emptyset$. Once we get prediction candidate set PE, the correct prediction should be $PE \cap TstE$.

3.2.2 Network Embedding Algorithm

Multi Dimensional Scaling (MDS) [93] is a classical nonlinear dimensionality reduction algorithm based on Euclidean distance. While the PPI network is represented by graph without knowing the coordinates of the nodes, so instead of using MDS directly we adopt the extension algorithm isometric feature mapping (Isomap) [88] that based on geodesic distance. However, without considering the difference between distance matrices, MDS and Isomap are equivalent to each other. Therefore, we choose MDS(Isomap) embedding technique for our method. Given a set of n nodes and a distance matrix whose elements are shortest paths between all node pairs, the basic idea of Isomap is to find coordinates in a geometric space for all the nodes such that the distance matrix derived from these coordinates approximates the original geodesic distance matrix as well as possible.

More specifically, it essentially takes the following steps to accomplish the embedding task.

a. Construct PPI interaction network(graph), and get the largest connected component.

b. Compute the shortest paths of all node pairs to get matrix $\mathbf{D}$.

c. Apply the double centering to $\mathbf{D}$ and get the symmetric, positive semi-define matrix: $\mathbf{A} = -\frac{1}{2}\mathbf{J}\mathbf{D}^2\mathbf{J}$, $\mathbf{J} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}'$, where $\mathbf{I}$ is the identity matrix that has the same size as $\mathbf{D}$; and $\mathbf{1}$ is a column vector with all one, and $\mathbf{1}'$ is the transpose of $\mathbf{1}$.

d. Extract the m largest eigenvalues $\lambda_1 ... \lambda_m$ of $\mathbf{A}$ and the corresponding m eigenvectors e_1 ... e_m, where m also means the dimensions of target geometric space.

e. Then, a m-dimensional spatial configuration of the n nodes is derived from the coordinate matrix $\mathbf{X} = \mathbf{E}_m\Lambda_m^{\frac{1}{2}}$, where $\mathbf{E}_m$ is the matrix with m eigenvectors and Λ_m is the diagonal matrix with m eigenvalues of $\mathbf{A}$.

Besides, there are many other embedding algorithms, such as Stochastic Neighbourhood Embedding (SNE) [35] and tSNE [34], Minimum Curvilinearity Embedding (MCE), non-centered MCE (ncMCE) proposed by Cannistraci et al. [17, 16] and so on. It is really hard to tell which one is better than others. Given the primary goal

of our paper is to compare the prediction efficacy of different evolutionary distance matrices and further analyze the fitnesses of different evolutionary models respect to the real PPI networks, we only include most recent MCE [17], ncMCE [16] and method proposed by Kuchaiev et al. [53] in our result comparison.

3.2.3 Evolutionary Distance based on Different Models

Instead of setting up the distance matrix based on shortest path between node pairs in the minimum spanning tree [17, 16], we propose a new evolutionary distance through analyzing the underlying evolutionary relations between node pairs. In practice, various evolutionary models may be needed for this analysis. One common mechanism by which PPI networks evolve is gene duplication; subsequently, another mechanism named post-duplication divergence may cause the PPI network to further evolve.

Briefly, the whole evolutionary process can be described as follows: given an ancestral PPIs network, any node (target node) may be duplicated at a certain probability; and then during the divergence process, some new connections may be formed between the duplication node and the target node and target node's neighbors. As the time goes on, the PPI networks evolves to its present form. However, in our application, the time arrow is reversed. That is, given a minimum spanning tree of a current PPI network, it consists of all proteins already; no new proteins would be added any more. Instead, we can infer the evolutionary relations between the existing protein pairs, for example, a protein is a duplication of its neighboring protein in the PPI work. Inspired by the most recent evolutionary model DUNE (Gene Duplication and Neofunctionalization model) [71], for any two nodes whose shortest path is 1 or 2 in the minimum spanning tree, they may be duplication for each other. In this work, we analyze the evolutionary relations and compute the evolutionary distance between node pairs based on modified DUNE, named as DANEOsf in Eq. (3.2) , where $DANEOsf\_EvoDist(i, j)$ represents the evolutionary distance between nodes (proteins) i and j, $SP(i, j)$ represents the shortest path between nodes i and j in the minimum spanning tree graph, $deg(i)$ and $deg(j)$ are the degrees for nodes i and j

respectively, $\Delta(MST)$ indicates the maximum degree of the minimum spanning tree, p_{dup} indicates the probability of interaction caused by gene duplication and divergence ($\alpha = 0.3$ in our experiments), and p_{neo} is the probability of neofunctionalization. Different with the original neofunctionalization proposed by DUNE [71], we introduce the scale-free idea into our neofunctionalization part such that, the larger the degree of a node is, the higher the probability at which the node will initiate neofunctionlization.

Fig. 3.1 shows the whole process of our method, where panels (c) and (d) are for the evolutionary analysis. Given a minimum spanning tree shown by Fig. 3.1(c), take node B for example, it can be regarded as the duplication of target nodes $\{A, C, D\}$, so there are possible connections between the duplication node B and nodes $\{A, C, D, E\}$ caused by gene duplication and divergence (red dash lines result from our evolutionary analysis). Meanwhile, another connection, as shown by green dash line, may be caused by neofunctionalization. After analyzing the evoluationary relations between all node pairs, we can get the evolutionary distance matrix as Fig. 3.1(f) shows. In other words, for any node pairs whose shortest path in MST is larger than 1, there is a predicted interaction between them if their shortest path in the evolved network becomes equal to 1.

As mentioned earlier, since it is not known a priori which evolutionary model best captures the underlying evolutionary mechanisms that a given PPI network has gone through, multiple models will be deployed, and the best one is chosen as the model that delivers the most accurate PPI prediction. In this work, we include the following models for the evolutionary analysis: Linear Preference Attachment (LPA)

[10] model Eq. (3.3) and Random Mutation model (RM) Eq. (3.4).

$$DANEOsf\_EvoDist(i,j) = \begin{cases} 1, if\ SP(i,j) = 1 \\ \\ 1,\ at\ p_{dup} = \alpha,\ or\ SP(i,j)\ at\ 1 - p_{dup};\ if\ SP(i,j) = 2,3 \\ \\ 1,\ at\ p_{neo} = (\dfrac{deg(i)deg(j)}{\Delta(MST)^2})^{SP(i,j)},\ or\ SP(i,j)\ at\ 1 - p_{neo}; \\ \quad if\ SP(i,j) > 3 \end{cases} \tag{3.2}$$

$$LPA\_EvoDist(i,j) = \begin{cases} 1, if\ SP(i,j) = 1 \\ \\ 1,\ at\ p_{lpa} = \dfrac{1}{2}\dfrac{deg(i)deg(j)}{\Delta(MST)^2},\ or\ SP(i,j)\ at\ 1 - p_{lpa}; \\ \quad if\ SP(i,j) > 1 \end{cases} \tag{3.3}$$

$$RM\_EvoDist(i,j) = \begin{cases} 1, if\ SP(i,j) = 1 \\ \\ 1,\ at\ p_{rm} = \beta,\ or\ SP(i,j)\ at\ 1 - p_{rm};\ if\ SP(i,j) > 1 \end{cases} \tag{3.4}$$

3.2.4 PPI Prediction

Each protein in the PPI network is assigned spatial coordinates from the embedding of the network (its distance matrix) into a geometric space by Isomap. Euclidean distance between all protein pairs can then be computed based on the coordinates. A threshold ϵ on the pairwise distance can be chosen empirically to determine whether there is an interaction between protein pairs. Namely, if the Euclidean distance between any two points is less than or equal to ϵ, there is a predicted interaction between their corresponding proteins in PPI networks. Otherwise, there is no interaction between them.

However, considering the large and sparse PPI network and some noises that might have inevitably "survived" despite the de-noising effect of the embedding process, we assign a confidence score to each prediction as [53] did, instead of strictly predicting node pairs into edges (interactions) and non-edges (non-interactions) only based on their Euclidean distance. As previously described, we divide edges (interactions) of a given PPI network into the training set (edges in MST) and testing set (edges not in MST). The following two probability density functions: $p(dist|edge)$ and $p(dist|nonedge)$ can be learned based on the PPI data. We use an Expectation Maximization (EM) algorithm [13] to construct these two density functions that containing maximum likelihood estimates of the parameters in a Gaussian mixture model.

$$p(dist|edge) = \sum_{i=1}^{k} \pi_{e,i} EG(dist, \mu_{e,i}, \sigma_{e,i}^2) \tag{3.5}$$

$$p(dist|nonedge) = \sum_{i=1}^{k} \pi_{ne,i} NEG(dist, \mu_{ne,i}, \sigma_{ne,i}^2) \tag{3.6}$$

The density distributions $p(dist|edge)$ and $p(dist|nonedge)$ are computed by Eq. (3.5) and Eq. (3.6) respectively. Eq. (3.5) and Eq. (3.6) are linear combinations of k Gaussian distributions, where k is the number of Gaussian components in the linear mixture model, $\mu_{e,i}$ and $\sigma_{e,i}^2$ are means and variance for edge Gaussian component i, $\mu_{ne,i}$ and $\sigma_{ne,i}^2$ are for nonedge Gaussian component i. In our experiments k was selected to be 3, since we observed there were at most 3 modes for the histograms corresponding to the densities of $p(dist|edge)$ and $p(dist|nonedge)$. Using Bayes' rule we can compute the posterior probability of edge and nonedge given a distance. The computing methods for edge and nonedge are shown by Eq. (3.7) and Eq. (3.8), where $p(edge)$ and $p(nonedge)$ represent the fraction estimation of edges and nonedges in the network.

$$p(edge|dist) = \frac{p(dist|edge)p(edge)}{p(dist)} \tag{3.7}$$

$$p(nonedge|dist) = \frac{p(nondist|edge)p(nonedge)}{p(dist)} \tag{3.8}$$

Table 3.1: PPI Network Information

Species	Nodes	Edges	Nodes(MaxComp)	Edges(MaxComp)	TrnE	TstE	CdtE
Yeast	5,119	22,637	5,055	22,285	5,054	17,231	12,768,931
Human	9,673	39,240	9,270	18,459	9,269	9,190	42,952,546

Finally, the confidence score of predicted interaction between nodes i and j can be given by Eq. (3.9). Based on that, we can draw ROC curves to compare the prediction efficacies among different evolutionary models.

$$S_{i,j} = \frac{p(edge_{i,j}|dist_{i,j})}{p(edge_{i,j}|dist_{i,j}) + p(nonedge_{i,j}|dist_{i,j})} \tag{3.9}$$

3.2.5 Data

We use yeast and human PPI networks downloaded from DIP (Release 20140117) [79] and HPRD (Release 9) [50] to test our algorithm. The detail information of these two datasets is shown in Table 3.1. Obviously, PPI networks of yeast and human are very large and sparse, especially serious for human PPI network that includes 19,599 nodes while only 39,240 known interactions. The maximum connected component of human PPI network only has 9,270 proteins, which is less than half of total. For yeast PPI network, the maximum connected component includes most of the network's nodes and edges. For these two data sets, PPI prediction here means that we need to find out the $TstE$ (true interactions) from all the candidate interactions ($CdtE$). Given the ratios of $TstE$ to $CdtE$ for yeast and human data sets are very small, it makes the prediction task very difficult, especially for human.

3.3 Results and Discussion

3.3.1 Embedding Dimension

We carry out experiments to demonstrate whether the independent variable, i.e., the dimension of the geometric space, would affect predicting efficacy. We embed the evolved training sub-network of the yeast PPI network into an n-dimensional ($n = 2, 3, 4, 5, 6, 7$) geometric space, then use the testing edge set $TstE$ to evalulate the

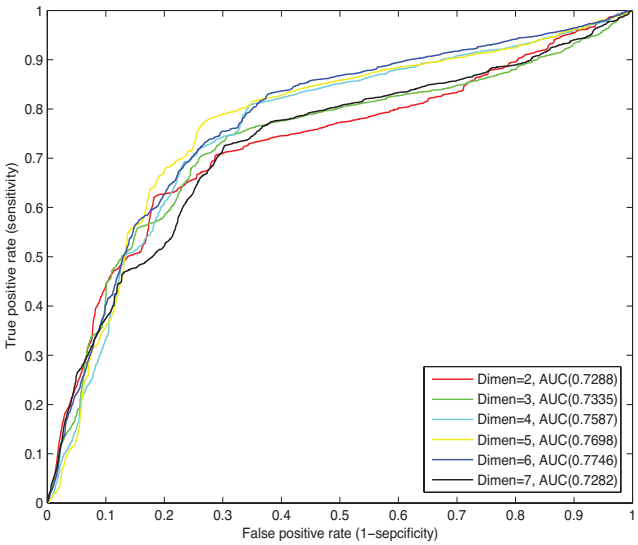


Figure 3.2: ROC curves of different embedding dimension.

prediction efficacy. As shown in Fig. 3.2, dimension can affect the performance, though somewhat slightly: the ROC score grows slowly, from 0.7288 to 0.7746, as dimension increases, and begin to decrease when dimension reaches 7. Therefore, in order to strike a balance between computational cost and prediction efficacy, dimension 5 was chosen to do the following experiments.

3.3.2 Evolutionary Analysis and Interaction Prediction for Real PPI Networks

As mentioned in the method section, our method consists of two prediction stages: elementary prediction based on evolutionary analysis and final prediction based on Euclidean distance. To make a comprehensive comparison, we apply our evolutionary distance based embedding (EDE) algorithm to the three evolutionary distance matrices (DANEOsf, LPA and RM) to compare their prediction efficacy; Moreover, we use the minimum spanning tree based shortest path matrix (SP) as an input to the minimum curvilinear embedding algorithms MCE-MDS, MCE-SVD and ncMCE-SVD (Multidimensional Scaling, Singular Value Decomposition and None-centered Singular

Value Decomposition versions) proposed by Cannistraci et al. [17, 16], and the embedding method proposed by Kuchaiev et al. [53]. As mentioned in [16], the basic idea of the embedding method adopted by Kuchaiev et al. [53] is equivalent to Isomap [88]. In other word, it is quite similar to MCE-MDS if we use SP as the input distance matrix. However, given the software of Kuchaiev et al. [53] is independently implemented, we include it in our comparison by using the software directly. As shown in Table 3.1, our experiments use 17,231 interactions as testing set and only 5,054 interactions as the training set for yeast, and 9,190 interactions as the testing set and only 9,269 as the training set for human PPI network.

To clearly show the results of different methods and input distance matrices, as shown in the Table 3.2 and Fig. 3.3∼3.7, we name each result in the format of AAA-BBB(CCC), where AAA indicates the method name, BBB indicates the embedding techniques without considering the difference of input matrices and CCC indicates the input distance matrix. For the results of yeast PPI prediction, as illustrated in Fig. 3.3, the DANEOsf based prediction EDE-MDS(DANEOsf) has the best ROC curve and highest ROC score that reaches 0.7867. The ROC score of SP based prediction for MCE-MDS, MCE-SVD ncMCE-SVD and Kuchaiev are 0.7668, 0.6851, 0.7369 and 0.6617 respectively. However, EDE-MDS(LPA) and EDE-MDS(RM) are significantly less or almost no predictive power; especially for EDE-MDS(RM), it only has a ROC score of 0.5041, and EDE-MDS(LPA)'s ROC score is 0.5859. This means that poor evolutionary models bear no useful information but a lot of noise, and consequently lead to get poor predictions. Moreover, compare with the traditional method (Kuchaiev) [53] whose ROC score is 0.6864, our DANEOsf based prediction gets a remarkable improvement with 0.1003 (14.6%) increase on ROC score.

Fig. 3.4 shows the ROC curves for the human PPI network. Because of its significantly larger size and sparsity, the performance for the human PPI network is generally not as good as that of yeast. Still, it can be seen clearly that our DANEOsf based prediction EDE-MDS(DANEOsf) gets high ROC score of 0.6979. Especially,

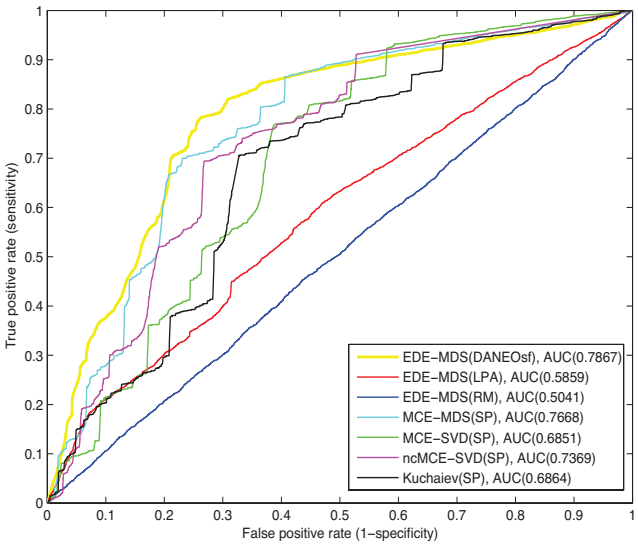


Figure 3.3: ROC curves of interaction prediction for Yeast.

when false positive rate $FPR \leq 0.2$ as shown by the dash line in Fig. 3.4, our EDE-MDS(DANEOsf) shows a clear advantage over other models.

However, in Fig. 3.4, the ROC score of ncMCE-SVD(SP) is even higher than that of our EDE-MDS(DANEOsf). Given the ROC socre of ncMCE-SVD(SP) is obvious lower than that of our EDE-MDS(DANEOsf) in Fig. 3.3 for yeast PPI network, we believe the performance of various embedding techniques may be sensitive to the difference PPI networks. So we make an extra comparison by using non-centered SVD as the embedding technique for our DANEOsf evolutionary distance matrix, we name the result as ncEDE-SVD(DANEOsf). The comparison between ncEDE-SVD(DANEOsf) and ncMCE-SVD(SP) is shown in Fig. 3.5. Obviously, the ncEDE-SVD(DANEOsf) has higher ROC score and gets an improvement with 5.4% increase on ROC score.

Overall, we have demonstrated that our method provides a useful tool at the network level to make de novo PPI prediction for both yeast and human PPI networks To compare these three methods in more detail, 15 repeated experiments of PPI prediction, involving all the steps described in the algorithm EDE in Fig. 3.1, have been conducted for each methods; due to the equal weight assigned to edges,

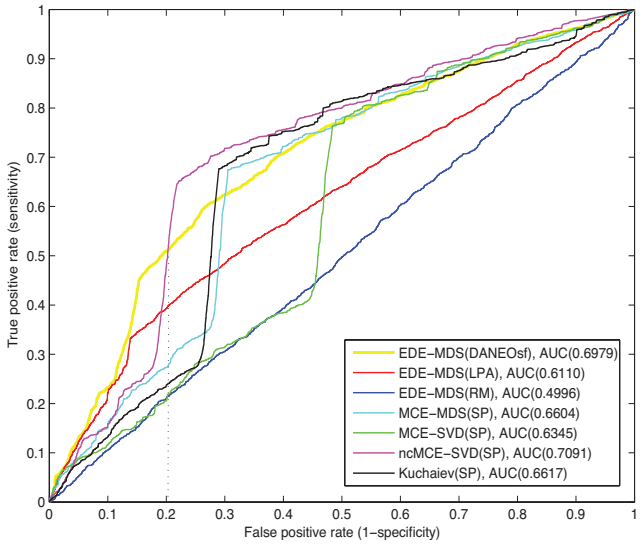


Figure 3.4: ROC curves of interaction prediction for Human.

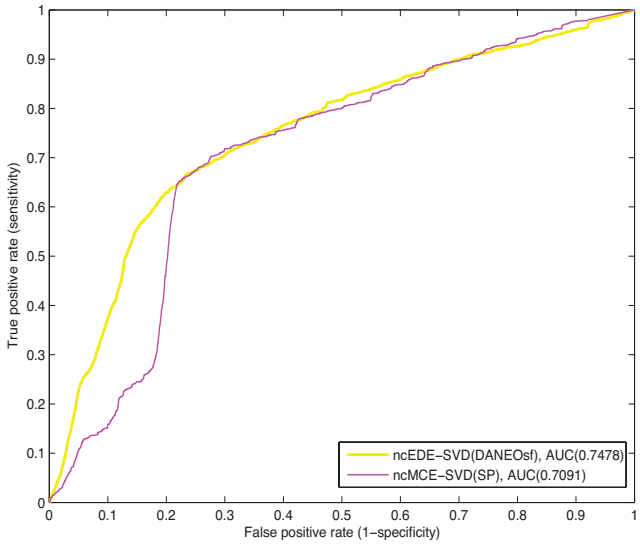


Figure 3.5: Extra comparison between ncECE-SVD(DANEOsf) and ncMCE-SVD(SP) on interaction prediction for Human.

Table 3.2: P values of paired-sample t-Test for ROC score vectors

P Values	*Yeast*		*Human*	
	MCE-MDS(SP) ($\overline{AUC}$=0.7308)	*Kuchaiev(SP)* ($\overline{AUC}$=0.6857)	*MCE-MDS(SP)* ($\overline{AUC}$=0.6430)	*Kuchaiev(SP)* ($\overline{AUC}$=0.6609)
Yeast *EDE-MDS(DANEOsf)* ($\overline{AUC}$=0.7520)	0.0237	5.1684e-09	–	–
Human *EDE-MDS(DANEOsf)* ($\overline{AUC}$=0.6835)	–	–	5.2605e-10	7.8320e-07

each experiment may get a different minimum spanning tree as the training network. In the 15 repetitions, only MCE-MDS(DANEOsf), MCE-MDS(SP) and Kuchaiev(SP) are included. For one reason, it has been demonstrated that the performance of different embedding algorithms are sensitive to different PPI networks. A fair comparison among various distance matrices should be based on same embedding techniques (MDS). For another, as shown in Fig. 3.3 and Fig. 3.4, the three ROC scores respectively obtained by EDE-MDS(DANEOsf), MCE-MDS(SP) and Kuchaiev(SP) are quite close to each other. Fig. 3.6 and Fig. 3.7 are the box plots of ROC scores of EDE-MDS(DANEOsf), MCE-MDS(SP) and Kuchaiev(SP) for yeast and human PPI prediction. It clearly shows that the EDE-MDS(DANEOsf) performs stably and better than MCE-MDS(SP) and Kuchaiev(SP). Moreover, we conducted paired-sample t-Test by making null hypothesis that the pairwise difference between ROC score vectors (EDE-MDS(DANEOsf), MCE-MDS(SP) and Kuchaiev(SP)) has a mean equal to zero, the result of t-Test reject the null hypothesis at 5% significance level. And the P values are shown by Table 3.2.

3.3.3 Discussion

Note that, given the incompleteness of the current PPI networks, the evaluation of prediction may be biased, because we do not know which missing links are truly negative and which of them are undetected links or links that will appear in the future. It is likely that the actual prediction performance may be even better than what is shown here, as what is considered as false positives (i.e., predicted edges are not present in the

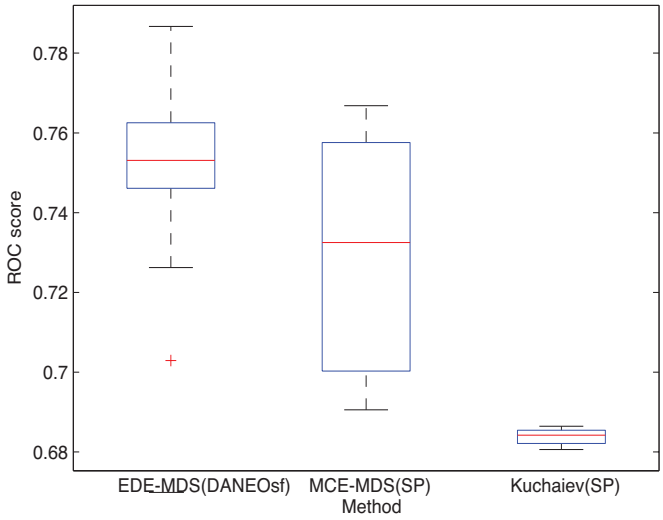


Figure 3.6: Box plot of ROC scores for Yeast PPI prediction based on EDE-MDS(DANEOsf), MCE-MDS(SP) [16] and Kuchaiev [53].

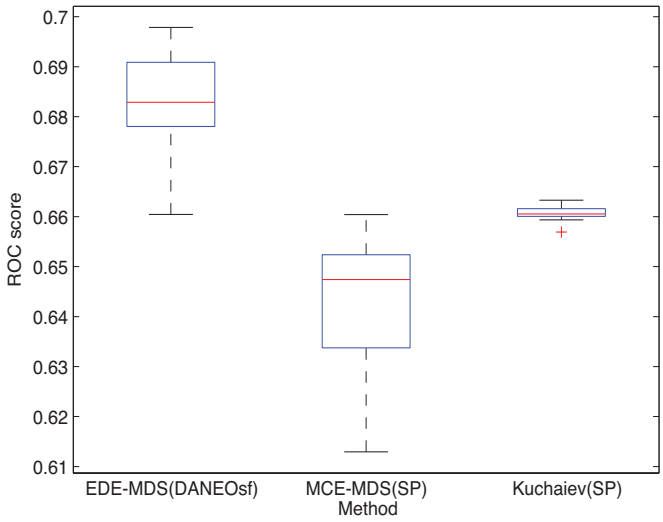


Figure 3.7: Box plot of ROC scores for Human PPI prediction based on EDE-MDS(DANEOsf), MCE-MDS(SP) [16] and Kuchaiev [53].

testing set) might turn out to be true positive when the predicted interactions are later validated by new experiments. For example, as we further validated the false positives (threshold for confidence score is 0.8) of HPRD by using BioGRID(version 3.2.110) [84] which is newer, the result shows 5 of false positives appeared in the BioGRID(version 3.2.110) and none of the 5 appeared in HPRD before. However, what is uncertain is how such changes will impact on the performance of various methods; depending on the data, it is possible that the current top performer may be outperformed by other methods.

Currently, there is still no consensus yet with respect to topological characteristics of PPI networks; while some reported scale-free [4], others did not [31], and various duplication-divergence based evolutionary models [41, 42, 1, 71] have been proposed. Of particular importance is that our method offers an effective way to select evolutionary models that best capture the underlying evolutionary mechanisms, evaluating the fitness of evolutionary models from the perspective of PPI prediction on real PPI networks.

3.4 Conclusions

In this chapter, we developed a novel evolutionary analysis and PPI prediction method that can make de novo PPI prediction using information at the network level. We demonstrated that incorporating evolutionary information into PPI networks can help, in some cases very significantly, improve the prediction performance as compared with using just geometric embedding. In addition, our method offers an effective way to select among multiple candidate evolutionary models the one that best captures the underlying evolutionary mechanisms, as measured by the PPI prediction on real PPI networks instead of simulated networks. The improved PPI prediction performance may suggest that our DANEOsf evolutionary model can uncover the underlying evolutionary mechanism for these two PPI networks better than other tested models.

Chapter 4

INFERENCE OF PROTEIN-PROTEIN INTERACTION NETWORKS FROM MULTIPLE HETEROGENEOUS DATA

4.1 Introduction

Reconstruction of PPI networks is a central task in systems biology, and inference from multiple heterogeneous data sources offers a promising computational approach to making de novo PPI prediction by leveraging complementary information and the partial network structure. In this chapter, we propose a new method based on ABC-DEP sampling method proposed in Chapter 2 and Regularized Laplacian kernel (RL) to infer PPI networks from multiple hetergeneous data. The method uses both topological features and various genomic kernels, which are weighted to form a kernel fusion. The weights are optimized using ABC-DEP sampling [40]. Unlike data fusion with genomic kernels for binary classification [56], the combined kernel in our case will be used instead to create a regularized Laplacian kernel [28, 43] for PPI prediction. We demonstrate how the method circumvents the issue of unbalanced data faced by many machine learning methods in bioinformatics. One main advantage of our method is that only a small partial network is needed for training in order to make the inference at the whole network level. Moreover, the results show that our method works particularly well with detecting interactions between nodes that are far apart in the network, which has been a difficult task for other methods. Tested on Yeast PPI data and compared to two control methods, traditional regularized Laplacian kernel method and regularized Laplacian kernel based on equally weighted kernels, our method shows a significant improvement of over 20% increase in performance measured by ROC score.

4.2 Methods and Data

4.2.1 Problem Definition

Formally, a PPI network can be represented as a graph $G = (V, E)$ with V nodes (proteins) and E edges (interactions). G is defined by the adjacency matrix A with $V \times V$ dimension:

$$A_{i,j} = \begin{cases} 1, if(i,j) \in E \\ 0, if(i,j) \notin E \end{cases} \tag{4.1}$$

where i and j are two nodes in the nodes set V, and (i, j) represents an edge between i and j, $(i, j) \in E$. The graph is called *connected* if there is a path of edges to connect any two nodes in the graph. For supervised learning, we randomly divide the network into three parts: connected training network $G_{tn} = (V, E_{tn})$, validation set $G_{vn} = (V_{vn}, E_{vn})$ and testing set $G_{tt} = (V_{tt}, E_{tt})$, such that $E = E_{tn} \cup E_{vn} \cup E_{tt}$, and any edge in G can only belong to one of these three parts.

A kernel is a symmetric positive definite matrix K, whose elements are defined as a real-valued function $K(x, y)$ satisfying $K(x, y) = K(y, x)$ for any two proteins x and y in the data set. Intuitively, the kernel for a given dataset can be regarded as a measure of similarity between protein pairs with respect to the biological properties, from which kernel function takes its value. Treated as an adjacency matrix, a kernel can also be thought of as a complete network in which all the proteins are connected by weighted edges. Kernel fusion is a way to integrate multiple kernels from different data sources by a linear combination. For our task, this combination is made of the connected training network and various feature kernels $K_i, i = 1, 2, 3...n$ by optimized weights $W_i, i = 0, 1, 2, 3...n$, which formally is defined by Eq. (4.2)

$$K_{fusion} = W_0 G_{tn} + \sum_{i=1}^{n} W_i K_i \tag{4.2}$$

Note that the training network is incomplete, i.e., with many edges taken away and reserved as testing examples. Therefore, our inferring task is to predict or recover the interactions in the testing set G_{tt} based on the kernel fusion.

4.2.2 How to Infer PPI Network?

Once the kernel fusion is obtained, it will be used to make PPI inference, in the spirit of random walk. However, instead of directly doing random walk, we apply Reguarized Laplacian (RL) kernel to the kernel fusion, which allows for PPI inference at the whole network level. The Regularized Laplacian kernel [43, 81] is also called the normalized random walk with restart kernel in Mantrach et al. [60] because of the underlying relations to the random walk with restart model [65, 90]. Formally, it is defined as Eq. (4.3)

$$RL = \sum_{k=0}^{\infty} \alpha^k (-L)^k = (I + \alpha * L)^{-1} \tag{4.3}$$

where $L = D - A$ is the Laplacian matrix made of the adjacency matrix A and

Algorithm 3 *PPI Inference*

Input: $RL \leftarrow$ *Regularized Laplacian prediction kernel*
$G_{tn} \leftarrow$ *training network*
$G_{vn} \leftarrow$ *validation set*
$G_{tt} \leftarrow$ *testing set*
$K \leftarrow$ *feature kernels*

Output: Inferred network

1: $W^{opt} \leftarrow$ *ABC-DEP*(G_{tn}, G_{vn}, RL, K)
2: *OPT-K*$\leftarrow W_0^{opt} G_{tn} + \sum_{i=1}^{n} W_i^{opt} K_i$ // *OPT-K* is the optimal kernel fusion based on optimal weights
3: $RL_{OPT\text{-}K} \leftarrow RL(OPT\text{-}K)$ // Apply RL model to the kernel fusion
4: Rank $RL_{OPT\text{-}K}$ and infer G_{tt}

the degree matrix D; and $0 < \alpha < \rho(L)^1$ where $\rho(L)$ is the spectral radius of L. Here, we use kernel fusion in place of the adjacent matrix, so that various feature kernels in Eq. (4.2) are incorporated in influencing the random walk with restart on the weighted networks [6]. With the regularized Laplacian matrix, no random walk is actually needed to measure how "close" two nodes are and then use that closeness to infer if the two corresponding proteins interact. Rather, RL_K is the inferred matrix, and is interpreted as a probability matrix P in which $P_{i,j}$ indicates the probability of an interaction for protein i and j. Algorithm 3 shows the general steps to infer PPI

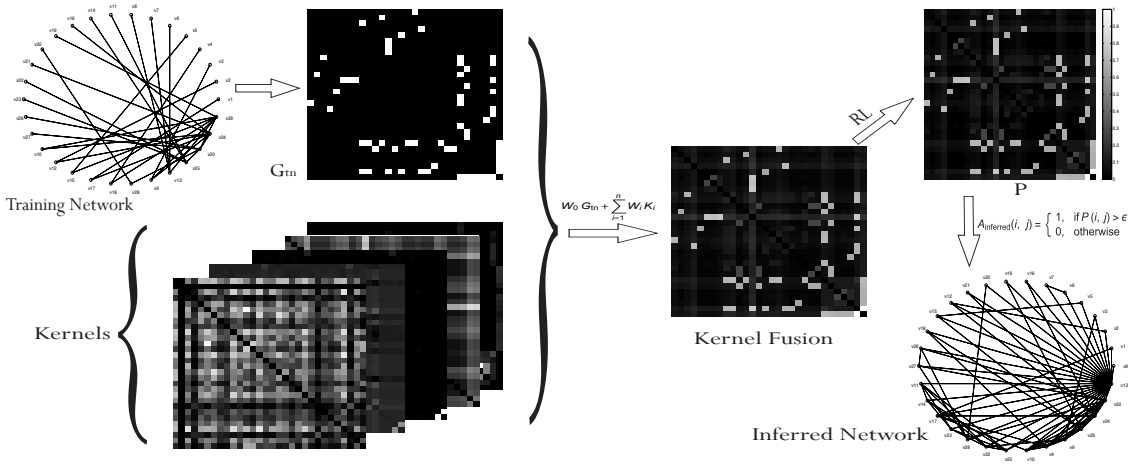


Figure 4.1: An example to show the inference process. The example comprises of a small module in the DIP yeast PPI network, which consists of protein P25358 (ELO2, Elongation of fatty acids protein 2) and its $1 \sim 3$ hops away neighbors. The kernel fusion and the regularized Laplacian are shown as heatmap. The lighter a cell is, the more likely the corresponding proteins interact.

network from a optimal kernel fusion. Fig. 4.1 contains a toy example to show the process of inference, where both the kernel fusion and the regularized Laplacian are shown as heatmap. The lighter a cell is, the more likely the corresponding proteins. However, to ensure good inference, it is important to learn optimal weights for G_{tn} and various K_i to build kernel fusion K_{fusion}. Otherwise, given the multiple heterogeneous kernels from different data sources, the kernel fusion without optimized weights is likely to generate erroneous inference on PPI.

4.2.3 ABC-DEP Sampling Method for Learning Weights

In this work we revise the ABC-DEP sampling method [40] to optimize the weights for kernels in Eq. (4.2). ABC-DEP sampling method, based on approximate Bayesian computation with differential evolution and propagation, shows strong capability of accurately estimating parameters for multiple models at one time. The parameter optimization task here is relatively easier than that in [40] as there is only one RL based prediction model. Specifically, given the connected training network G_{tn} and N feature kernels in Eq. (4.2), the length of the particle in ABC-DEP would be

Algorithm 4 *ABC-DEP*

Input: G_{tn}, G_{vn}, RL, K
$M \leftarrow iteration\ times$
$N_p \leftarrow particles$
Output: W^{opt}

while $t \leq M$ **do**
  if $t = 1$ **then**
    Initialize N_p particles, each particle contains weights $W_i, 0 < W_i < 1, i = 0, 2, 3...n$ for training network and n-1 feature kernels
    $P_t, I_t \leftarrow \{P^i, I^i\}_{i=1}^{N_p}$ // P^i is a particle, I^i is the weight or importance of P^i. P_t, I_t represents the t^{th} generation of particles and weights.
  else
    $\{P_t, I_t\}_{i=1}^{N_p} \leftarrow Sampling((P_{t-1}, I_{t-1}))$
  end if
  $(P_{t+1}, I_{t+1}) \leftarrow DEP(P_t, I_t, G_{tn}, G_{vn}, RL, K)$
  $t \leftarrow t + 1$
end while
$Normalize(P, I)$
$W^{opt} \leftarrow P^i\ if\ I^i = max(I)$

$N + 1$, where particle can also be seen as a sample including the $N + 1$ weight values. As mentioned before, the PPI network is divided into three parts: the connected training network G_{tn}, validation set G_{vn} and testing set G_{tt}. To obtain the optimal particle(s), a population of particles with size N_p is intialized, and ABC-DEP sampling is run iteratively until a particle is found in the evolving population that maximizes the AUROC of inferring training network G_{tn}, validation set G_{vn}. The validation set G_{vn} is used to avoid over-fitting as the algorithm converges. Algorithm 4 shows the detailed sampling process.

Algorithm 4 is the main structure in which a population of particles with random weights is initialized. Given the particle population, Algorithm 5 samples through the parameter space for good particles and assigns them weights according to the predicting quality of their corresponding kernel fusion K_{fusion}. We accept or reject a new candidate particle based on Similuted Annealing method [52]. Through the evolution process, bad particles will be filtered out and good particles will be kept for the next generation. We repeat this process until the algorithm converges. The

Algorithm 5 DEP

Input: $G_{tn}, G_{vn}, RL, K, N_p$
Output: P, I

for $i = 1$ to N_p **do**
  Randomly select P^f, P^j, P^k where $i \neq j \neq k \neq f$
  // P^i is the target particle, P^f, P^j and P^k are three randomly selected particles. $P^i.\theta$, $P^j.\theta$, $P^k.\theta$ and $P^f.\theta$ represent particles' parameter vectors that consist of weights for feature kernels.
  if $P^i.\theta = P^j.\theta = P^k.\theta = P^f.\theta$ **then**
    $Z^i \leftarrow Propagation(P^i)$
  else
    $Z^i \leftarrow DifferentialEvolution(P^i, P^j, P^k, P^f)$
  end if
end for
for $i = 1$ to N_p **do**
  $r'_{G_{tn}}, r'_{G_{vn}} = Inference(RL, Z^i, K, G_{tn}, G_{vn})$
  $r' = r'_{G_{tn}} + r'_{G_{vn}}$ // In the $Inference$ function, particle Z^i is used to weight kernels in K to get kernel fusion K_{fusion}. $r'_{G_{tn}}, r'_{G_{vn}}$ represent results (AUROC) of recovering G_{tn} and G_{vn} based on K_{fusion} respectively
  $r_{G_{tn}}, r_{G_{vn}} \leftarrow Inference(RL, P^i, K, G_{tn}, G_{vn})$.
  $r \leftarrow r_{G_{tn}} + r_{G_{vn}}$
  if $rand(0,1) < e^{\frac{r'-r}{T(t)}}$ **then**
    $P^i \leftarrow Z^i, I^i \leftarrow I^i * \frac{\beta}{\alpha - r'}$
  else
    $P^i \leftarrow P^i, I^i \leftarrow I^i * \frac{\beta}{\alpha - r}$
  end if
end for
$Normalize(P, I)$

optimal particle is used to build kernel fusion K_{fusion} for PPI prediction.

4.2.4 Data and Kernels

We use yeast PPI networks downloaded from DIP database (Release 20150101) [79] to test our algorithm. Notably, some interactions without Uniprotkb ID have been filtered out in order to do name mapping and make use of genomic similarity kernels [56]. As a result, the PPI network contains 5,093 proteins and 22,423 interactions, from which the largest connected component is used to serve as golden standard network. It consists of 5,030 proteins and 22,394 interactions. Only tens of proteins and interactions are not included in the largest connected component, which makes the golden standard data almost as complete as the original network. As mentioned before, the golden standard PPI network is divided into three parts that are connected training network G_{tn}, validation set G_{vn} and testing set G_{tt}, where validation set G_{vn} is used to find optimal weights for feature kernels and testing set G_{tt} is used to evaluate the inference capability of our method.

Six feature kernels are included in this study and the following list is about the detailed information of these kernels.

G_{tn}: G_{tn} is the connected training network that provides connectivity information. It can also be thought of as a base network to do the inference.

$K_{Jaccard}$ [45]: It measures the similarity of protein pairs i, j in term of $\frac{neigbors(i) \cap neighbors(j)}{neighbors(i) \cup neighbors(j)}$.

K_{SN}: It measures the total number of neighbors of protein i and j, $K_{SN} = neighbors(i) + neighbors(j)$.

K_B [56]: It is a sequence-based kernel matrix that is generated using the BLAST [5].

K_E [56]: This is a gene co-expression kernel matrix constructed entirely from microarray gene expression measurements.

K_{Pfam} [56]: This is a generalization of the previous pairwise comparison-based matrices in which the pairwise comparison scores are replaced by expectation values derived from hidden Markov models (HMMs) in the Pfam database [83].

All these kernels are normalized to the scale of $[0, 1]$ in order to avoid bias.

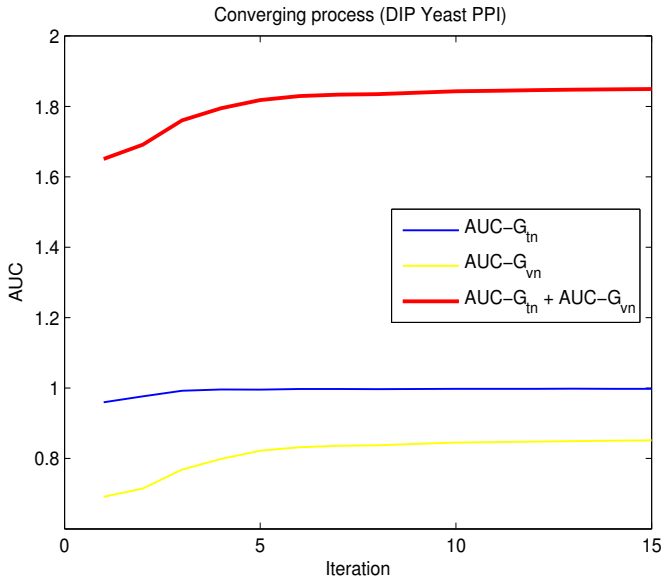


Figure 4.2: The converging process of ABC-DEP samping that used to obtain optimal weights.

4.3 Results and Discussion

4.3.1 Inferring PPI network

To show how well our method can infer PPI network from the kernel fusion, we make the task challenging by randomly dividing the golden standard yeast PPI network into following three parts: the connected training network G_{tn} has 5,030 nodes and 5,394 edges, the validation set G_{vn} has 1,000 edges, and the testing set G_{tt} has 16,000 edges. This means that we need to infer and recover a large number of testing edges based on the kernel fusion and a small validation set. Firstly, we check the converging process of finding the optimal weights that used to combine feature kernels, which is shown by the Fig. 4.2. It clearly shows that when the AUC of predicting the training network G_{tn} reaches to 1 quickly, but the AUC of predicting the validation set G_{vn} is still in a upward trend. So G_{tn} alone cannot guarantee the optimality of the weights when the algorithm converges, which is the reason the validation set G_{vn} is used. After several iterations, the ABC-DEP algorithm is converged when both AUCs have become steady.

With the optimal weights obtained from ABC-DEP sampling, we build the kernel fusion K_{fusion} by Eq. (4.2). PPI network inference is made with RL kernel(4.3). The performance of inference is evaluated by how well the testing set G_{tt} is recovered. Specifically, all node pairs are ranked in decreasing order by their edge weights in the RL matrix, and edges in the testing set G_{tt} are the labeled as positive and node pairs with no edges in G are labeled as negative. A ROC curve is plotted for true positive v.s. false positives, by running down the ranked list of node pairs. Fig. 4.3 shows the ROC curves and AUCs for three PPI network inferences: $RL_{OPT\text{-}K}$, $RL_{G_{tn}}$ and $RL_{EW\text{-}K}$, where $RL_{OPT\text{-}K}$ indicates the RL based PPI inference is from kernel fusion that built by optimal weights, $RL_{G_{tn}}$ indicates RL based PPI inference is solely from the training network G_{tn}, and $RL_{EW\text{-}K}$ represents RL based PPI inference is from kernel fusion built by equal weights, e.g. $w_i = 1, i = 0, 1...n$. Additionally, $G_{set} \sim n$ indicates there is n number of edges in the set G_{set}, e.g. $G_{tn} \sim 5394$ means the connected training network G_{tn} contains 5,394 edges. As shown by Fig. 4.3, the PPI reference $RL_{OPT\text{-}K}$ based on our method significantly outperforms the other two control methods, with a 20% increase over $RL_{G_{tn}}$ and a 23.6% over $RL_{EW\text{-}K}$ in term of AUC. It is noted that the AUC of PPI inference $RL_{EW\text{-}K}$ based on the equally weighted built kernel fusion is even worse than that of $RL_{G_{tn}}$ based on a really small training network. It means there should be a lot of noises if we just naively combine different feature kernels to do PPI prediction. Our method provides an effective way to make good uses of various features for improving PPI prediction performance.

4.3.2 Effects of the Training Data

Usually, given a golden standard data, we need to retrain the prediction model for different division of training set and testing set. However, if optimal weights have been found for building kernel fusion, our PPI network inference method enable us to train the model once, and do prediction or inference for different testing sets. To demonstrate that, we keep the two PPI inferences $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$ obtained before (in last section) unchanged, and evaluate the prediction ability for different

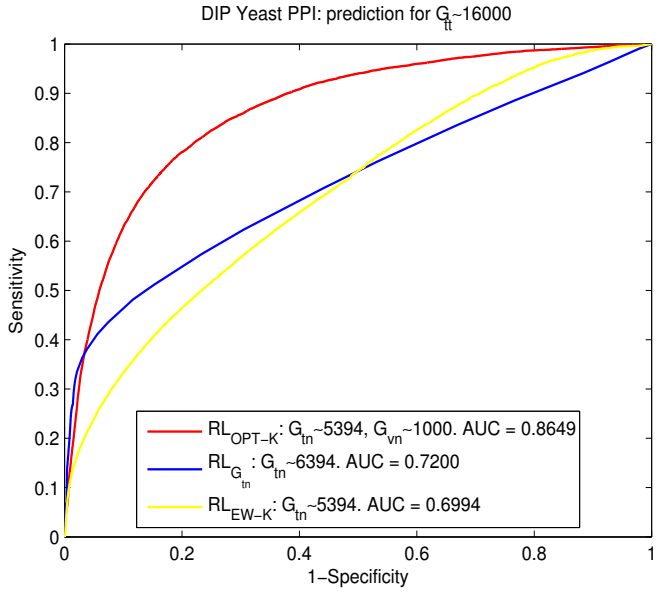


Figure 4.3: ROC curves of predicting $G_{tt} \sim 16000$ by $RL_{G_{tn}\sim 5394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$.

testing sets. We also examine how performance is affected by sizes of various sets. Specifically, while the size of training network G_{tn} for $RL_{G_{tn}}$ increases, sizes of $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$ are kept unchanged. Therefore, we design several experiments by dividing the golden standard network into G^i_{tn} and G^i_{tt}, $i = 1, ..., n$, and building PPI inference $RL_{G^i_{tn}}$ to predict G^i_{tt} for every time. To make comparison, we also use $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$ to predict G^i_{tn}. Fig. 4.4 shows the ROC curves of predicting $G_{tt} \sim 15000$ by $RL_{G_{tn}\sim 7394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$. Fig. 4.5, Fig. 4.6 and Fig. 4.7 show similar results but just for different G_{tn} and G_{tt} sets. As shown by the Fig. 4.4, Fig. 4.5, Fig. 4.6 and Fig. 4.7, $RL_{OPT\text{-}K}$ trained on only 5,394 golden standard edges still performs better than the control methods that employ significantly more golden standard edges.

4.3.3 Detection of Interacting Pairs Far Apart in the Network

It is known that the basic idea of using random walk or random walk based kernels [90, 58, 6, 28] for PPI prediction is that good interacting candidates usually are not faraway from the start node, e.g. only 2,3 edges away in the network. Consequently, the testing nodes have been chosen to be within a certain distance range,

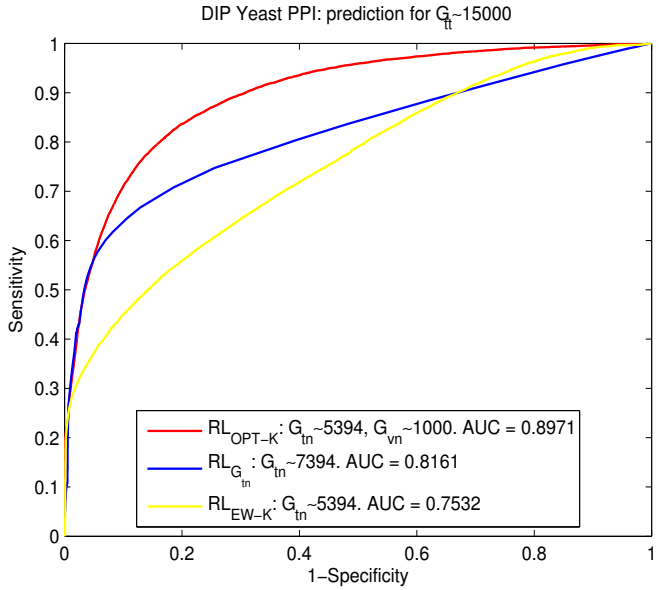


Figure 4.4: ROC curves of predicting $G_{tt} \sim 15000$ by $RL_{G_{tn}\sim 7394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$.

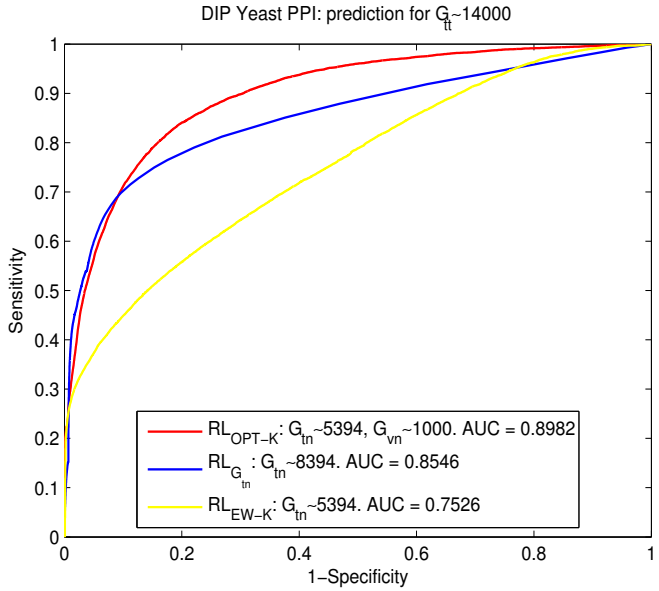


Figure 4.5: ROC curves of predicting $G_{tt} \sim 14000$ by $RL_{G_{tn}\sim 8394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$.

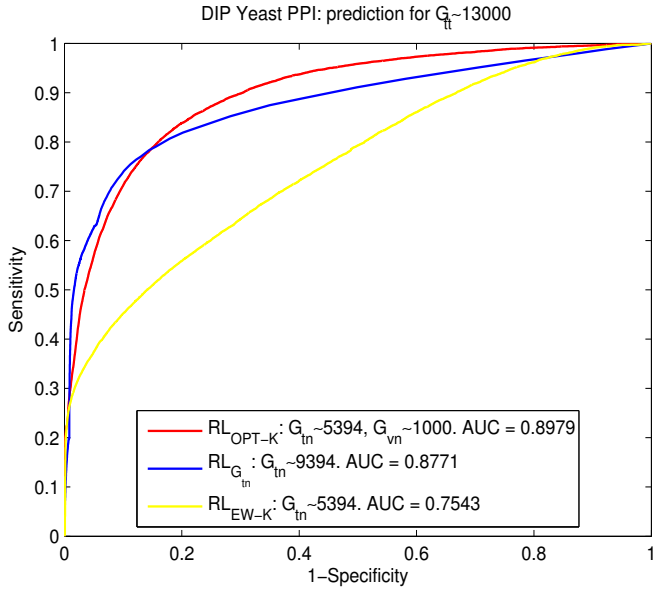


Figure 4.6: ROC curves of predicting $G_{tt} \sim 13000$ by $RL_{G_{tn}\sim 9394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$.

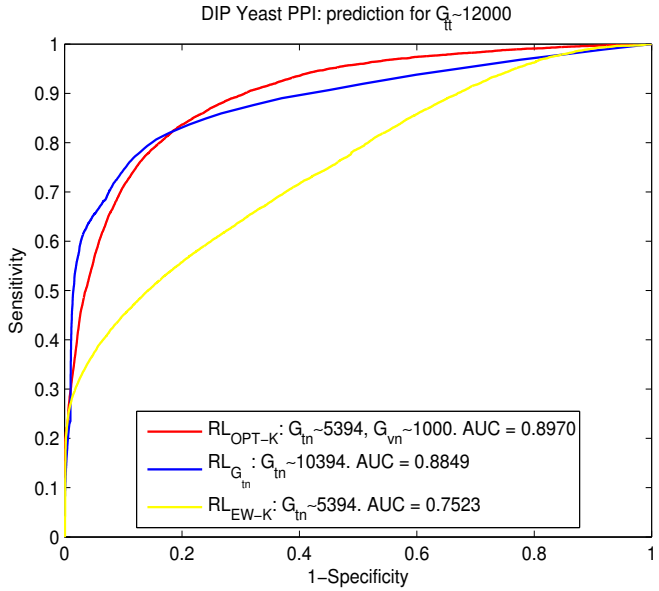


Figure 4.7: ROC curves of predicting $G_{tt} \sim 12000$ by $RL_{G_{tn}\sim 10394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$.

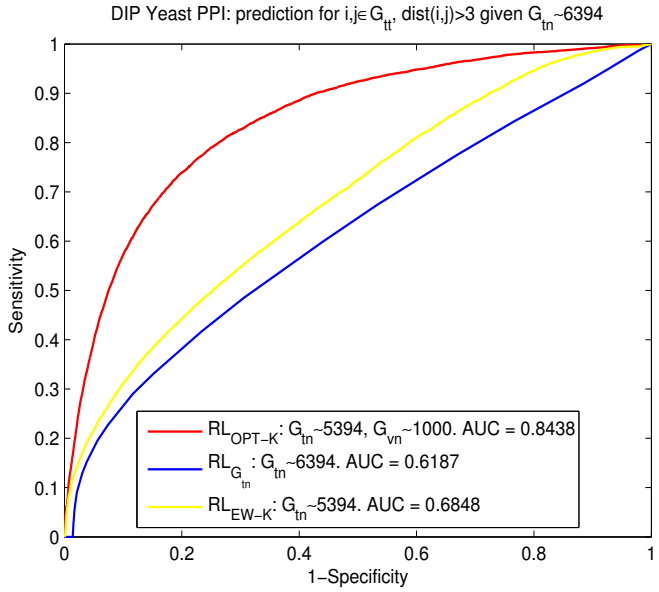


Figure 4.8: ROC curves of predicting $G_{tt}^{(dist(i,j)>3)}$ by $RL_{G_{tn}\sim 6394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$.

which largely contributes to the good performance reported by many network-level link prediction methods. In reality, however, a method that is capable and good at detecting interacting pairs far apart in the network can be even more useful, such as in uncovering cross talk between pathways that are not nearby in the PPI network.

To investigate how our proposed method performs at detecting faraway interactions, we still use $RL_{G_{tn}\sim 6394}$, $RL_{OPT\text{-}K}$ and $RL_{EW\text{-}K}$ for inferring PPIs, but we select node pairs (i, j) that satisfy $dist(i, j) > 3$ *given* $G_{tn} \sim 6394$ from G_{tt} as new testing set and name it $G_{tt}^{(dist(i,j)>3)}$. Fig. 4.8 shows that $RL_{OPT\text{-}K}$ has not only a significant margin over the control methods in detecting long-distance PPIs but also maintains a high ROC score of 0.8438 comparable to that of all PPIs. In contrast, $RL_{G_{tn}\sim 6394}$ performs poorly and worse than $RL_{EW\text{-}K}$, which means the traditional RL kernel based on adjacent training network alone cannot detect faraway interactions well.

4.3.4 Analysis of Weights and Efficiency

As the method incorporates multiple heterogeneous data, it can be insightful to inspect the final optimal weights. In our case, the optimal weights are 0.8608,

0.1769, 0.9334, 0, 0.0311, 0.9837, respectively for feature kernels G_{tn}, $K_{Jaccard}$, K_{SN}, K_B, K_E and K_{Pfam}. These weights indicate that K_{SN} and K_{Pfam} are the predominant contributors to PPI prediction. This observation is consistent with the intuition that proteins interact via interfaces made of conserved domains [23], and PPI interactions can be classified based on their domain families and domains from the same family tend to interact [44, 66, 12]. Although the true strength of our method lies in integrating multiple heterogeneous data for PPI network inference, the optimal weights can serve as a guidance to select most relevant features when time and resources are limited.

Lastly, despite of the common concern of time efficiency with methods based on evolutionary computing, the issue is mitigated in our case. In our experiments, only a small number of particles, 150 to be exact, is needed for the initial population for ABC-DEP sampling. Also, as shown in the Fig. 4.2 our ABC-DEP algorithm is quickly converged, within 10 iterations. Moreover, since the PPI inference from $RL_{OPT\text{-}K}$ is shown to be less sensitive to the size of training data, only 5,394 gold standard edges, less than 25% of the total number, are used. And we do not need to retrain the model for different testing data, which is another time saving property of our method.

4.4 Conclusions

In this chapter we developed a novel supervised method that enables inference of PPI networks from topological and genomic feature kernels in an optimized integrative way. Tested on DIP yeast PPI network, the results show that our method exhibits competitive advantages over control methods in several ways. First, the proposed method achieved superior performance in PPI prediction, as measured by ROC score, over 20% higher than the baseline, and this margin is maintained even when the control methods use a significantly larger training set. Second, we also demonstrated that by integrating topological and genomic features into regularized Laplacian kernel, the method avoids the short-range problem encountered by random-walk based methods – namely the inference becomes less reliable for nodes that are far from the start node of the random walk, and show obvious improvements on predicting faraway interactions.

Lastly, our method can also provide insights into the relations between PPIs and various similarity features of protein pairs, thereby helping us make good use of these features. As more features with respect to proteins are collected from various -omics studies, they can be used to characterize protein pairs in terms of feature kernels from different perspectives. Thus we believe that our method provides a useful framework in fusing various feature kernels from heterogeneous data to improve PPI prediction.

Chapter 5

PROTEIN-PROTEIN INTERACTION NETWORK INFERENCE FROM MULTIPLE KERNELS WITH OPTIMIZATION BASED ON RANDOM WALK BY LINEAR PROGRAMMING

5.1 Introduction

In Chapter 4 we proposed a new method based on ABC-DEP sampling method proposed in Chapter 2 and Regularized Laplacian kernel (RL) to infer PPI networks from multiple hetergeneous data. Although the optimal weights obtained by ABC-DEP for multiple hetergenesou data can help us remarkably improve the prediction performance, how to quickly learn weights remains a challenge. In this chapter, we develop a method to infer de novo PPIs by combining multiple data sources represented in kernel format and obtaining optimal weights based on random walk over the existing partial network. Our proposed method utilizes Baker algorithm and the training data to construct a transition matrix which constrains how a random walk would traverse the partial network. Multiple heterogeneous features for the proteins in the network, including gene expression and Pfam domain profiles, are then combined into the form of a weighted kernel, which provides a new "adjacency matrix" for the whole network but is required to comply with the transition matrix on the part of the training subnetwork. This requirement is met by adjusting the weights to minimize the element-wise difference between the transition matrix and the weighted kernel. The minimization problem is solved by linear programming. The weighted kernel is then transformed to Regularized Laplacian(RL) kernel to infer missing or new edges in the PPI network. The results on synthetic data and real data from Yeast show that the accuracy of PPI prediction measured by AUC is increased by up to 19% as compared to a control method without using optimal weights. Moreover, the weights learned by our method

Weight Optimization by Linear Programming (WOLP) are very consistent with that learned by sampling, and can provide insights into the relations between PPIs and various feature kernels, thereby improving PPI prediction.

5.2 Methods

5.2.1 Problem Definition

Formally, a PPI network can be represented as a graph $G = (V, E)$ with V nodes (proteins) and E edges (interactions). G is defined by the adjacency matrix A with $V \times V$ dimension:

$$A(i,j) = \begin{cases} 1, if(i,j) \in E \\ 0, if(i,j) \notin E \end{cases} \tag{5.1}$$

where i and j are two nodes in the nodes set V, and (i, j) represents an edge between i and j, $(i, j) \in E$. The graph is called *connected* if there is a path of edges to connect any two nodes in the graph. Given many PPI networks are not connected and has many connected component with various size, we select a large connected component (e.g. largest connected component) as golden standard network to do supervised learning, we divide the golden standard network into three parts: connected training network $G_{tn} = (V, E_{tn})$, validation set $G_{vn} = (V_{vn}, E_{vn})$ and testing set $G_{tt} = (V_{tt}, E_{tt})$, such that $E = E_{tn} \cup E_{vn} \cup E_{tt}$, and any edge in G can only belong to one of these three parts.

A kernel is a symmetric positive definite matrix K, whose elements are defined as a real-valued function $K(u, v)$ satisfying $K(u, v) = K(u, v)$ for any two proteins u and v in the data set. Intuitively, the kernel built from a given dataset can be regarded as a measure of similarity between protein pairs with respect to the biological properties, from which kernel function takes its value. Treated as an adjacency matrix, a kernel can also be thought of as a complete network in which all the proteins are connected by weighted edges. Kernel fusion is a way to integrate multiple kernels from different data sources by a linear combination. For our task, this combination is made

of the connected training network and various feature kernels $K_i,\ i = 1, 2, 3...n$, which formally is defined by Eq. (5.2)

$$K_{fusion} = W_0 G_{tn} + \sum_{i=1}^{n} W_i K_i,\ \ where\ K_i(u, v) = \frac{K_i(u, v)}{\sum\limits_{w} K_i(u, w)} \tag{5.2}$$

Note that the training network is incomplete, i.e., with many edges taken away and reserved as testing examples. Therefore, the task is to infer or recover the interactions in the testing set G_{tt} based on the kernel fusion. Once the kernel fusion is obtained, it will be used to make PPI inference, in the spirit of random walk. However, instead of directly doing random walk, we apply regularized Laplacian (RL) kernel to the kernel fusion, which allows for PPI inference on the whole network level. The regularized Laplacian kernel [43, 81] is also called the normalized random walk with restart kernel in Mantrach et al. [60] because of the underlying relations to the random walk with restart model [65, 90]. Formally, it is defined as Eq. (5.3), where $L = D - A$ is the Laplacian matrix made of the adjacency matrix A and the degree matrix D, and $0 < \alpha < \rho(L)^{-1}$ and $\rho(L)$ is the spectral radius of L. Here, we use kernel fusion in place of the adjacency matrix, generating a regularized Laplacian matrix RL_K, so that various feature kernels in Eq. (5.2) are incorporated in influencing the random walk with restart on the weighted networks [6]. With the regularized Laplacian matrix, no random walk is actually needed to measure how "close" two nodes are and then use that closeness to infer if the two corresponding proteins interact. Rather, RL_K is interpreted as a probability matrix P in which $P_{i,j}$ indicates the probability of an interaction for protein i and j.

$$RL = \sum_{k=0}^{\infty} \alpha^k (-L)^k = (I + \alpha * L)^{-1} \tag{5.3}$$

To ensure good inference, it is important to learn optimal weights for G_{tn} and various K_i to build kernel fusion K_{fusion}. Otherwise, given the multiple heterogeneous kernels from different data sources, the kernel fusion without optimized weights is likely to generate erroneous inference on PPI.

5.2.2 Weight Optimization with Linear Programming (WOLP)

Given a PPI network, the probability of interaction between any two proteins is measured in terms of how likely a random walk in the network starting at one node will reach the other node. Here, instead of solely using the adjacency matrix A to build the transition matrix, we integrate kernel features as edge strength. Then the stochastic transition matrix Q can be built by:

$$Q(i,j) = K_{fusion}(i,j) \tag{5.4}$$

Assuming the network is reasonably large, for a start node s, the probability distribution p of reaching all nodes via random walk in t steps can be obtained by applying the transition matrix Q t times:

$$p^t = Q^t p^0 \tag{5.5}$$

where the initial distribution p^0 is

$$p_i^0 = \begin{cases} 1, if\ i = s \\ 0, otherwise \end{cases} \tag{5.6}$$

The stationary distribution p, when letting t go to infinity, is obtained by solving the following eigenvector equation:

$$p = Q\ p \tag{5.7}$$

This stationary distribution provides constraints at optimizing the weights. For example, the positive training examples (nodes that are closer to the start node s) should have higher probability than the negative training examples (nodes that are far away from s). In Backstrom et al. [6], this is used as constraint in minimizing the L2 norm of the weights for optimal weights. In the work of Backstrom et al. [6], a gradient descent optimization method is adopted to get optimal weights, and only the pair-wise features for the existing edges in the network are utilized, which means $Q(i,j)$ is nonzero only for edge (i,j) that already exists in the training network. To leverage more information from multiple heterogeneous sources, in our case the $Q(i,j)$, as defined in Eq. (5.4), are nonzero unless there is no features for edge i,j in all kernels K_a. Having many

non-zero elements in Q makes it much more difficult for the traditional gradient descent optimization method to converge and to find the global optima.

In this work, we propose to solve the weights optimization differently. We can consider the random walk with restarts process shown in Eq. (5.5) as a Markov model, with a stationary distribution p. Knowing the stationary distribution, the transition matrix can be obtained by solving the reverse eign problem using the well-known Metropolis algorithm or Barker algorithm. In this work, we adopt Barker algorithm [7], which gives the transition matrix as follows.

$$Q^b(i,j) = \frac{p_j}{p_i + p_j} \tag{5.8}$$

Now we can formulate weights optimization by minimizing the element-wise difference between Q^b and Q. Namely,

$$W^* = \underset{W}{argmin}\, ||Q - Q^b||^2 \tag{5.9}$$

As the number of elements in the transition matrix is typically much larger than the number of weights, Eq. (5.9) provides more equations than the number of variables, making it an overdetermined linear equation system. This overdetermined linear equation system can be solved with linear programming using standard programs in [64, 27].

Now, in the spirit of supervised learning, given the training network G_{tn} and a start node s, we calculate p' by doing random walk that start at s in G_{tn} as an approximation of p, and $Q^b(i,j) = \frac{p'_j}{p'_i + p'_j}$. Note that $Q^b(i,j)$ from Barker algorithm is an asymmetric matrix whereas Q composed from kernel fusion is a symmetric matrix. So, we do not need to use all equations obtained from Eq. (5.9) to calculate the weights. Instead we can just use equations derived from the upper or lower triangle part of the matrices Q^b and Q. This reduction of number of equations will not pose an issue as the system is overdetermined; rather this will help mitigate the issue of being overdetermined. Specifically, as shown in Fig. 5.1, for all destination nodes V in G_{tn}, namely reachable from start node s, we divide them into three subsets D, L and M, where D consists of near neighbors of s in G_{tn} with the shortest path between s and

nodes D_i satisfying $d(s, D_i) < \epsilon 1$; and L includes faraway nodes of s in G_{tn} with the shortest path between s and nodes L_i satisfying $d(s, L_i) > \epsilon 2$; and the rest of nodes are in subset M. Then the system of equations of Eq. (5.9) is updated to Eq. (5.10), where $u < v$ indicates lower triangle mapping, and $u > v$ indicates upper triangle mapping.

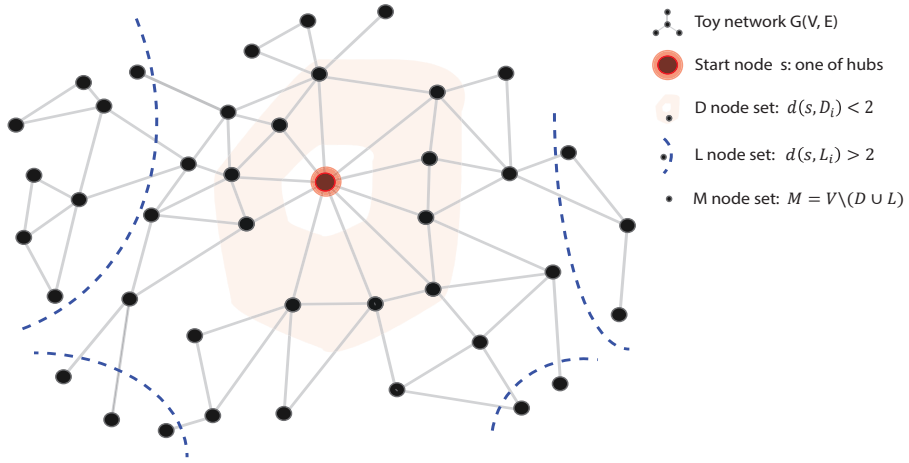


Figure 5.1: Schematic illustration of node sets D, M and L, with respect to source node s.

$$
\begin{aligned}
&W_0 G_{tn}(u, v) + \sum_{i=1}^{n} W_i K_i(u, v) = Q^b(u, v), \\
&if\ u, v \in D \cup L\ \wedge\ K_i(u, v)! = 0 \wedge (u < v \vee u > v)
\end{aligned}
\tag{5.10}
$$

The optimized weights W^* can then be plugged back into Eq. (5.4) to form an optimal transition matrix for the whole set of nodes, and the random walk from the source node using this optimal transition matrix hence leverages the information from multi data sources and is expected to give more accurate prediction for missing and/or de novo links: nodes that are most frequented by random walk are more likely, if not yet detected, to have a direct link to the source node. The formal procedure for solving this overdetermined linear system and inferring PPIs for a particular node is shown by Algorithm 6.

5.2.3 PPI Prediction and Network Inference

As we discussed in introduction section, the use of random walk from a single start node is not efficient for all-against-all prediction, especially for the large and

Algorithm 6 *Basic WOLP*

Input: $G_{tn} \leftarrow$ training network
$s \leftarrow$ start node
$K \leftarrow$ feature kernels
Output: p
1: $p' \leftarrow RWR(G_{tn}, s)$ // *RWR* is the Random Walk with Restart algorithm [90]
2: $Q^b(i,j) \leftarrow \frac{p'_j}{p'_i + p'_j}$
3: $W \leftarrow$ *by solving Eq.*(5.10) // W indicates the optimal weights
4: $Q \leftarrow$ *by Eq.*(5.2) *and Eq.*(5.4)
5: $p \leftarrow RWR(Q, s)$

Algorithm 7 *PPI Inference(WOLP) for the Connected PPI Network*

Input: $RL \leftarrow$ Regularized Laplacian prediction kernel
$G \leftarrow$ PPI network
$K \leftarrow$ feature kernels with same size of G
Output: Inferred network
1: $\{G_{tn}, G_{vn}, G_{tt}\} \leftarrow G$
2: $W^{opt} \leftarrow$ *Supervised WOLP*$(G_{tn}, G_{vn}, G_{tt}, RL, K)$
3: *OPT-K*$\leftarrow W_0^{opt} G_{tn} + \sum_{i=1}^{n} W_i^{opt} K_i$ // *OPT-K* is the optimal kernel fusion based on weights learned by WOLP
4: $RL_{OPT\text{-}K} \leftarrow RL(OPT\text{-}K)$ // Apply RL model to the kernel fusion
5: Rank $RL_{OPT\text{-}K}$ and infer G_{tt}

sparse PPI networks. Therefore, it would be of great interest if the weights learned by WOLP based on a single start node can also work network wide. Actually, it is widely observed that the many biological networks contain several hubs (i.e., nodes with with high degree) [9]. Thus we extend our algorithm to all-against-all PPI inference by hypothesizing that the weights learned based on a start node with high degree would be utilizable by other nodes. We will verify this hypothesis by doing all-against-all PPI inference for real PPI network.

We design a supervised WOLP version that can learn weights more accurately for the large and sparse PPI network. Similarly, if the whole PPI network is connected, then the golden standard network is itself; otherwise, the golden standard network that used to do supervised learning should be a large component of the disconnected PPI network. To do so, we divide the golden standard network into three parts: connected training network $G_{tn} = (V, E_{tn})$, validation set $G_{vn} = (V_{vn}, E_{vn})$ and testing set $G_{tt} = (V_{tt}, E_{tt})$, such that $E = E_{tn} \cup E_{vn} \cup E_{tt}$, and any edge in G can only belong

Algorithm 8 *Supervised WOLP*

Input: $G_{tn}, G_{vn}, G_{tt}, RL, K$
Output: W^{opt}

$s \leftarrow$ a start node with large degree in G_{tn}
$r \leftarrow$ diameter of G_{tn} respect to the start node s
$D \leftarrow$ direct neighbors of start node s
while $r > 1$ **do**
  $L \leftarrow V_i \; if \; d(s, V_i) >= r$ // V is the nodes set of G_{tn}, d is the shortest path
  $p' \leftarrow RWR(G_{tn}, s)$ // random walk with restarts from start node s in G_{tn}
  $Q^b(i,j) \leftarrow \frac{p'_j}{p'_i + p'_j}$
  $W1 \leftarrow$ *by solving Eq.*(5.10) with upper triangle mapping
  $W2 \leftarrow$ *by solving Eq.*(5.10) with lower triangle mapping
  $OPT\text{-}K1 \leftarrow W1_0 G_{tn} + \sum_{i=1}^{n} W1_i K_i$
  $OPT\text{-}K2 \leftarrow W2_0 G_{tn} + \sum_{i=1}^{n} W2_i K_i$
  $R1 \leftarrow Inference(RL, OPT\text{-}K1, G_{vn})$
  $R2 \leftarrow Inference(RL, OPT\text{-}K2, G_{vn})$
  // In the $Inference$ function, RL has been applied to kernel fusion $OPT\text{-}Ki$ to infer validation edges G_{vn}. Ri represent results of inferring G_{vn}
  if $R1 > R^{opt}$ **then**
    $R^{opt} \leftarrow R1$
    $W^{opt} \leftarrow W1$ // R^{opt} indicates the optimal result of inferring G_{vn}, W^{opt} indicates the optimal weights
  end if
  if $R2 > R^{opt}$ **then**
    $R^{opt} \leftarrow R2$
    $W^{opt} \leftarrow W2$
  end if
  $r \leftarrow r - 1$
end while

to one of these three parts. Then we use WOLP to learn weights based on G_{tn} and G_{vn}, and finally use G_{tt} to verify the prediction capability of these weights. The main structure of our method is shown by Algorithm 7, and the supervised version of WOLP is shown by Algorithm 8. The *while* loop in Algorithm 8 is used to find optimal setting of D, L and mapping strategy(upper or lower) that can generate best weights W_{opt} with respect to inferring and G_{tn} and G_{vn}.

Moreover, many existing network-level link prediction or matrix completion methods [6, 53, 16, 96, 62] can only work well on connected PPI networks, but detection of interacting pairs for disconnected PPI networks has been a challenge for these methods. However, our WOLP method can solve the problem effectively. Because various feature kernels can connect all the disconnected components of the originally disconnected PPI network; and we believe once the optimal weights have been learned based on the training network generated from a large connected component (e.g. largest connected component), they can also be used to build the kernel fusion when the prediction task scale up to the originally disconnected PPI network. To do so, we update the Algorithm 7 to Algorithm 9 that shows the detailed process of interaction prediction for disconnected PPI networks. Given an originally disconnected network G, firstly, we learn the optimal weights by Algorithm 8 based on a large connected component G_{cc} of G. After that, we randomly divide the edge set E of the disconnected G into training edge set G_{tn} and testing edge set G_{tt}, and use the optimal weights we learned before directly to linearly combine G_{tn} and other corresponding feature kernels to build the kernel fusion, and finally evaluate the performance through predicting G_{tt}. Here we call G_{tn} training edge set, because G_{tn} no longer needs to be connected to learn any weights.

5.3 Results and Discussion

We examine the soundness and robustness of the proposed algorithms with use of both synthetic and real data. Our goal here is to demonstrate that the weights

Algorithm 9 *PPI Inference(WOLP) for the Disconnected PPI Network*

Input: $RL \leftarrow$ Regularized Laplacian prediction kernel
$G \leftarrow$ disconnected PPI network
$K \leftarrow$ feature kernels with same size of G
Output: Inferred network
1: $G_{cc} \leftarrow G$ // get a large connected component from G
2: $\{G_{tn'}, G_{vn'}, G_{tt'}\} \leftarrow G_{cc}$
3: $W^{opt} \leftarrow$ *Supervised WOLP*$(G_{tn'}, G_{vn'}, G_{tt'}, RL, K')$ // K' is a sub matrix of K with same size of $G_{tn'}$
4: $\{G_{tn}, G_{tt}\} \leftarrow G$
5: *OPT-K*$\leftarrow W_0^{opt} G_{tn} + \sum_{i=1}^{n} W_i^{opt} K_i$ // *OPT-K* is the optimal kernel fusion based on weights learned by WOLP
6: $RL_{OPT\text{-}K} \leftarrow RL(OPT\text{-}K)$ // Apply RL model to the kernel fusion
7: Rank $RL_{OPT\text{-}K}$ and infer G_{tt}

obtained by our method can help build a better kernel fusion leading to more accurate PPI prediction.

5.3.1 Experiments on Single Start Node and Synthetic Data

A synthetic scale-free network G_{syn} with 5,093 nodes is generated by Copying model [54]: G_{syn} starts with three nodes connected in a triad. Remaining nodes have been added one by one with exactly two edges for each. For instance, when a node u is added, two edges$(u, v_i), i = 1, 2$ between u and existing nodes v_i will be added accordingly. Node v_i is randomly selected with probability 0.8, and otherwise v_i is selected with probability proportional to its current degree. The parameters we chose is to guarantee G_{syn} has similar size and density to DIP yeast PPI network [79] that we will use to do PPI inference later. Then we build eight synthetic feature kernels for G_{syn}. The feature kernels can be classified into three categories: 3 noisy kernels, 4 positive kernels and a mixture kernel, which are defined by Eq. (5.11)

$$\begin{cases} K_{noise} = R_{5093} + (R_{5093} + \eta). * rand_{diff}(J_{5093}, G_{syn}, \rho_i) \\ K_{postive} = R_{5093} + (R_{5093} + \eta). * rand_{sub}(G_{syn}, \rho_i) \\ K_{mixture} = R_{5093} + (R_{5093} + \eta). * rand_{sub}(G_{syn}, \rho_i) \\ \qquad +(R_{5093} + \eta). * rand_{diff}(J_{5093}, G_{syn}, \rho_i) \end{cases} \tag{5.11}$$

Where R_{5093} indicates a 5093 by 5093 random matrix with elements between $[0, 1]$, which can also be seen as background noise matrix; J_{5093} indicates a 5093 by 5093 all-one matrix, $rand_{diff}(J_{5093}, G_{syn}, \rho_i)$ is used to randomly generate a difference matrix (if (i, j) = 1 in G_{syn} and (i, j) should be 0 in the difference matrix) between J_{5093} and G_{syn} with density ρ_i; $rand_{sub}(G_{syn}, \rho_i)$ is used to generate a subnetwork from G_{syn} with density ρ_i; ρ_i are different for each kernel; η is a positive parameter between $[0, 1]$ and R_{5093} will be rebuilt every time for each kernel.

The general process of experimenting with synthetic data is: we generate synthetic network G_{syn}, synthetic feature kernels K firstly, and then divide nodes V of G_{syn} into D, L and M, where D and L can be seen as training nodes, M can be seen as testing nodes. By using G_{syn}, start node s and K, we can get the stationary distribution p based on the optimized kernel fusion $K_{OPT} = W_0 G_{syn}(u, v) + \sum_{i=1}^{n} W_i K_i(u, v)$. Finally, we try to prove that K_{OPT} is better than the control kernel fusion $K_{EW} = G_{syn} + \sum_{i=1}^{n} K_i$ built by equal weights, if the $p(M)$ is more similar to $p'(M)$ based on G_{syn}, as compared to $p''(M)$ based on the control kernel fusion K_{EW}, where $p(M)$ indicates the rank of stationary probabilities respect to the testing node M. We evaluate the rank similarity between pairs $(p(M), p'(M))$ and $(p''(M), p'(M))$ by discounted cumulative gain (DCG) [20].

We carry out 10 experiments, each time we select one of the oldest 3 nodes as start node, and rebuild synthetic kernel K. In Table 5.1, the results show that DCG@20 between $p(M)$ and $p'(M)$ is consistently higher than that between $p''(M)$ and $p'(M)$ in all 10 experiments, indicating that the optimal weights W obtained by WOLP can help us build optimized kernel fusion that with better prediction capability, as compared to the control kernel fusion.

5.3.2 Experiments on Network Inference with Real Data

We use the yeast PPI network downloaded from DIP database (Release 20150101) [79] and the high-confidence human PPI network downloaded from PrePPI database [101] to test our algorithm.

Table 5.1: DCG@20 of rank comparison

Repetition	DCG@20$(p(M), p'(M))$	DCG@20$(p''(M), p'(M))$
1	0.7101	0.6304
2	0.9305	0.4423
3	0.4035	0.2657
4	0.8524	0.5690
5	0.7256	0.4417
6	0.3683	0.3009
7	0.7707	0.2753
8	1.0034	0.3663
9	0.7119	0.4603
10	0.6605	0.6123

5.3.2.1 Data and Kernels of Yeast PPI Networks

For the yeast PPI network, some interactions without Uniprotkb ID have been filtered out in order to do name mapping and make use of genomic similarity kernels [56]. As a result, the originally disconnected PPI network contains 5,093 proteins and 22,423 interactions. The largest connected component consists of 5,030 proteins and 22,394 interactions, and is used to serve as the golden standard network. Same to the subsection 4.2.4 in Chapter 4, six feature kernels are included in this study.

5.3.2.2 Data and Kernels of Human PPI Networks

The originally disconnected human PPI network has 3,993 proteins and 6,669 interactions, which is much sparser than the yeast PPI network. The largest connected component that serve as the golden standard network contains 3,285 proteins and 6,310 interactions.

Eight feature kernels are included in PPI inference for the human data.

G_{tn}: G_{tn} is the connected training network that provides connectivity information. It can also be thought of as a base network to do the inference.

$K_{Jaccard}$ [45]: It measures the similarity of protein pairs i, j in term of $\frac{neigbors(i) \cap neighbors(j)}{neighbors(i) \cup neighbors(j)}$.

K_{SN}: It measures the total number of neighbors of protein i and j, $K_{SN} = neighbors(i) + neighbors(j)$.

Table 5.2: Division of golden standard PPI networks

Species	G_{tn}	G_{vn}	G_{tt}
Yeast	$V, E = \{5,030, 5,394\}$	$V, E = \{-, 1,000\}$	$V, E = \{-, 16,000\}$
Human	$V, E = \{3,285, 3,310\}$	$V, E = \{-, 300\}$	$V, E = \{-, 2,700\}$

K_B: It is a sequence-based kernel matrix that is generated using the BLAST [5].

K_D: It is a domain-based similarity kernel matrix measured by the method of neighborhood correlation [82].

K_{BP}: It is a biological process based semantic similarity kernel measured by Resnik with BMA [76].

K_{CC}: It is a cellular component based semantic similarity kernel measured by Resnik with BMA [76].

K_{MF}: It is a molecular function based semantic similarity kernel measured by Resnik with BMA [76].

5.3.2.3 PPI Inference based on the Largest Connected Component

For cross validation, the golden standard PPI network (largest connected component) is randomly divided into three parts that are connected training network G_{tn}, validation edge set G_{vn} and testing edge set G_{tt}, where G_{vn} is used to find optimal weights for feature kernels and G_{tt} is used to evaluate the inference capability of our method. The Table 5.2 shows detailed division for yeast and human PPI networks. With the weights learned by WOLP and using i_{th} hub as the start node, we build the kernel fusion *WOLP-K-i* by Eq. (5.2). PPI network inference is made by RL kernel Eq. (5.3), and named as $RL_{WOLP\text{-}K\text{-}i}$, $i = 1, 2, 3$. The performance of inference is evaluated by how well the testing set G_{tt} is recovered. Specifically, all node pairs are ranked in decreasing order by their edge weights in the RL matrix, and edges in the testing set G_{tt} are labeled as positive and node pairs with no edges in G are labeled as negative. An ROC curve is plotted for true positive v.s. false positives, by running down the ranked list of node pairs. To make comparison, besides the PPI inferences

$RL_{WOLP\text{-}K\text{-}i}$, $i = 1, 2, 3$ learned by our WOLP, we also include other two PPI network inferences: $RL_{G_{tn}}$ and $RL_{EW\text{-}K}$, where $RL_{G_{tn}}$ indicates RL based PPI inference is solely from the training network G_{tn}, and $RL_{EW\text{-}K}$ represents RL based PPI inference is from kernel fusion built by equal weights, e.g. $w_i = 1$, $i = 0, 1...n$. Additionally, $G_{set} \sim n$ indicates there is n number of edges in the set G_{set}, e.g. $G_{tn} \sim 5394$ means the connected training network G_{tn} contains 5,394 edges.

The comparisons in terms of ROC curve and AUC for yeast and human data are shown by Fig. 5.2 and Fig. 5.3, the PPI reference $RL_{WOLP\text{-}K\text{-}i}$, $i = 1, 2, 3$ based on our WOLP method significantly outperforms the two basic control methods, with about 17% increase over $RL_{G_{tn}}$ and about 19.6% over $RL_{EW\text{-}K}$ in term of AUC for the yeast data, and about 12.7% increase over $RL_{G_{tn}}$ and about 11.3% over $RL_{EW\text{-}K}$ in term of AUC for the human data. It is noted that the AUC of PPI inference $RL_{EW\text{-}K}$ based on the equally weighted built kernel fusion is no better or even worse than that of $RL_{G_{tn}}$ based on a really small training network, especially for the yeast data. It means there should be a lot of noises if we just naively combine different feature kernels to do PPI prediction.

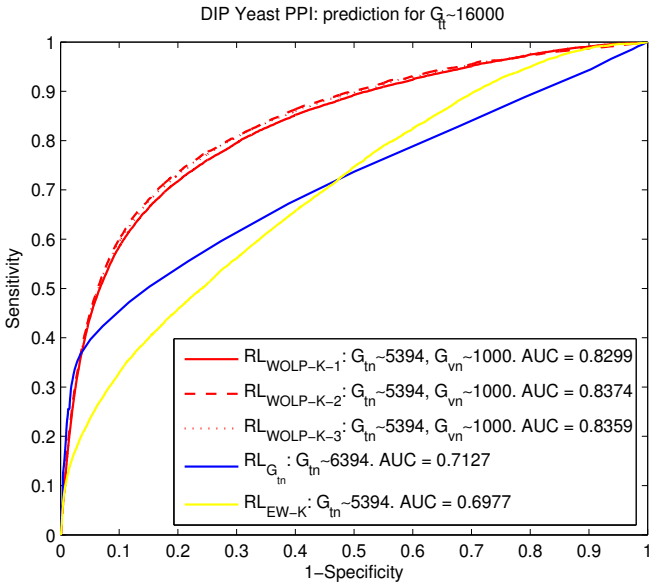


Figure 5.2: Yeast: ROC curves of predicting $G_{tt} \sim 16000$ by $RL_{G_{tn} \sim 5394}$, $RL_{WOLP\text{-}K\text{-}i}$ and $RL_{EW\text{-}K}$.

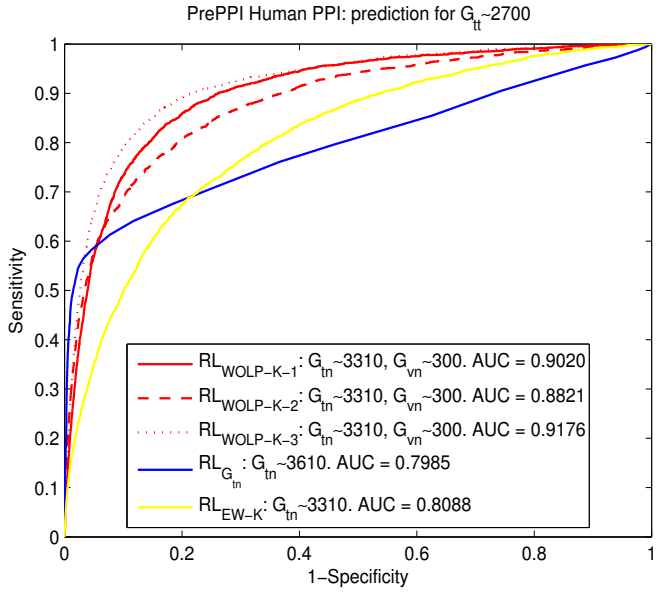


Figure 5.3: Human: ROC curves of predicting $G_{tt} \sim 2700$ by $RL_{G_{tn}\sim 3610}$, $RL_{WOLP\text{-}K\text{-}i}$ and $RL_{EW\text{-}K}$.

Besides inferring PPI network by using weights learned based on the top three hubs in G_{tn}, we also test the predicting capability of PPI inferences by using top ten hubs as start nodes to learn the weights. We make 10 repetitions for the whole process: generating G_{tn}, choosing $i_{th},\ i = 1, 2, ...10$ hub as start node to learn the weights, then using these weights to build kernel fusion and finally to do the PPI inference. For the results based on top ten hubs in each repetition, the average AUC of inferring G_{tt} for yeast data and human data are shown in Table 5.3 and Table 5.4 respectively. And the comparison shows the predicting capability of our method is consistently better than that of $RL_{G_{tn}}$ and $RL_{EW\text{-}K}$ for both yeast and human data.

5.3.2.4 Effects of the Training Data

Usually, given a golden standard data, we need to retrain the prediction model for different division of training set and testing set. However, if optimal weights have been found for building kernel fusion, our PPI network inference method enable us to train the model once, and do prediction or inference for different testing sets. To demonstrate that, we keep the two PPI inferences $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ obtained

Table 5.3: Comparison of AUCs for yeast PPI prediction

Rep	Avg AUC($RL_{WOLP\text{-}K\text{-}1\sim10}$)	AUC($RL_{G_{tn}}$)	AUC($RL_{EW\text{-}K}$)
1	0.8367± 0.0134	0.7127	0.6976
2	0.7937± 0.0584	0.7768	0.7014
3	0.7802± 0.0545	0.7732	0.7009
4	0.7811± 0.0507	0.7406	0.7029
5	0.8349± 0.0301	0.7477	0.6991
6	0.8160± 0.0492	0.7180	0.7091
7	0.7670± 0.0636	0.7513	0.6992
8	0.8018± 0.0539	0.7739	0.7042
9	0.7989± 0.0552	0.7302	0.7017
10	0.8172± 0.0388	0.7387	0.6953

Table 5.4: Comparison of AUCs for human PPI prediction

Rep	Avg AUC($RL_{WOLP\text{-}K\text{-}1\sim10}$)	AUC($RL_{G_{tn}}$)	AUC($RL_{EW\text{-}K}$)
1	0.8871± 0.0122	0.8228	0.7823
2	0.8986± 0.0144	0.8106	0.8127
3	0.8988± 0.0088	0.8216	0.8088
4	0.8955± 0.0114	0.8161	0.8142
5	0.8994± 0.0089	0.8190	0.8088
6	0.8875± 0.0182	0.7927	0.8067
7	0.8904± 0.0237	0.8302	0.8096
8	0.8978± 0.0121	0.8205	0.8153
9	0.9011± 0.0101	0.7995	0.8130
10	0.8818± 0.0281	0.8078	0.8104

Table 5.5: Effects of training data size on prediction performance (AUC) for yeast

	$G_{tt} \sim 15000$	$G_{tt} \sim 14000$	$G_{tt} \sim 13000$
$RL_{WOLP\text{-}K\text{-}1:G_{tn}\sim 5394}$	0.8658	-	-
$RL_{G_{tn}\sim 7394}$	0.7931	-	-
$RL_{EW\text{-}K:G_{tn}\sim 5394}$	0.7519	-	-
$RL_{WOLP\text{-}K\text{-}1:G_{tn}\sim 5394}$	-	0.8659	-
$RL_{G_{tn}\sim 8394}$	-	0.8538	-
$RL_{EW\text{-}K:G_{tn}\sim 5394}$	-	0.7537	-
$RL_{WOLP\text{-}K\text{-}1:G_{tn}\sim 5394}$	-	-	0.8659
$RL_{G_{tn}\sim 9394}$	-	-	0.8619
$RL_{EW\text{-}K:G_{tn}\sim 5394}$	-	-	0.7520

before (in last section) unchanged, and evaluate the prediction ability for different testing sets. We also examine how performance is affected by sizes of various sets. Specifically, while the size of training network G_{tn} for $RL_{G_{tn}}$ increases, sizes of G_{tn} for $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ are kept unchanged. Therefore, we design several experiments by dividing the golden standard network into G^i_{tn} and G^i_{tt}, $i = 1, ..., n$, and building PPI inference $RL_{G^i_{tn}}$ to predict G^i_{tt} for every time. To make comparison, we also use $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ to predict G^i_{tt}. As shown by the Table 5.5, for yeast data, $RL_{WOLP\text{-}K\text{-}1}$ trained on only 5,394 golden standard edges still performs better than the control methods, even for the $RL_{G_{tn}}$ that employ significantly more golden standard edges. Similarly, for the result of human data as shown by the Table 5.6, $RL_{WOLP\text{-}K\text{-}1}$ trained on only 3,310 golden standard edges still performs better than the control method $RL_{G_{tn}}$ that employ over 1,000 more golden standard edges.

5.3.2.5 Detection of Interacting Pairs Far Apart in the Network

It is known that the basic idea of using random walk or random walk based kernels [90, 58, 6, 28] for PPI prediction is that good interacting candidates usually are not faraway from the start node, e.g. only 2, 3 edges away in the network. Consequently, the testing nodes have been chosen to be within a certain distance range, which largely contributes to the good performance reported by many network-level link prediction

Table 5.6: Effects of training data size on prediction performance (AUC) for human

	$G_{tt} \sim 2600$	$G_{tt} \sim 2100$	$G_{tt} \sim 1600$
$RL_{WOLP\text{-}K\text{-}1:G_{tn}\sim 3310}$	0.9277	-	-
$RL_{G_{tn}\sim 3710}$	0.8359	-	-
$RL_{EW\text{-}K:G_{tn}\sim 3310}$	0.8590	-	-
$RL_{WOLP\text{-}K\text{-}1:G_{tn}\sim 3310}$	-	0.9305	-
$RL_{G_{tn}\sim 4210}$	-	0.8779	-
$RL_{EW\text{-}K:G_{tn}\sim 3310}$	-	0.8620	-
$RL_{WOLP\text{-}K\text{-}1:G_{tn}\sim 3310}$	-	-	0.9338
$RL_{G_{tn}\sim 4710}$	-	-	0.9227
$RL_{EW\text{-}K:G_{tn}\sim 3310}$	-	-	0.8639

methods. In reality, however, a method that is capable and good at detecting interacting pairs far apart in the network can be even more useful, such as in uncovering cross talk between pathways that are not nearby in the PPI network. To investi-

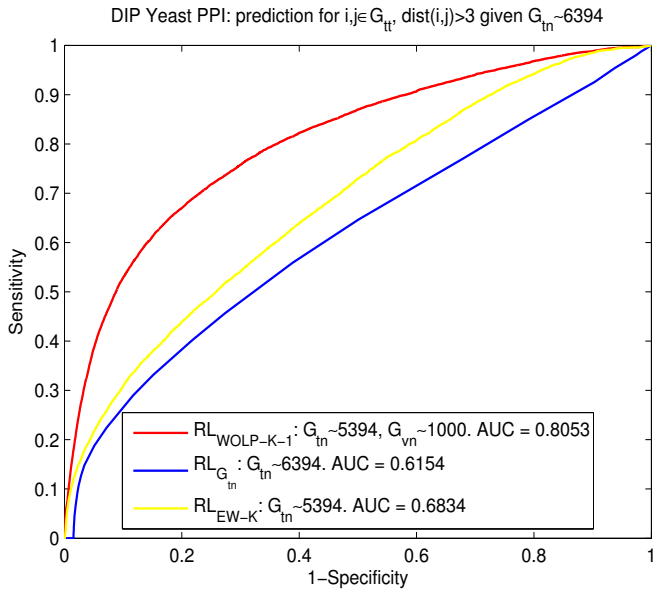


Figure 5.4: Yeast: ROC curves of predicting $G_{tt}^{(dist(i,j)>3)}$ by $RL_{G_{tn}\sim 6394}$, $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$.

gate how our proposed method performs at detecting faraway interactions, we still use $RL_{G_{tn}\sim 6394}$, $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ for yeast data, and $RL_{G_{tn}\sim 3610}$, $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ for human data to infer PPIs, but we select node pairs (i, j) that satisfy

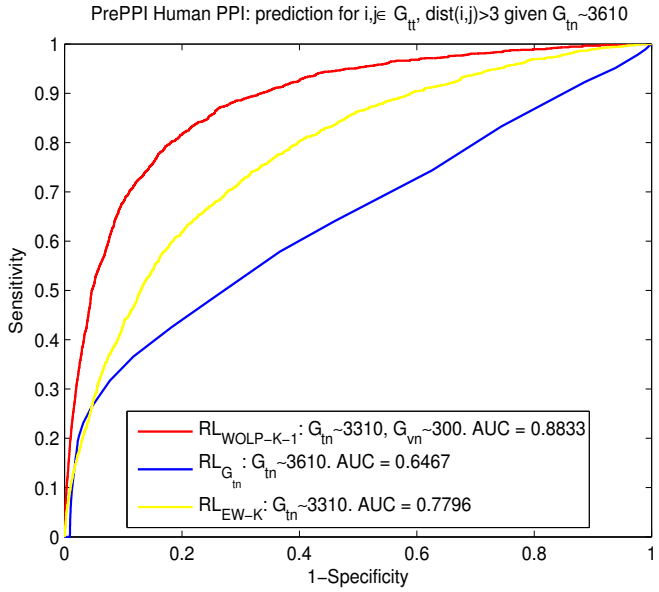


Figure 5.5: Human: ROC curves of predicting $G_{tt}^{(dist(i,j)>3)}$ by $RL_{G_{tn}\sim 3610}$, $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$.

$dist(i,j) > 3$ *given* G_{tn} from G_{tt} as new testing set and name it $G_{tt}^{(dist(i,j)>3)}$. Fig. 5.4 and Fig. 5.5 show the results of yeast and human data respectively, which demonstrate that $RL_{WOLP\text{-}K\text{-}1}$ has not only a significant margin over the control methods in detecting long-distance PPIs but also maintains a high ROC scores of 0.8053 (for yeast data) and 0.8833 (for human data) comparable to that of all PPIs. In contrast, in both Fig. 5.4 and Fig. 5.5, $RL_{G_{tn}}$ performs poorly and worse than $RL_{EW\text{-}K}$, which means the traditional RL kernel based on adjacent training network alone cannot detect faraway interactions well.

5.3.2.6 Detection of Interacting Pairs for Disconnected PPI Networks

For the originally disconnected yeast PPI network, we randomly divide the edge set E into training edge set G_{tn} with 6,295 edges and testing edge set G_{tt} with 16,128 edges. Similarity, based on a random division, the number of edges of training edge set G_{tn} and testing edge set G_{tt} are 3,305 and 3,364 for the originally disconnected human PPI network. The detailed information of the originally disconnected yeast and human PPI networks can be found in the subsection of data description. The Fig. 5.6 and Fig.

5.7 show the predicting results of yeast and human data respectively, which indicate $RL_{WOLP\text{-}K\text{-}i}$, $i = 1, 2, 3$ perform steady well on inferring interactions for both yeast and human data and are obviously better than $RL_{EW\text{-}K}$. $RL_{G_{tn}}$ is not included in this comparison, because it is not feasible for prediction tasks of disconnected PPI networks.

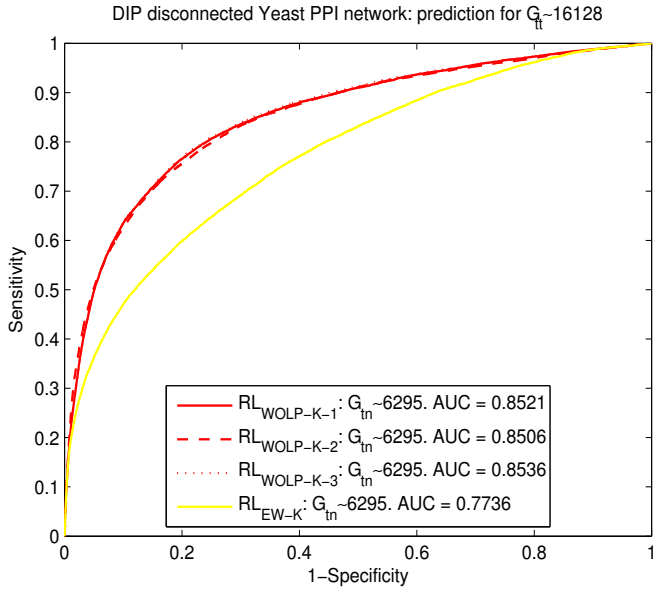


Figure 5.6: Yeast: ROC curves of predicting $G_{tt} \sim 16128$ by $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ for disconnected PPI network.

5.3.2.7 Analysis of Weights

As our method incorporates multiple heterogeneous data, it can be insightful to inspect the final optimal weights. Therefore, we compare the average of weights learned by WOLP to the average of weights learned from revised ABC-DEP sampling method [40], which is more computationally demanding. For the yeast data, the Fig. 5.8 shows that these two methods produce consistent results: these weights indicate that K_{SN} and K_{Pfam} are the predominant contributors to PPI prediction. This observation is consistent with the intuition that proteins interact via interfaces made of conserved domains [23], and PPI interactions can be classified based on their domain families and domains from the same family tend to interact [44, 66, 12]. For the human data, due to

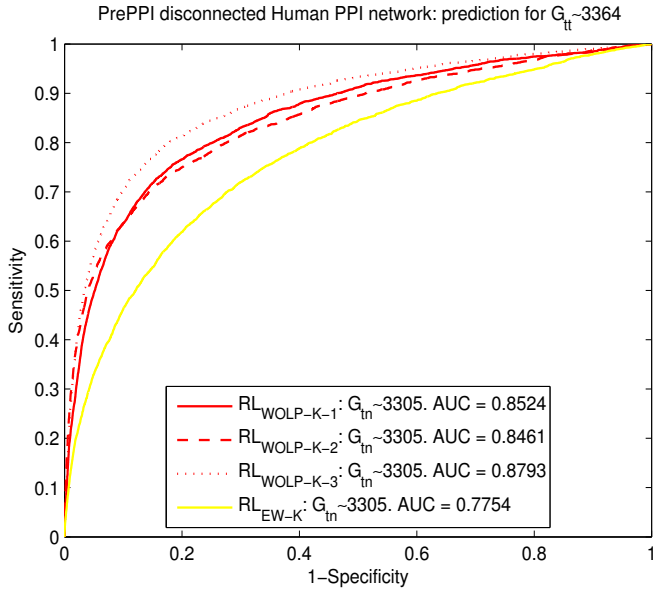


Figure 5.7: Human: ROC curves of predicting $G_{tt} \sim 3364$ by $RL_{WOLP\text{-}K\text{-}1}$ and $RL_{EW\text{-}K}$ for disconnected PPI network.

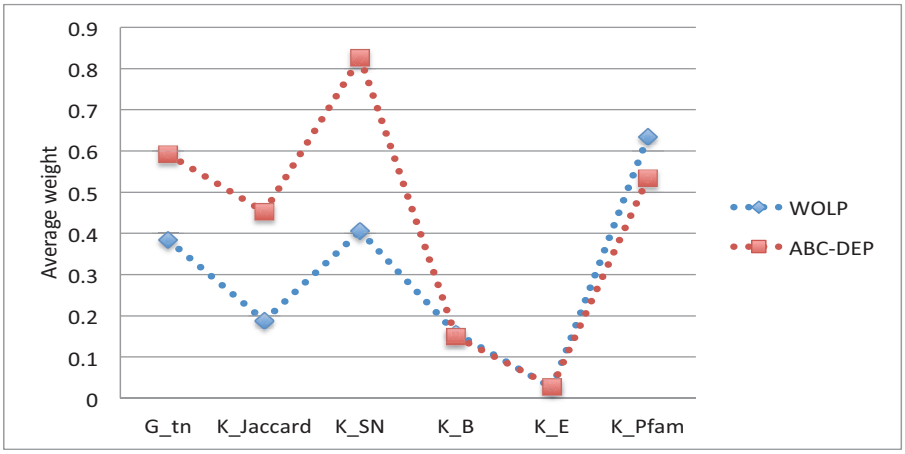


Figure 5.8: Yeast: comparison of average weights learned by WOLP and ABC-DEP sampling method.

the extreme sparsity of the human PPI network, limited golden standard interactions can be included in the validation set to help optimize weights, which makes the weight optimization problem more challenging, especially for the sampling method. Although the result of human data that shown in Fig. 5.9 is not good as that of the yeast data, these two methods also produce quite consistent distribution, and K_{SN} is the most

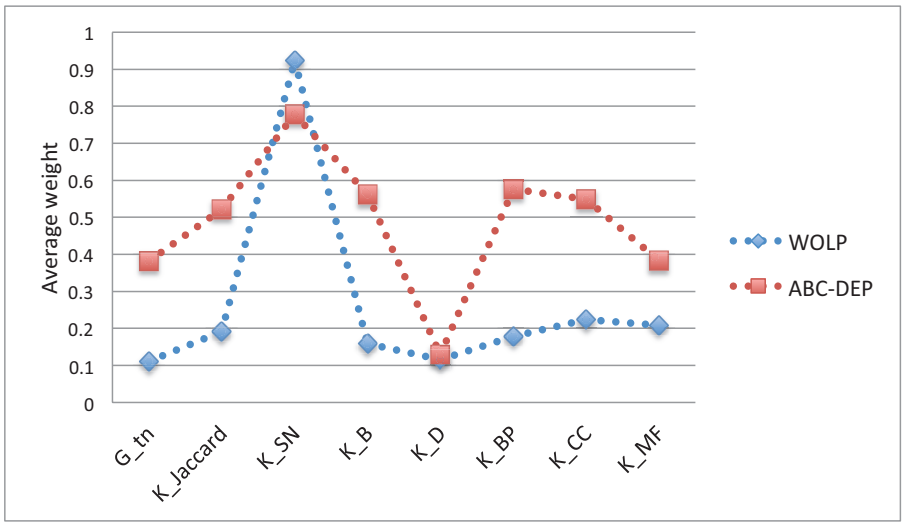


Figure 5.9: Human: comparison of average weights learned by WOLP and ABC-DEP sampling method.

predominant contributor. Although the true strength of our method lies in integrating multiple heterogeneous data for PPI network inference, the optimal weights can serve as a guidance to select most relevant features when time and resources are limited.

5.4 Conclusions

In this chapter we developed a novel and fast optimization method using linear programming to integrate multiple heterogeneous data for PPI inference problem. The proposed method, verified with synthetic data and tested with DIP yeast PPI network and PrePPI high-confidence human PPI network , enables quick and accurate inference of PPI networks from topological and genomic feature kernels in an optimized integrative way. Compared to the baseline (G_{tn} and EW-K), our WOLP method achieved performance improvement in PPI prediction with over 19% higher AUC on yeast data and 11% higher AUC on human data, and this margin is maintained even when the control methods use a significantly larger training set. We also demonstrated that by integrating topological and genomic features into regularized Laplacian kernel, the method avoids the short-range problem encountered by random-walk based methods – namely the inference becomes less reliable for nodes that are far from the start node of

the random walk, and shows obvious improvements on predicting faraway interactions; The weights learned by our WOLP are highly consistent with the weights learned by sampling based method, which can provide insights into the relations between PPIs and various similarity features of protein pairs, thereby helping us make good use of these features. Moreover, we further demonstrated those relations are also maintained when the golden standard network (largest connected component) scale up to the original PPI network that consists of disconnected components. That is to say, the weights learned based on the connected training subnetwork of the largest connected component can also help to detect interactions for the originally disconnected PPI networks effectively and accurately. As more features with respect to proteins are collected from various -omics studies, they can be used to characterize protein pairs in terms of feature kernels from different perspectives. Thus we believe that our method can provide us a quick and accurate way to fuse various feature kernels from heterogeneous data, thereby improving PPI prediction.

Chapter 6

COMPLETING SPARSE AND DISCONNECTED PROTEIN-PROTEIN NETWORK BY DEEP LEARNING AND REGULARIZED LAPLACIAN KERNEL

6.1 Introduction

Many of existing network level methods predict PPIs under the assumption that the training network should be connected. However, that assumption greatly affects their prediction power and limits the application area because current golden standard PPI networks are usually very sparse and disconnected. Therefore, how to effectively predict PPIs based on a training network that has limited interactions but a large amount of disconnected components remains a challenge. In this chapter, we develop novel PPI prediction method based on deep learning neural network and regularized Laplacian kernel. Inspired by the study of network evolutionary analysis discussed in Chapter 2, we use the neural network to implicitly simulate and guide the evolution process of a PPI network by using rows of an ancient network as inputs and rows of the disconnected training network as labels. Instead of simulating the evolution of the whole network structure with the growth of nodes and edges as models discussed in Chapter 2, we only focus on the edge evolution and assume nodes are already existing. After the training step, an evolved contact matrix whose rows are outputs of the neural network can be obtained. Then we can predict PPIs by applying the regularized Laplacian kernel to the transition matrix that is built upon the evolved PPI network. The results from cross-validation experiments show that the PPI prediction accuracies for yeast data and human data measured as AUC are increased by up to 8.4% and 14.9% respectively, as compared to the baseline. Moreover, the evolved PPI network can also help us leverage complementary information from the disconnected training

network and multiple heterogeneous data sources. Tested by the yeast data with six heterogeneous feature kernels, the results show our method can further improve the prediction performance by up to 2%, which is very close the upper bound that is obtained by a sampling method.

6.2 Methods and Data

6.2.1 Problem Definition

Formally, a PPI network can be represented as a graph $G = (V, E)$ with V nodes (proteins) and E edges (interactions). G is defined by the adjacency matrix A with $V \times V$ dimension:

$$A(i,j) = \begin{cases} 1, if(i,j) \in E \\ 0, if(i,j) \notin E \end{cases} \tag{6.1}$$

where i and j are two nodes in the nodes set V, and (i, j) represents an edge between i and j, $(i, j) \in E$. We divide the golden standard network into two parts: the training network $G_{tn} = (V, E_{tn})$, and testing set $G_{tt} = (V_{tt}, E_{tt})$, such that $E = E_{tn} \cup E_{tt}$, and any edge in G can only belong to one of these two parts. The detailed process of dividing the golden standard network is shown by Algorithm 10. We set the α (the preset ratio of $G_{tn}(, E)$ to $G(, E)$) less than a small value to make the G_{tn} extremely sparse and with a large number of disconnected components.

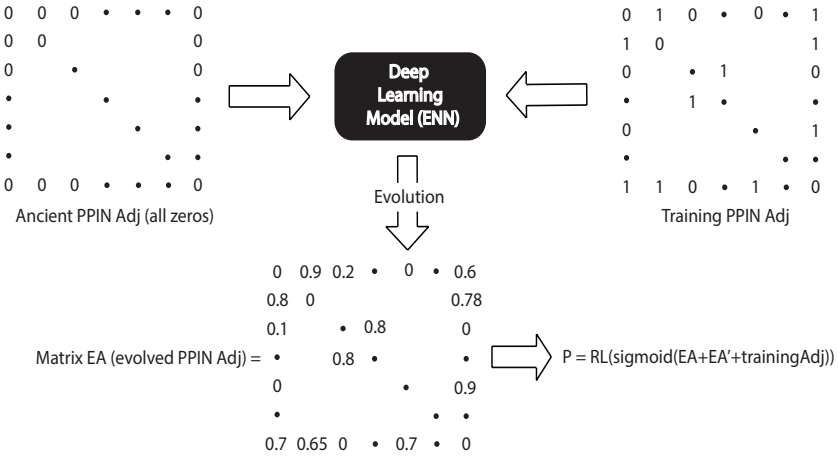


Figure 6.1: The flow chart of ENN-RL method.

The Fig. 6.1 shows the flow chart of our method that is named as evolution neural network based regularized Laplcian kernel (ENN-RL); and the Algorithm 11 describes the detailed training and prediction processes. We use the ENN to obtain an evolved PPI network, and thereby building a transition matrix for the regularized Laplacian kernel to predict PPIs. We assume the ancient PPI network already has m proteins but no interactions, which is represented by an all-zero $m \times m$ matrix. Then a deep learning model, the evolution neural network, is adopted to drive and guide the ancient PPI network to evolve interactions among proteins. To guide the evolutionary direction, we define the evolution as a supervised learning problem by using the training network G_{tn} as the target/label for the evolution neural network. Each protein is represented by the corresponding row of the matrix that contains the interaction information for that protein. We train the evolution neural network by each row of the ancient matrix as the input and the corresponding row of $trainingAdj$ as the label, where $trainingAdj$ is the adjacent matrix of G_{tn}. After the training process is completed, we build the evolved PPI network/matrix EA by the outputs of neural network's last layer. Based on EA, we build a transition matrix by Eq. (6.2), where $EA + EA'$ makes the transition matrix symmetric and positive semi-definite. Finally, we apply the regularized Laplacian (RL) kernel defined by Eq. (6.3) to the transition matrix T to get the inference matrix P, in which $P_{i,j}$ indicates the probability of an interaction for protein i and j. For Eq. (6.3), $L = D - T$ is the Laplacian matrix made of the transition matrix T and the degree matrix D, and $0 < \beta < \rho(L)^{-1}$ and $\rho(L)$ is the spectral radius of L.

$$T = sigmoid(EA' + EA + trainingAdj) \tag{6.2}$$

$$RL = \sum_{k=0}^{\infty} \beta^k (-L)^k = (I + \beta * L)^{-1} \tag{6.3}$$

Algorithm 10 *Division of Edges*

Input: $G \leftarrow$ PPI network
$m \leftarrow$ The number of nodes
$\alpha \leftarrow$ The preset ratio of $G_{tn}(, E)$ to $G(, E)$
Output: G_{tn} and G_{tt}

for *each node* $w \in V$ **do**
  $nb \leftarrow neighbors(w)$ // nb is neighbor set of node w
  $nb \leftarrow shuffle(nb)$ // Shuffle the neighbor set
  $t \leftarrow length(nb) * \alpha$ // Set a threshold for dividing neighbors of w
  for $i = 1$ to $length(nb) - 1$ **do**
    if $i < t$ **then**
      $G_{tn} \leftarrow G_{tn} \cup (w, nb[i])$ // $(w, nb[i])$ indicates an edge between w and $nb[i]$
    else
      if $(nb[i], w) \notin G_{tn}$ **then**
        $G_{tt} \leftarrow G_{tt} \cup (w, nb[i])$
      end if
    end if
  end for
end for

Algorithm 11 *ENN-RL PPI inference*

Input: $ENN \leftarrow$ Evolution neural network
$RL \leftarrow$ Regularized Laplacian prediction kernel
$G_{tn} \leftarrow$ Training network
$G_{tt} \leftarrow$ Testing set
$m \leftarrow$ The number of nodes
Output: Inferred interactions

$intialAdj \leftarrow allzero(m, m)$ // $intialAdj$ a $m \times m$ all-zero matrix
$trainingAdj \leftarrow edgesToAdjMatrix(G_{tn})$ // Transform edges into adjacency matrix
for $i \in 0, ..., m - 1$ **do**
  $input_i \leftarrow initialAdj[i][:]$ // $input_i$ is i^{th} row of initialAdj
  $label_i \leftarrow trainAdj[i][:]$ // $label_i$ is i^{th} row of trainAdj
  $ENN(input_i, label_i)$ // Training the evolution neural network ENN
end for
$EA \leftarrow allzero(m, m)$ // EA is a $m \times m$ all-zero matrix
for $i \in 0, ..., m - 1$ **do**
  $input_i \leftarrow initialAdj[i][:]$
  $EA[i] \leftarrow ENN(input_i)$ // $EA[i]$ is the output of last layer of ENN given the input $input_i$
end for
$P \leftarrow RL(sigmoid(EA + EA' + trainAdj))$ // Get the inference matrix P based on RL
Rank P and infer G_{tt}

6.2.2 Evolution Neural Network

The structure of the evolution neural network is shown in the Fig. 6.2, which contains five layers including the input layer, three hidden layers and the output layer. Sigmoid is adopted as the activation function for each neuron, and layers are connected with dropouts. Dropouts can not only help us prevent over-fitting, but also indicate the mutation events during the evolution process here. For specific configuration of the neural network in our experiments, the number of neurons in the input and output layer depends on the network size m of specific PPI data; a typical autoencoder structure is chosen for the three hidden layers, where encoder and decoder indicate biological devolution and evolution processes respectively; and cross entropy is used as the loss function. It is worth noting that different with the traditional autoencoder, we did not include the layerwise isomorphism pretraining to initial the weights for our neural network since the inputs are all zero vectors. The neural network is implemented by the TensorFlow library [2], deployed on Biomix cluster at Delaware Biotechnology Institute.

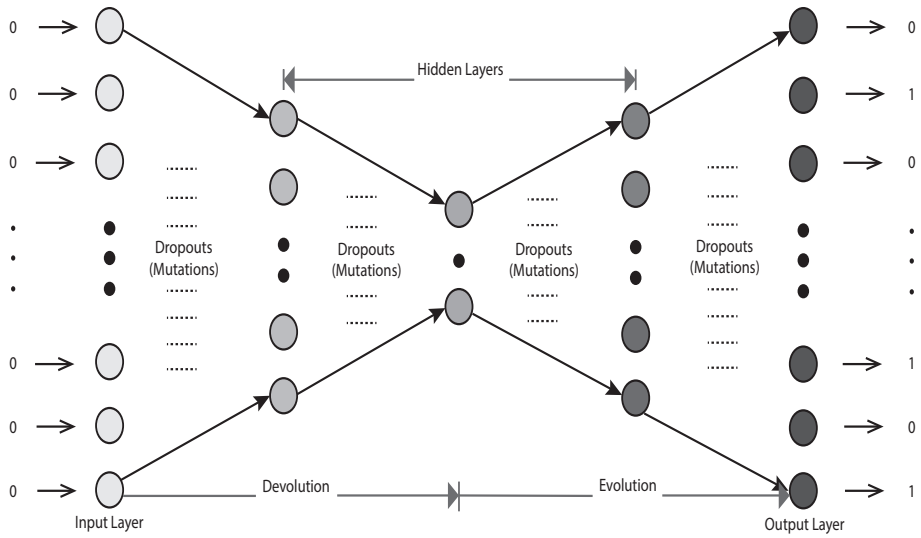


Figure 6.2: The evolution neural network ENN.

Table 6.1: PPI Network Information

Species	Proteins	Interactions
Yeast	5,093	22,423
Human	9,617	37,039

Table 6.2: Division of yeast golden standard PPI interactions

α	G_{tn}	$G_{tn}(\#C)$	$G_{tn}(minC, avgC, maxC)$	G_{tt}
0.25	4,061	2,812	(1, 1.81, 2,152,)	18,362
0.125	1,456	3,915	(1, 1.30, 1,006)	20,967

$G_{tn}(\#C)$: the number of components in G_{tn}
$G_{tn}(minC, avgC, maxC)$: the minimum, average and maximum size of components in G_{tn}

6.2.3 Data

We use yeast and human PPI networks downloaded from DIP (Release 20140117) [79] and HPRD (Release 9) [50] to train and test our method. After removing the self-interactions, the detailed information of these two datasets are shown in the Table 6.1.

6.3 Results and Discussion

6.3.1 Experiments on Yeast PPI Data

To show how well our model can predict PPIs from the extremely sparse training network with disconnected components, we set α, the ratio of interactions in G_{tn} to total edges in G, to be less than 0.25. As shown in the Table 6.2, the G_{tn} has only 4,061 interactions, and contains 2,812 disconnected components, where the minimum, average and maximum size of components are 1, 1.81 and 2,152 respectively. Based on the G_{tn}, we train our model and predict the large testing set G_{tt} that has 18,362 interactions according to the Algorithm 11.

Then we compared our ENN-RL method to the control method ADJ-RL which applies regularized Laplacian kernel directly to the training network G_{tn}. As shown in Fig. 6.3, the AUC increase from 0.8112 for the control method to 0.8358 for ENN-RL.

Table 6.3: AUC summary of repetitions for yeast PPI data

Methods	Avg±Std ($\alpha = 0.25$)	Avg±Std ($\alpha = 0.125$)
ENN-RL	0.8339± 0.0016	0.8195± 0.0023
ADJ-RL	0.8104± 0.0039	0.7403± 0.0083

Moreover, we make the prediction task more challenging by setting the α to be less than 0.125, which makes the G_{tn} sparser with only 1,456 interactions, but 3,915 disconnected components; and the maximum component in G_{tn} only has 1,006 interactions. The results in Fig. 6.4 shows the gap between ENN-RL ROC curve and ADJ-RL ROC curve is obviously increased; and our ENN-RL gained %8.39 improvement in AUC. If comparing Fig. 6.3 and Fig. 6.4, it is easy to see that the AUC of ADJ-RL decreases by 0.055 from 0.8112 in Fig. 6.3 to 0.7557 in Fig. 6.4. However, our ENN method performs stably with only 0.016 decrease in AUC. This suggests that traditional random walk methods usually need the training network to be connected; and the prediction performance largely depends on the size and density of the maximum connected component. However, when the training network becomes sparse and disconnected, the traditional random walk based methods will lose the predictive power because they cannot predict interactions among those disconnected components. We repeated the whole experiments up to ten times, the Table 6.3 shows the average performance with the standard deviation. All these results show our method performs stably and effectively in overcoming the limitation of traditional random walk based methods; and the improvements are statistically significant.

6.3.2 Experiments on Human PPI Data

We further tested our method by the human PPI data downloaded from HPRD (Release 9) [50], which is much larger and sparser than the yeast PPI network. Similarly, we carried out two comparisons by setting the α to be less than 0.25 and 0.125 respectively to divide G in to training network G_{tn} and testing set G_{tt}. The detailed information about the division can be found in the Table 6.4.

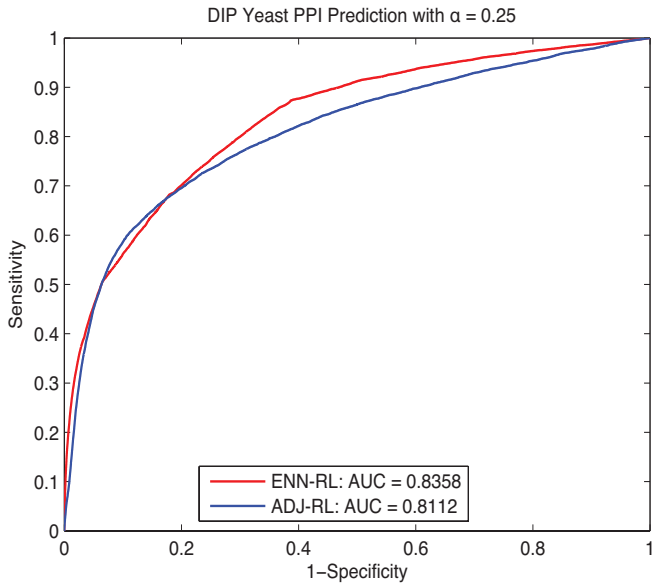


Figure 6.3: Yeast: ROC curves of predicting $G_{tt} \sim 18362$ with $\alpha = 0.25$.

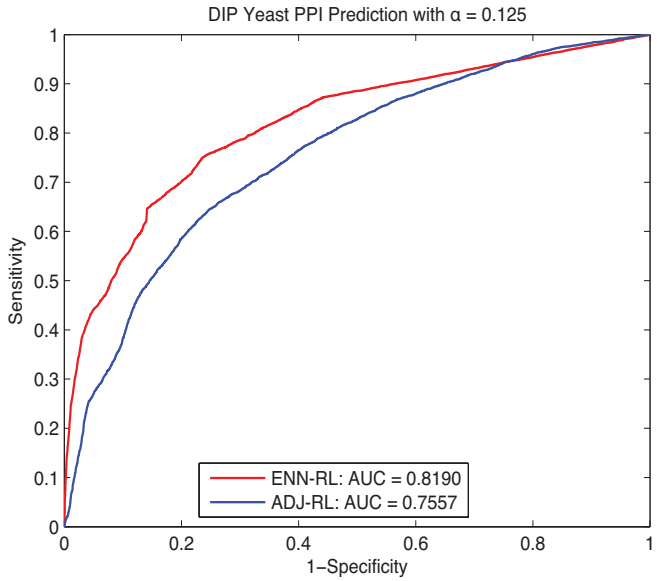


Figure 6.4: Yeast: ROC curves of predicting $G_{tt} \sim 20967$ with $\alpha = 0.125$.

The prediction performances in Fig. 6.5 and Fig. 6.6 show our ENN-RL has obviously better ROC curves and higher AUC than that of ADJ-RL. Especially for the test with $\alpha = 0.125$, our ENN-RL method obtains up to 14.9% improvement for predicting 34,779 testing interactions based on a training set G_{tn} with only 2,260

Table 6.4: Division of human golden standard PPI interactions

α	G_{tn}	$G_{tn}(\#C)$	$G_{tn}(minC, avgC, maxC)$	G_{tt}
0.25	6,567	5,370	(1, 1.79, 3,970)	30,472
0.125	2,260	7,667	(1, 1.25, 1,566)	34,779

$G_{tn}(\#C)$: the number of components in G_{tn}
$G_{tn}(minC, avgC, maxC)$: the minimum, average and maximum size of components in G_{tn}

interactions but 7,667 disconnected components. Similar tendency is also observed from Fig. 6.5 and Fig. 6.6. When α is decreased from 0.25 to 0.125, the AUC of ADJ-RL decreases by up to 0.072, while our ENN-RL only decreased by 0.021. We also did ten repetitions as shown in Table 6.5 to demonstrate the stable performance of the ENN-RL. All these results on human PPI data further indicate our ENN-RL model is a promising tool to predict edges for any sparse and disconnected training network.

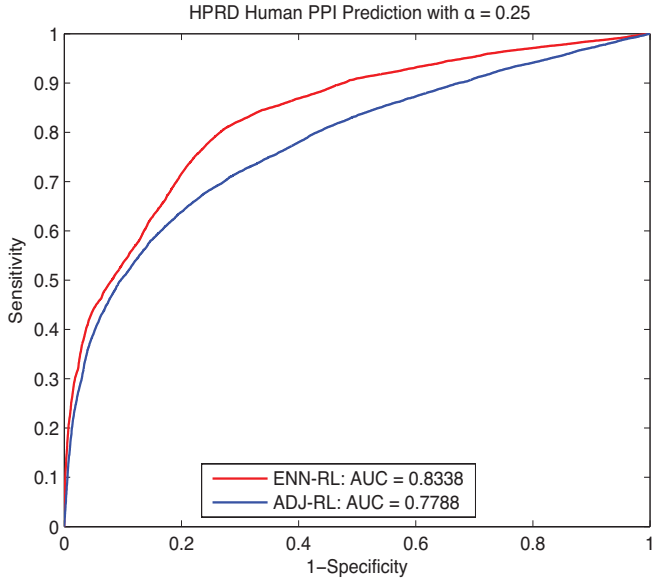


Figure 6.5: Human: ROC curves of predicting $G_{tt} \sim 30742$ with $\alpha = 0.25$.

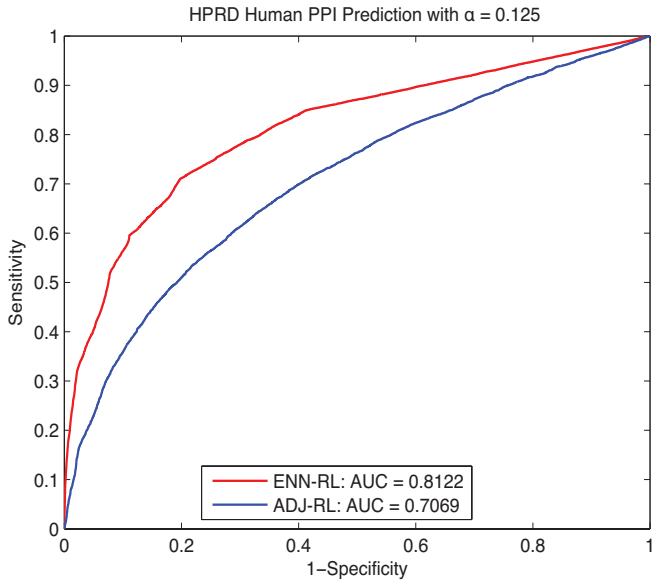


Figure 6.6: Human: ROC curves of predicting $G_{tt} \sim 34779$ with $\alpha = 0.125$.

Table 6.5: AUC summary of repetitions for human PPI data

Methods	Avg±Std ($\alpha = 0.25$)	Avg±Std ($\alpha = 0.125$)
ENN-RL	0.8320± 0.0012	0.8140± 0.0013
ADJ-RL	0.7795± 0.0047	0.6970± 0.0059

6.3.3 Optimize Weights for Heterogeneous Feature Kernels

Most recently, Huang et al. [39, 37] developed a method to infer *de novo* PPIs by applying regularized Laplacian kernel to a kernel fusion that based on optimally weighted heterogeneous feature kernels. To find the optimal weights, they proposed weight optimization by linear programming (WOLP) method that based on random walk over a connected training networks. Firstly, they utilized Barker algorithm and the training network to construct a transition matrix which constrains how a random walk would traverse the training network. Then the optimal kernel fusion can be obtained by adjusting the weights to minimize the element-wise difference between the transition matrix and the weighted kernels. The minimization problem is solved by linear programming.

Given a large disconnected network, although Huang et al. [39, 37] demonstrated that the weights learned from the maximum connected component can also be used to build kernel fusion for that large disconnected network, the weights will not be optimal when the maximum connected component is very small compared to the original disconnected network. As we all know that current available golden standard PPI networks are usually disconnected and remains far from complete. Therefore, it would be of great interest if we can obtain the transition matrix directly from these disconnected components, including but to limited to the maximum connected component, and use that transition matrix to help us find the optimal weights for heterogeneous feature kernels. To verify this idea, we use the transition matrix T obtained by Eq. (6.2) to find the optimal weights based on the linear programming Eq. (6.4).

$$W^* = \underset{W}{argmin} \, ||(W_0 G_{tn} + \sum_{i=1}^{n} W_i K_i) - T||^2 \tag{6.4}$$

We tested this method by the yeast PPI network with same setting in Table 6.2; and six feature kernels are included:

G_{tn}: G_{tn} is training network with $\alpha = 0.25$ or 0.125 in Table 6.2

$K_{Jaccard}$ [45]: It measures the similarity of protein pairs i, j in term of $\frac{neigbors(i) \cap neighbors(j)}{neighbors(i) \cup neighbors(j)}$.

K_{SN}: It measures the total number of neighbors of protein i and j, $K_{SN} = neighbors(i) + neighbors(j)$.

K_B [56]: It is a sequence-based kernel matrix that is generated using the BLAST [5].

K_E [56]: This is a gene co-expression kernel matrix constructed entirely from microarray gene expression measurements.

K_{Pfam} [56]: Similarity measure derived from Pfam HMMs [83].

All these kernels are normalized to the scale of $[0, 1]$ in order to avoid bias.

To make a comprehensive analysis, we also included prediction results based on the kernel fusion built by the approximate bayesian computation and modified differential evolution sampling (ABCDEP) method [38], and the kernel fusion built by equally weighted feature kernels EK for comparison. Similar to the comparison in [39, 37], the ABCDEP and EK based results can serve as the upper bound and

lower bound of the prediction performance. Comparisons for two settings $\alpha = 0.25$ and $\alpha = 0.125$ are shown by the Fig. 6.7 and Fig. 6.8 respectively. In the Fig. 6.7, the AUC of ENNlp-RL increase by up to 0.02 compared to ENN-RL, which is obvious better than other methods and very close to the upper bound 0.8529 obtained by ABCDEP-RL. Similarly, in Fig. 6.8, although the maximum component of G_{tn} has only 1,006 and total number of training interactions is just 1,456, the ENNlp-RL still get about 0.02 increase from the ENN-RL, which is also very close to the ABCDEP-RL. Therefore, all these results indicate that the transition matrix T learned by our ENN model can further improve the prediction performance by leveraging useful information from heterogeneous feature kernels.

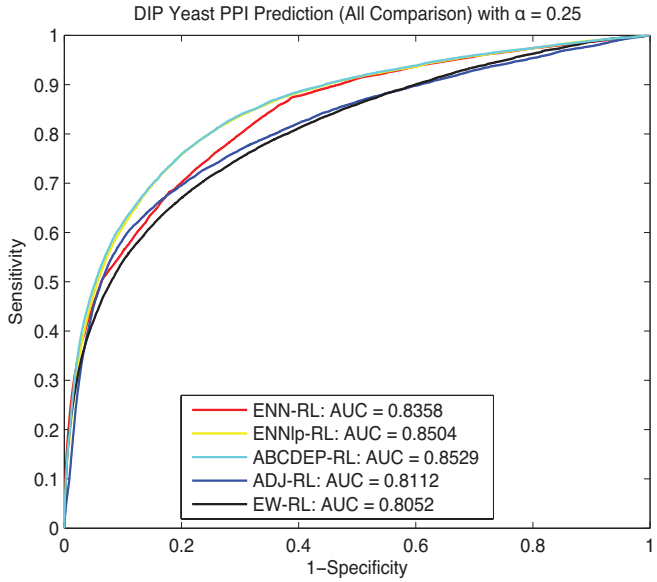


Figure 6.7: Yeast: ROC curves of predicting $G_{tt} \sim 18362$ with $\alpha = 0.25$.

6.4 Conclusions

In this chapter we developed a novel method based on deep learning neural network and regularized Laplacian kernel to predict interactions for sparse and disconnected PPI networks. We built the neural network with a typical auto-encoder structure to implicitly simulate the evolution processes of PPI networks. Based on

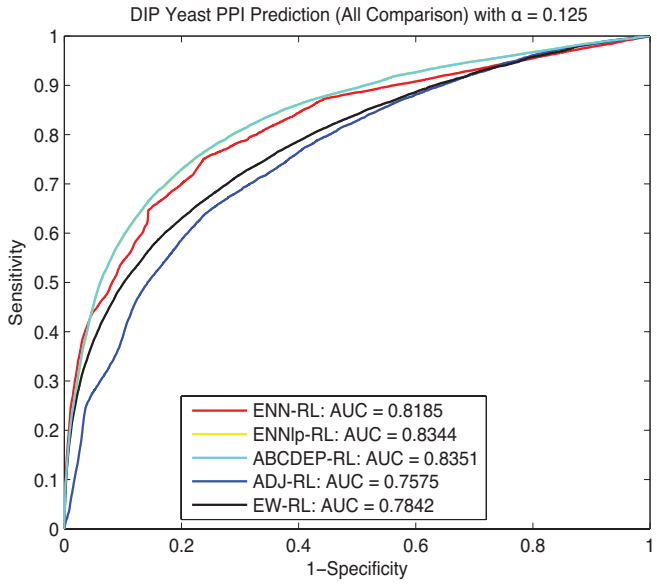


Figure 6.8: Yeast: ROC curves of predicting $G_{tt} \sim 20967$ with $\alpha = 0.125$.

the supervised learning using rows of assumed ancient network as inputs and rows of sparse and disconnected training network as labels, we can obtain an evolved PPI network as the outputs of the deep neural network. Then we predicted PPIs by applying regularized Laplacian kernel to the transition matrix built upon that evolved PPI network. Tested on DIP yeast PPI network and HPRD human PPI network, the results show that our method exhibits competitive advantages over the traditional regularized Laplacian kernel that based on the training network only. The proposed method achieved significant improvement in PPI prediction, as measured by ROC score, over 8.39% higher than the baseline for yeast data, and 14.9% for human data. Moreover, the transition matrix learned from our evolution neural network can also help us to build optimized kernel fusion, which effectively overcome the limitation of traditional WOLP method that needs a relatively large and connected training network to obtain the optimal weights. Then we also tested it by the DIP yeast data with six feature kernels, the prediction result shows the AUC can be further improved and very close to the upper bound. Given current golden standard PPI networks are usually disconnected and very sparse, we believe our model will be a promising tool that can

effectively utilize disconnected networks to predict PPIs.

Chapter 7

CONCLUSIONS

In this dissertation, after discussing the background, related works, and motivation of the PPI network study. We presented various algorithms/methods that we have developed in advancing the state-of-the-art of network level PPI analysis and prediction. In Chapter 2, we developed a novel method based on Approximate Bayesian Computation and modified Differential Evolution algorithm (ABC-DEP) that is capable of conducting evolution model selection and parameter estimation simultaneously and detecting the underlying evolutionary mechanisms for PPI networks more accurately. In Chapter 3, we developed a novel method that combines evolutionary models and geometric embedding to make prediction of de novo protein interactions in a PPI network. In addition, with this method, we can evaluate the fitness of given evolution models in terms of their prediction capacities. In Chapter 4 and 5, we proposed two PPI network inference methods that show obvious advantages over traditional network-level prediction methods in terms of features utilization, prediction for far-away interactions, and demand of golden standard training data. Chapter 4 is a sampling based method that can help us get the optimal kernel fusion with maximum prediction power; Chapter 5 is based on linear programming, which strikes a balance between the prediction power and computational efficiency. In Chapter 6, the method based on deep learning neural network and regularized Laplacian kernel can directly infer PPIs from disconnected sparse training networks, which are the typical situation in our current knowledge of interactome of various model organisms.

While these methods have been presented separately in the preceding chapters, they were designed to inter-operate with each other synergistically as part of a system. The Fig. 7.1 shows schematically how these parts fit together to produce a coherent

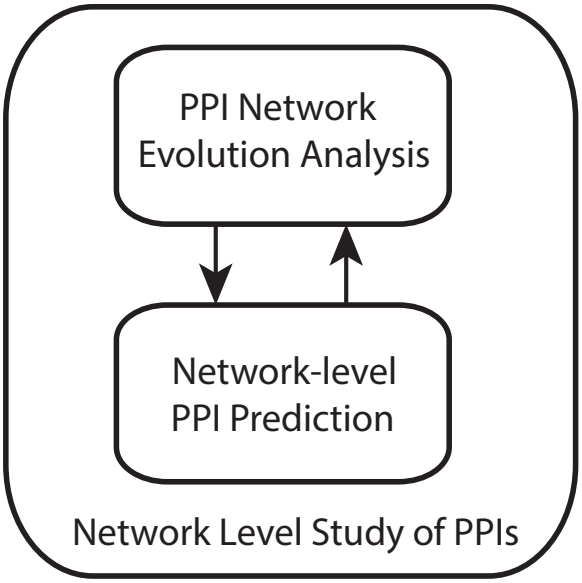


Figure 7.1: A system of our study.

system for PPI network study. As illustrated in Fig. 7.1, through network evolution analysis and PPI prediction/inference, more completed PPI networks will be obtained. And this process can continue iteratively for further improvement.

Although our methods are tested by PPI networks, we believe they can be used by researchers to analyze and study various complex networks, including, but not limited to, biological networks, social networks and transportation networks. Based on the methods in Chapter 2 and 3, by using existing or new evolution/growth models as inputs, researchers can get pie charts to show predominant models for given complex networks. It can not only predict potential connections for regional node pairs, but also capture the overall growth process for the given networks. For our inference methods in Chapter 4 and 5, they can effectively infer connections for sparse complex networks through leveraging complementary information from the sparse network and multiple heterogeneous feature kernels. Once the optimal kernel fusion is built, it can be reusable to predict connections for any partial training network. Meanwhile, when new feature kernels are available, researchers can also use our method to update the optimal weights and build new kernel fusion. When the given complex network is disconnected or feature kernels are unavailable, the method in Chapter 6 provides a

good solution to predict connections. We believe our methods will be very helpful to explore new applications or enhance current application performance for various complex networks, e.g., path analysis and module detection for biological networks, recommendation system for social networks and consumer networks, new line planning for transportation networks and so on.

For the future work, following derivative problems would be worthwhile to investigate and study.

Parallel implementation of sampling methods. For our ABCDEP sampling methods, we have used the CULAtools GPU linear algebra library perform the matrix calculations. Although the parallel architecture greatly improved the efficiency compared to a conventional CPU implementation, there is still a need to pursue faster parallel implementation to meet the computational demands as networks scale up.

Application exploration. Methods in this dissertation have been developed with a focus to effectively predict interactions for large, sparse and disconnected PPI networks. In future, more efforts could be made to exploring applications, such as dynamic interaction prediction, path analysis, module detection, etc.

Quick and user-friendly server. Currently, although source code of our methods can be accessible, it requires some programming skills, which is not very user-friendly, especially for researchers with no computer science background. Hence, it is important to develop quick and user-friendly tools for all public users.

BIBLIOGRAPHY

[1] Vázquez A., A. Flammini, A. Maritan, and A. Vespignani. Modeling of protein interaction networks. *Complexus*, 1(1):38–44, 2003.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] R. Albert. Scale-free networks in cell biology. *J Cell Sci*, 118(Pt 21):4947–57, 2005.

[4] Rka Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118(21):4947–4957, 2005.

[5] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

[6] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 635–644, New York, NY, USA, 2011. ACM.

[7] Jon Baker. An algorithm for the location of transition states. *Journal of Computational Chemistry*, 7(4):385–395, 1986.

[8] A. L. Barabási and Z. N. Oltvai. Network biology: understanding the cell's functional organization. *Nat Rev Genet*, 5(2):101–13, 2004.

[9] Albert-László Barabási. Scale-free networks: A decade and beyond. *Science*, 325(5939):412–413, 2009.

[10] Albert-Lszl Barabási and Rka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[11] Albert R Barabási A-L. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[12] Doron Betel, Ruth Isserlin, and Christopher W. V. Hogue. Analysis of domain correlations in yeast protein complexes. *Bioinformatics*, 20(suppl 1):i55–i62, 2004.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Secaucus, 2006.

[14] Arianna Bottinelli, Bruno Bassetti, Marco Cosentino Lagomarsino, and Marco Gherardi. Influence of homology and node age on the growth of protein-protein interaction networks. *Phys. Rev. E*, 86:041919, Oct 2012.

[15] K. S. Brown, C. C. Hill, G. A. Calero, C. R. Myers, K. H. Lee, J. P. Sethna, and R. A. Cerione. The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Phys Biol*, 1(3-4):184–95, 2004.

[16] Carlo Vittorio Cannistraci, Gregorio Alanis-Lobato, and Timothy Ravasi. Minimum curvilinearity to enhance topological prediction of protein interactions by network embedding. *Bioinformatics*, 29(13):i199–i209, 2013.

[17] Carlo Vittorio Cannistraci, Timothy Ravasi, Franco Maria Montevecchi, Trey Ideker, and Massimo Alessio. Nonlinear dimension reduction and clustering by minimum curvilinearity unfold neuropathic pain and tissue embryological classes. *Bioinformatics*, 26(18):i531–i539, 2010.

[18] Hung-Hsuan Chen, Liang Gou, Xiaolong (Luke) Zhang, and C. Lee Giles. Discovering missing links in networks using vertex similarity measures. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 138–143, New York, NY, USA, 2012. ACM.

[19] Young-Rae Cho, Marco Mina, Yanxin Lu, Nayoung Kwon, and Pietro Guzzi. M-finder: Uncovering functionally associated proteins from interactome data integrated with go annotations. *Proteome Science*, 11(Suppl 1):S3, 2013.

[20] Hinrich Sch utze Christopher D. Manning, Prabhakar Raghavan. *Introduction to Information Retrieval*. Cambridge University Press, New York, USA, 2008.

[21] Roger Craig and Li Liao. Phylogenetic tree information aids supervised learning for predicting protein-protein interaction based on distance matrices. *BMC Bioinformatics*, 8(1):6, 2007.

[22] K. Csillery, M. G. Blum, O. E. Gaggiotti, and O. Francois. Approximate bayesian computation (abc) in practice. *Trends Ecol Evol*, 25(7):410–8, 2010.

[23] Minghua Deng, Shipra Mehta, Fengzhu Sun, and Ting Chen. Inferring domain-domain interactions from protein-protein interactions. *Genome Research*, 12(10):1540–1548, 2002.

[24] Yue Deng, Lin Gao, and Bingbo Wang. ppipre: predicting protein-protein interactions by combining heterogeneous features. *BMC Systems Biology*, 7(Suppl 2):S8, 2013.

[25] S.N. Dorogovtsev and J.F.F. Mendes. Minimal models of weighted scale-free networks, 2004.

[26] Frank Emmert-Streib. Limitations of gene duplication models: Evolution of modules in protein interaction networks. *PLoS ONE*, 7(4):e35531, 04 2012.

[27] David Chin-Lung Fong and Michael Saunders. Lsmr: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

[28] Franois Fouss, Kevin Francoisse, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural Networks*, 31(0):53 – 72, 2012.

[29] Alvaro Gonzalez and Li Liao. Predicting domain-domain interaction based on domain profiles with feature selection and support vector machines. *BMC Bioinformatics*, 11(1):537, 2010.

[30] J. D. Han, N. Bertin, T. Hao, D. S. Goldberg, G. F. Berriz, L. V. Zhang, D. Dupuy, A. J. Walhout, M. E. Cusick, F. P. Roth, and M. Vidal. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995):88–93, 2004.

[31] T. Hase, H. Tanaka, Y. Suzuki, S. Nakagawa, and H. Kitano. Structure of protein interaction networks and their implications on drug design. *Plos Computational Biology*, 5(10), 2009.

[32] Takeshi Hase, Hiroshi Tanaka, Yasuhiro Suzuki, So Nakagawa, and Hiroaki Kitano. Structure of protein interaction networks and their implications on drug design. *PLoS Comput Biol*, 5(10):e1000550, 10 2009.

[33] D. J. Higham, M. Rasajski, and N. Przulj. Fitting a geometric graph to a protein-protein interaction network. *Bioinformatics*, 24(8):10931099, Apr 2008.

[34] Geoffrey Hinton and Yoshua Bengio. Visualizing data using t-sne. In *Cost-sensitive Machine Learning for Information Retrieval 33*, pages 2579–2605. MIT Press, 2008.

[35] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT Press, 2003.

[36] F. Hormozdiari, P. Berenbrink, N. Przulj, and S. C. Sahinalp. Not all scale-free networks are born equal: the role of the seed graph in ppi network evolution. *PLoS Comput Biol*, 3(7):e118, 2007.

[37] Lei Huang, Li Liao, and Cathy H. Wu. Protein-protein interaction network inference from multiple kernels with optimization based on random walk by linear programming. In *Proceedings of 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 201–207, Washington DC, USA, 2015. IEEE computer society.

[38] Lei Huang, Li Liao, and Cathy H. Wu. Inference of protein-protein interaction networks from multiple heterogeneous data. *EURASIP Journal on Bioinformatics and Systems Biology*, 2016(1):1–9, 2016.

[39] Lei Huang, Li Liao, and Cathy H. Wu. Protein-protein interaction prediction based on multiple kernels and partial network with linear programming. *BMC Systems Biology*, 10(2):45, 2016.

[40] Lei Huang, Li Liao, and C.H. Wu. Evolutionary model selection and parameter estimation for protein-protein interaction network based on differential evolution algorithm. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 12(3):622–631, May 2015.

[41] I. Ispolatov, P. Krapivsky, I. Mazo, and A. Yuryev. Cliques and duplication-divergence network growth. *New J Phys*, 7:145, 2005.

[42] I. Ispolatov, P. L. Krapivsky, and A. Yuryev. Duplication-divergence model of protein interaction network. *Phys Rev E Stat Nonlin Soft Matter Phys*, 71(6 Pt 1):061911, 2005.

[43] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. Application of kernels to link analysis. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 586–592, New York, NY, USA, 2005. ACM.

[44] Zohar Itzhaki, Eyal Akiva, Yael Altuvia, and Hanah Margalit. Evolutionary conservation of domain-domain interactions. *Genome Biology*, 7(12):R125, 2006.

[45] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[46] H. Jeong, S. P. Mason, A. L. Barabasi, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–2, 2001.

[47] Y. Jin, D. Turaev, T. Weinmaier, T. Rattei, and H. A. Makse. The evolutionary dynamics of protein-protein interaction networks inferred from the reconstruction of ancient networks. *PLoS One*, 8(3):e58134, 2013.

[48] M. P. Joy, A. Brock, D. E. Ingber, and S. Huang. High-betweenness proteins in the yeast protein interaction network. *J Biomed Biotechnol*, 2005(2):96–103, 2005.

[49] Suk-Hoon Jung, Woo-Hyuk Jang, and Dong-Soo Han. A computational model for predicting protein interactions based on multidomain collaboration. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1081–1090, 2012.

[50] T. S. Keshava Prasad, Renu Goel, Kumaran Kandasamy, Shivakumar Keerthikumar, Sameer Kumar, Suresh Mathivanan, Deepthi Telikicherla, Rajesh Raju, Beema Shafreen, Abhilash Venugopal, Lavanya Balakrishnan, Arivusudar Marimuthu, Sutopa Banerjee, Devi S. Somanathan, Aimy Sebastian, Sandhya Rani, Somak Ray, C. J. Harrys Kishore, Sashi Kanth, Mukhtar Ahmed, Manoj K. Kashyap, Riaz Mohmood, Y. L. Ramachandra, V. Krishna, B. Abdul Rahiman, Sujatha Mohan, Prathibha Ranganathan, Subhashri Ramabadran, Raghothama Chaerkady, and Akhilesh Pandey. Human protein reference database-2009 update. *Nucleic Acids Research*, 37(suppl 1):D767–D772, 2009.

[51] R. Khanin and E. Wit. How scale-free are biological networks. *Journal of Computational Biology*, 13(3):810–818, 2006.

[52] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[53] Oleksii Kuchaiev, Marija Rašajski, Desmond J. Higham, and Nataša Pržulj. Geometric de-noising of protein-protein interaction networks. *PLoS Comput Biol*, 5(8):e1000454, 08 2009.

[54] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 57–, Washington, DC, USA, 2000. IEEE Computer Society.

[55] Jérôme Kunegis, Damien Fay, and Christian Bauckhage. Network growth and the spectral evolution model. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 739–748, New York, NY, USA, 2010. ACM.

[56] Gert R. G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[57] Chengwei Lei and Jianhua Ruan. A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity. *Bioinformatics*, 2012.

[58] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. Link prediction: the power of maximal entropy random walk. In *CIKM'11*, pages 1147–1156, 2011.

[59] Linyuan L and Tao Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):11501170, 2011.

[60] Amin Mantrach, Nicolas van Zeebroeck, Pascal Francq, Masashi Shimbo, Hugues Bersini, and Marco Saerens. Semi-supervised classification and betweenness computation on large, sparse, directed graphs. *Pattern Recognition*, 44(6):1212 – 1224, 2011.

[61] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–3, 2002.

[62] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*, ECML PKDD'11, pages 437–452, Berlin, Heidelberg, 2011. Springer-Verlag.

[63] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[64] Christopher C. Paige and Michael A. Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, March 1982.

[65] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 653–658, New York, NY, USA, 2004. ACM.

[66] Jong Park, Michael Lappe, and Sarah A Teichmann. Mapping protein family interactions: intramolecular and intermolecular protein family interaction repertoires in the {PDB} and yeast1. *Journal of Molecular Biology*, 307(3):929 – 938, 2001.

[67] Ashwini Patil, Kengo Kinoshita, and Haruki Nakamura. Domain distribution and intrinsic disorder in hubs in the human proteinprotein interaction network. *Protein Science*, 19(8):1461–1468, 2010.

[68] Rob Patro, Emre Sefer, Justin Malin, Guillaume Marcais, Saket Navlakha, and Carl Kingsford. Parsimonious reconstruction of network evolution. *Algorithms for Molecular Biology*, 7(1):25, 2012.

[69] Robert Patro, Geet Duggal, Emre Sefer, Hao Wang, Darya Filippova, and Carl Kingsford. The missing models: A data-driven approach for learning how networks grow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 42–50, New York, NY, USA, 2012. ACM.

[70] Piotr Pawlowski, Szymon Kaczanowski, and Piotr Zielenkiewicz. A kinetic model of the evolution of a protein interaction network. *BMC Genomics*, 14(1):172, 2013.

[71] G. Jack Peterson, Steve Pressé, Kristin S. Peterson, and Ken A. Dill. Simulated evolution of protein-protein interaction networks with realistic topology. *PLoS ONE*, 7(6):e39052, 06 2012.

[72] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.

[73] Nataa Prulj. Protein-protein interactions: Making sense of networks via graph-theoretic modeling. *BioEssays*, 33(2):115–123, 2011.

[74] Storn Rainer and Price Kenneth. Differential evolution for continuous function optimization (an algorithm by kenneth price and rainer storn). http://www1.icsi.berkeley.edu/~storn/code.html. Accessed: 2013-12-20.

[75] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–5, 2002.

[76] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[77] Yook S-H., Oltvai ZN., and Barabsi A-L. Functional and topological characterization of protein interaction networks. *PROTEOMICS*, 4(4):928–942, 2004.

[78] Lukasz Salwinski and David Eisenberg. Computational methods of analysis of proteinprotein interactions. *Current Opinion in Structural Biology*, 13(3):377 – 382, 2003.

[79] Lukasz Salwinski, Christopher S. Miller, Adam J. Smith, Frank K. Pettit, James U. Bowie, and David Eisenberg. The database of interacting proteins: 2004 update. *Nucleic Acids Research*, 32(90001):449451, Jan 2004.

[80] Rohit Singh, Daniel Park, Jinbo Xu, Raghavendra Hosur, and Bonnie Berger. Struct2net: a web service to predict proteinprotein interactions using a structure-based approach. *Nucleic Acids Research*, 38(suppl 2):W508–W515, 2010.

[81] AlexanderJ. Smola and Risi Kondor. Kernels and regularization on graphs. In Bernhard Schlkopf and ManfredK. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 144–158. Springer Berlin Heidelberg, 2003.

[82] Nan Song, Jacob M. Joseph, George B. Davis, and Dannie Durand. Sequence similarity network reveals common ancestry of multidomain proteins. *PLoS Comput Biol*, 4(5):1–19, 05 2008.

[83] Erik L.L. Sonnhammer, Sean R. Eddy, and Richard Durbin. Pfam: A comprehensive database of protein domain families based on seed alignments. *Proteins: Structure, Function, and Bioinformatics*, 28(3):405–420, 1997.

[84] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, 34(suppl 1):D535–D539, 2006.

[85] Jingchun Sun, Yan Sun, Guohui Ding, Qi Liu, Chuan Wang, Youyu He, Tieliu Shi, Yixue Li, and Zhongming Zhao. Inpreppi: an integrated evaluation method based on genomic context for predicting protein-protein interactions in prokaryotic genomes. *BMC Bioinformatics*, 8(1):414, 2007.

[86] Panagiotis Symeonidis, Nantia Iakovidou, Nikolaos Mantas, and Yannis Manolopoulos. From biological to social networks: Link prediction based on multi-way spectral clustering. *Data & Knowledge Engineering*, 87(0):226 – 242, 2013.

[87] R. Tanaka, T. M. Yi, and J. Doyle. Some protein interaction data do not exhibit power law statistics. *Febs Letters*, 579(23):5140–5144, 2005.

[88] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[89] T. Thorne and M. P. Stumpf. Graph spectral analysis of protein interaction network evolution. *J R Soc Interface*, 9(75):2653–66, 2012.

[90] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Random walk with restart: fast solutions and applications. *Knowledge and Information Systems*, 14(3):327–346, 2008.

[91] T. Toni and M. P. Stumpf. Simulation-based model selection for dynamical systems in systems and population biology. *Bioinformatics*, 26(1):104–10, 2010.

[92] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[93] Michael A. A. Cox Trevor F. Cox. *Multidimensional Scaling*. London: Chapman and Hall, London, 1994.

[94] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(5):695–703, 1988.

[95] X. Wan, S. Cai, J. Zhou, and Z. Liu. modularity and disassortativity in protein-protein interaction networks. *Chaos*, 20(4):045113, 2010.

[96] Hua Wang, Heng Huang, Chris Ding, and Feiping Nie. Predicting protein–protein interactions from multimodal biological data sources via nonnegative matrix tri-factorization. *Journal of Computational Biology*, 20(4):344–358, 2015/04/27 2013.

[97] Richard C. Wilson and Ping Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9):2833–2841, 2008.

[98] Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20(suppl 1):i363–i370, 2004.

[99] Zhu-Hong You, Ying-Ke Lei, Jie Gui, De-Shuang Huang, and Xiaobo Zhou. Using manifold embedding for assessing and predicting protein interactions from high-throughput experimental data. *Bioinformatics*, 26(21):2744–2751, 2010.

[100] Aidong Zhang. *Protein Interaction Networks: Computational Analysis*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[101] Q. C. Zhang, D. Petrey, J. I. Garzon, L. Deng, and B. Honig. Preppi: a structure-informed database of protein-protein interactions. *Nucleic Acids Res*, 41(Database issue):D828–33, 2013.

[102] Qiangfeng Cliff Zhang, Donald Petrey, Lei Deng, Li Qiang, Yu Shi, Chan Aye Thu, Brygida Bisikirska, Celine Lefebvre, Domenico Accili, Tony Hunter, Tom Maniatis, Andrea Califano, and Barry Honig. Structure-based prediction of protein-protein interactions on a genome-wide scale. *Nature*, 490(7421):556–560, 10 2012.

[103] Lin Zhu, Zhu-Hong You, De-Shuang Huang, and Bing Wang. t-LSe: A novel robust geometric approach for modeling protein-protein interaction networks. *PLoS ONE*, 8(4):e58368, Apr 2013.

Appendix A

REPRINT PERMISSION LETTERS

BMC Systems Biology

Copyright

- Copyright on any open access article in a journal published by BioMed Central is retained by the author(s).
- Authors grant BioMed Central a license to publish the article and identify itself as the original publisher.
- Authors also grant any third party the right to use the article freely as long as its integrity is maintained and its original authors, citation details and publisher are identified.
- The Creative Commons Attribution License 4.0 formalizes these and other terms and conditions of publishing articles.

In addition to BioMed Central's copyright policy, some journals also follow an Open Data policy and the Creative Commons CC0 1.0 Public Domain Dedication waiver applies to all published data in these journals. Further information can be found on the individual journals pages.

Where an author is prevented from being the copyright holder (for instance in the case of US government employees or those of Commonwealth governments), minor variations may be required. In such cases the copyright line and license statement in individual articles will be adjusted, for example to state '© 2016 Crown copyright'. Authors requiring a variation of this type should inform BioMed Central during or immediately after submission of their article. Changes to the copyright line cannot be made after publication of an article.

Exceptions to copyright policy
Our policy pages provide details concerning copyright and licensing for articles which were previously published under policies that are different from the above. For instance, occasionally BioMed Central may co−publish articles jointly with other publishers, and different licensing conditions may then apply. In all such cases, however, access to these articles is free from fees or any other access restrictions.

Information specifically regarding permissions and reprints can be found here. Please contact us if there are questions.

BMC Systems Biology

ISSN: 1752-0509

Contact us

- Editorial email: bmcsystbiol@biomedcentral.com
- Support email: info@biomedcentral.com

Copyright

- Copyright on any open access article in a journal published by SpringerOpen is retained by the author(s).
- Authors grant SpringerOpen a license to publish the article and identify itself as the original publisher.
- Authors also grant any third party the right to use the article freely as long as its integrity is maintained and its original authors, citation details and publisher are identified.
- The Creative Commons Attribution License 4.0 formalizes these and other terms and conditions of publishing articles.

In addition to SpringerOpen's copyright policy, some journals also follow an Open Data policy and the Creative Commons CC0 1.0 Public Domain Dedication waiver applies to all published data in these journals. Further information can be found on the individual journals pages and in the license agreement.

Where an author is prevented from being the copyright holder (for instance in the case of US government employees or those of Commonwealth governments), minor variations may be required. In such cases the copyright line and license statement in individual articles will be adjusted, for example to state '© 2016 Crown copyright'. Authors requiring a variation of this type should inform SpringerOpen during or immediately after submission of their article. Changes to the copyright line cannot be made after publication of an article.

Exceptions to copyright policy
There may be exceptions concerning copyright and licensing for articles which were previously published under policies that are different from the above. For instance, occasionally SpringerOpen may co−publish articles jointly with other publishers, and different licensing conditions may then apply. In all such cases, however, access to these articles is free from fees or any other access restrictions.

Information specifically regarding permissions and reprints can be found here. Please contact us if there are questions.

- ISSN: 1687-4153