

Abstract

Computational Reconstruction of Biological Networks

by

Yuk Lap YIP

2009

Networks describe the interactions between different objects. In living systems, knowing which biological objects interact with each other would deepen our understanding of the functions of both individual objects and their working modules. Due to experimental limitations, currently only small portions of these interaction networks are known. This thesis describes methods for computationally inferring the complete networks based on the known portions and related data. These methods exploit special data properties and problem structures to achieve high accuracy. The training set expansion method handles sparse and uneven training data by learning from information-rich regions of the network, and propagating the information to help learn from the information-poor regions. The multi-level learning framework combines information at different levels of a concept hierarchy, and lets the predictors at the different levels to propagate information between each other. Combined optimization between levels allow the integrated use of data features at different levels to improve prediction accuracy and noise immunity. Finally, proper incorporation of heterogeneous data facilitates the identification of interactions uniquely detectable by each kind of data. This thesis also describes some work on data integration and tool sharing, which are crucial components of network analysis studies.

# Computational Reconstruction of Biological Networks

A Dissertation

Presented to the Faculty of the Graduate School

of

Yale University

in Candidacy for the Degree of

Doctor of Philosophy

*by*

Yuk Lap YIP

Dissertation Directors:

Mark Gerstein and Drew McDermott

May 2009

© 2009 by Yuk-Lap Yip. All rights reserved.

# Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this university or to any other institution for a degree, diploma or other qualifications.

.....

Yuk Lap YIP

May 2009



# Acknowledgments

I am in great debt to many who have supported and encouraged me during my past five years in New Haven. Without them, it would not have been possible for me to reach this final stage of doctoral study.

First of all, I thank God for His numerous blessings. I thank my family, Dad, Mom, and Joanna, for their generosity to let me study abroad, and for their continuous support in all aspects. I thank my beloved Iris for traveling this long journey with me.

I would like to express my sincere gratitude to my adviser Mark Gerstein, who always does everything he could in helping my study. He has devoted an uncountable number of hours to detailed discussions about my research, study, and many other interesting topics.

I thank my adviser Drew McDermott for his great efforts in helping me progress through the many stages of my study.

I thank my thesis committee members Martin Schultz and Hongyu Zhao for their advice on my work.

I have received a lot of support from my mentor Kei Cheung, from the help during the admission process, the financial support in my first year of study, his guidance in the

various research projects, to the countless warm greetings and care. I thank him for all his kindness.

I would like to thank my fellow colleagues in the Gerstein lab, especially Roger Alexander, Jiang Du, Tara Gianoulis, Philip Kim, Alberto Paccanaro, Prianka Patel, Haiyuan Yu, and everyone in the YeastHub/Y6K/Networks/Meta groups. I have enjoyed very much working with people in the lab with diverse backgrounds and characters, and have learned a lot from each and every one of you, academically and non-academically.

I thank my collaborators at Yale Center of Medical Informatics and in Hong Kong, who have fueled my research and provided me a lot of insights.

For the various research projects, I appreciate the suggestions from and discussions with Kevin Bleakley, Carole Goble, Mudassar Iqbal, Daniel Marbach, Rama Ranganathan, William Russ, Gustavo Stolovitzky, Jean-Philippe Vert and Jiri Vohradsky. I would also like to thank the Yale University Biomedical High Performance Computing Center, especially Robert Bjornson and Nicholas Carriero, the DREAM organizing committee, and the anonymous reviewers of my research articles. I would like to acknowledge the AL Williams Professorship funds, the grants from NIH, NINDS, NLM and NSF that have directly and indirectly supported my research.

Last but not least, I would like to thank my many friends in the Calvary Baptist Church, the Yale Cantonese community, the Computer Science department, and other groups and individuals, who have greatly enriched my life throughout these years.

# Contents

<b>Declaration</b>	<b>i</b>
<b>Acknowledgments</b>	<b>iii</b>
<b>Contents</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Biological Background</b>	<b>8</b>
2.1 DNA, RNA and proteins . . . . .	8
2.2 Biological networks . . . . .	12
<b>I Reconstructing Biological Networks</b>	<b>15</b>
<b>3 Exploiting Data Properties: Training Set Expansion</b>	<b>16</b>
3.1 Introduction . . . . .	16



3.2	Related work: existing approaches for network reconstruction . . . . .	20
3.2.1	The pairwise kernel approach . . . . .	20
3.2.2	The direct approach . . . . .	20
3.2.3	The matrix completion approach . . . . .	22
3.2.4	The local modeling approach . . . . .	22
3.3	Our proposal: the training set expansion approach . . . . .	24
3.3.1	Prediction propagation (pp) . . . . .	25
3.3.2	Kernel initialization (ki) . . . . .	27
3.3.3	Combining the two methods (pp+ki) . . . . .	28
3.4	Prediction accuracy . . . . .	29
3.4.1	Data and setup . . . . .	29
3.4.2	Results . . . . .	33
3.5	Analysis . . . . .	40
3.6	Discussion . . . . .	43
<b>4</b>	<b>Utilizing Problem Structures: Multi-level Learning</b>	<b>44</b>
4.1	Introduction . . . . .	44
4.2	Related work . . . . .	47
4.3	Problem definition . . . . .	49

4.4	Methods . . . . .	50
4.4.1	Information flow architectures . . . . .	51
4.4.2	Different approaches to coupling the levels . . . . .	52
4.4.3	Global vs. local modeling, and data sparsity issues . . . . .	55
4.4.4	The concrete algorithm . . . . .	58
4.5	Experiments . . . . .	62
4.5.1	Data . . . . .	62
4.5.2	Evaluation procedure . . . . .	65
4.5.3	Results . . . . .	66
4.6	Discussion . . . . .	69
<b>5</b>	<b>Handling Errors in Data: Consistent Prediction of Interactions at Different Levels</b>	<b>71</b>
5.1	Introduction . . . . .	71
5.2	How is DDI inference affected by noise in PPI network? . . . . .	74
5.3	Handling errors in the PPI network . . . . .	78
5.3.1	The original EM algorithm . . . . .	78
5.3.2	Incorporating protein features . . . . .	83
5.3.3	Empirical study . . . . .	86
5.3.4	Constrained likelihood estimation . . . . .	88

5.4	Discussion . . . . .	93
<b>6</b>	<b>Adding New Perspectives to Existing Problems: Discovering New Information in New Data</b>	<b>95</b>
6.1	Introduction . . . . .	95
6.2	Problem definition . . . . .	98
6.3	The learning method . . . . .	99
6.3.1	Learning noise models from deletion data . . . . .	99
6.3.2	Learning differential equation models from perturbation time series data . . . . .	102
6.3.3	Combining the predictions of the models . . . . .	105
6.4	Performance study . . . . .	107
6.4.1	Datasets and performance metrics . . . . .	107
6.4.2	Results . . . . .	108
6.5	Discussion and future directions . . . . .	113
<b>II</b>	<b>Data Integration</b>	<b>115</b>
<b>7</b>	<b>Semantic Web</b>	<b>116</b>
7.1	Introduction . . . . .	116
7.2	RDF data warehouse . . . . .	119

7.2.1	RDF data stores . . . . .	121
7.2.2	Metadata and data . . . . .	122
7.3	Biological use case: YeastHub . . . . .	129
7.3.1	Example queries . . . . .	131
7.3.2	Performance . . . . .	135
7.3.3	Implementation . . . . .	138
7.4	Discussion . . . . .	138
<b>8</b>	<b>Web 2.0</b>	<b>142</b>
8.1	Introduction . . . . .	142
8.2	Mashup scenarios . . . . .	146
8.2.1	Life sciences scenario: annotating microarray data . . . . .	147
8.2.2	Public health scenario 1: correlating cancer and environmental factors	154
8.2.3	Public health scenario 2: predicting West Nile Virus cases . . . . .	156
8.3	Strengths and weaknesses . . . . .	161
8.4	HCLS 3.0 . . . . .	167
8.5	HCLS 2.0 + HCLS 3.0 = e-HCLS . . . . .	172

<b>III</b>	<b>Network Tools Developed for the Community</b>	<b>175</b>
<b>9</b>	<b>tYNA: the Yale Network Analyzer</b>	<b>176</b>
9.1	Introduction . . . . .	176
9.2	Using tYNA . . . . .	179
9.2.1	Uploading networks and categories . . . . .	179
9.2.2	Loading networks into workspaces . . . . .	180
9.2.3	Single-network operations (advanced view) . . . . .	181
9.2.4	Multiple-network operations (advanced view) . . . . .	183
9.2.5	Mining and edge overlap (advanced view) . . . . .	185
9.2.6	Saving and downloading analyzed networks . . . . .	186
9.3	Implementation . . . . .	186
9.4	Discussion . . . . .	186
<b>10</b>	<b>The Coevolution Server</b>	<b>189</b>
10.1	Introduction . . . . .	189
10.2	Scoring Functions . . . . .	190
10.2.1	Correlation-based functions . . . . .	190
10.2.2	Perturbation-based functions . . . . .	192
10.2.3	Independence tests . . . . .	193

10.3	Preprocessing options . . . . .	193
10.3.1	Sequence filtering and weighting . . . . .	193
10.3.2	Site filtering . . . . .	194
10.3.3	Site pair filtering . . . . .	194
10.3.4	Other options . . . . .	194
10.4	Scores analysis . . . . .	195
10.5	Example . . . . .	195
10.6	Discussion . . . . .	196
10.7	Appendix: our implementation of the SCA method . . . . .	197
10.7.1	Introduction . . . . .	197
10.7.2	Design choices . . . . .	198
<b>11</b>	<b>Conclusion</b>	<b>202</b>
	<b>Bibliography</b>	<b>206</b>

# Chapter 1

## Introduction

The boat is not the material it is made from, but something else, much more interesting, which organises the material of the planks: the boat is the relationship between the planks. Similarly, the study of life should never be restricted to objects, but must look into their relationships.

– Antoine Danchin, “The Delphic Boat” [45]

In computer science, graphs are used to represent object relationships, with each node representing an object and each edge between two nodes stating that the two corresponding objects have a certain relationship. Graphs are also called networks in some domains. For example, in a computer network, each node is a machine and there is an edge between two nodes if the machines are physically or logically connected. In a social network of friends, each node is a person and there is an edge between two nodes if the two persons know each other. In some contexts, the term “network” specifically means a graph with weighted edges [24]. We shall not make such a distinction here, and shall treat “graph” and

“network” as synonyms.

This thesis is about the reconstruction of biological networks by computational means. In these networks, each node is a biological object, and an edge represents a specific type of interaction between two biological objects. For example, in a protein interaction network, each node is a protein, and there is an edge between two nodes if the corresponding proteins have a physical interaction. In a gene regulatory network, each node is a gene and its proteins, and there is a directed edge from a node to another if the former regulates the transcription of the latter. There are many other types of interesting biological networks, such as metabolic networks, genetic interaction networks and co-evolution networks. A brief introduction of the underlying biological concepts, as well as other concepts useful for understanding the content of this thesis, is given in Chapter 2.

Knowing the interconnections between the objects in these networks is an important first step to the greater goal of understanding the complex dynamics inside the biological systems [106]. For example, the knowledge of what interaction partners a protein has can help identify its function [186]. Analyzing the whole interaction network can provide insights into the structures and mechanisms of physical binding, which cannot be obtained by studying single objects alone [105]. Large-scale genetic and gene-drug interaction networks are also useful in drug discovery [141].

While it would be ideal to have full access to the biological networks, currently only small portions of them have been revealed experimentally [90, 177]. On the other hand, in the past decade many high-throughput experimental techniques have been developed and popularized to provide different kinds of information about the biological objects, most notably gene expression measured by microarray [162] and sequence information by second-generation sequencing [170]. The huge amount of data generated from these experiments



contain very rich information that can be utilized in predicting the unobserved portions of the biological networks. As such, computational reconstruction of biological networks has become an important research topic in bioinformatics [36, 171, 196, 199].

Network reconstruction can be formally cast as a machine-learning problem of the following general form. The inputs to the problem are:

- A number of objects, each described by a vector of feature values. Some additional features may be available for pairs of objects, such as the likelihood for a pair of objects to interact according to some physical experiments that are not totally reliable.
- A gold standard positive set of known interactions.
- A gold standard negative set of known non-interactions.

The goal is to learn a predictor from the inputs, so that when presented any two arbitrary objects  $i$  and  $j$ , it will predict the chance that  $(i, j)$  is an edge of the network.

Since the network reconstruction problem fits in a standard machine learning setting, one could tackle it by applying an existing learning algorithm. Indeed, there have been studies that use support vector machines [17, 174], Bayesian approaches [73, 98] and other standard machine learning methods [13, 59] to reconstruct biological networks.

While standard machine learning methods could make accurate predictions in some cases, we claim that if some domain knowledge about the problem structure and data properties is available, it is possible to design learning algorithms that make good use of the knowledge to achieve higher prediction accuracy. In the first part of this thesis, we demonstrate how this abstract idea is turned into practice in several studies.

In Chapter 3, we study the problem of supervised learning of protein interaction networks. We discuss several major difficulties in this problem, namely the large number of node pairs (18 million for the 6,000 nodes of yeast), the small number of known interactions and non-interactions, and the uneven distribution of these gold-standard examples across the different nodes, and the existence of sub-class structures. We tackle the problem by building local models with training set expansion [203], which consists of semi-supervised methods [34] that augment the original training sets by propagating information from information-rich regions of the training network to information-poor regions. We show that the resulting algorithms outperform a series of state-of-the-art algorithms when tested on multiple benchmark datasets.

In Chapter 4, we continue to explore the idea of training set expansion for the protein interaction network. In addition to making horizontal expansion (generating more training examples for other nodes), in this case the expansion is also vertical (generating more training examples for nodes at other levels). This idea is inspired by a special hierarchical structure of the protein interaction network: each protein interaction involves corresponding domain interactions, which in turn involve residue interactions. Each of the three levels of interactions contains unique data features for learning the corresponding network. We show that by considering all three levels of network reconstruction together, it is possible to improve the prediction accuracy at each level [204].

In Chapter 5, we focus on the protein and domain levels, and study how inference at the domain level could be affected by the errors at the protein level, which is a real concern given the high false positive and false negative rates of protein interaction networks constructed from high-throughput experiments, which are commonly used as the protein level input. We propose different methods to perform consistent predictions at the two

levels, using maximum likelihood and constrained optimization. The resulting algorithms display improved noise immunity.

In Chapter 6, we switch to the problem of predicting gene regulatory networks in an unsupervised setting, which is more realistic for organisms that are not well studied. We consider two types of data features, namely steady-state gene expression profiles after gene knockout, and dynamic expression time series after an initial perturbation. While the two types of data provide complementary information for predicting gene regulation, many existing algorithms have overlooked the potential of the gene knockout expression profiles. By developing a new procedure for identifying gene regulation from such profiles, we were able to combine the information hidden in the two types of data and make more accurate predictions. The effectiveness of the algorithm was demonstrated in a public challenge using benchmark datasets, in which our algorithm achieved the best accuracy among 28 other teams [202].

While the content in the first part of the thesis represents work in the core step of a typical network reconstruction study, we emphasize that a successful study also implicitly involves a lot of non-trivial tasks before and after the actual reconstruction stage. In the second and third parts of the thesis, we explore two of them, namely data integration and software sharing.

In the second part of the thesis, we discuss our work on data gathering and integration, which is a difficult task as biological data are distributed and involve multiple naming conventions and data formats. We study two different approaches to integrating biological data. In Chapter 7, we use the knowledge representation formalism of semantic web [20] to build a common platform called YeastHub [39] for integrating heterogeneous data from different sources.

We treat the integration of data by semantic web as a long-term endeavor, as it involves collaborative ontology building, large-scale data conversion, infrastructure and application software design and development, and extensive training for software engineers and users. To provide a short-term solution, in Chapter 8, we discuss our work in using Web 2.0 techniques [140] in integrating life sciences data. With the simple user-friendly interfaces, biologists could easily build reusable modules for performing their daily data integration tasks without writing any programs or scripts. We explore the potential and current limitations of such techniques in several applications in public health and molecular biology [40, 165]. At the end of the chapter, we compare the two data integration approaches, and suggest possible future directions.

While it is scientifically significant to demonstrate the effectiveness of new algorithms on some specific datasets, the research community would benefit much more if the algorithms are made publicly accessible, so that other groups could apply them on their own data without spending extra resources on re-implementation. In the third part of the thesis, we describe two web platforms that we developed for sharing our algorithms.

In Chapter 9, we describe tYNA (the Yale Network Analyzer) [206], which is a web tool for network analysis. Its functionality includes statistics calculations, motif finding, visualization, and network comparisons. The tool has been used by around 200 researchers worldwide to analyze around 1,500 networks.

In Chapter 10, we describe a tool for studying residue co-evolution [205], which can be viewed as a special kind of network with each node representing an amino acid residue and two nodes are connected if the corresponding residues have undergone co-evolution across different species. There are many ways to mathematically quantify the likelihood of co-evolution between two residues. On the web site, we provide the implementation of more

than 100 variations of such co-evolution scoring functions, and allow users to study the co-evolution networks of their own proteins. The site has processed more than 1,000 tasks, and the programs have been downloaded and installed locally on the servers of a number of other research groups to facilitate their large-scale studies of residue co-evolution.

We conclude the thesis in Chapter [11](#) and point out potential future directions.

## Chapter 2

# Biological Background

In this chapter, we introduce some basic biological concepts and experimental techniques. The goal is to explain the important terms useful for understanding this thesis, without delving into too much detail. A lot of related basic concepts will be omitted, and some concepts will be presented in a simplified way. In particular, exceptions will not be mentioned if they are rare. Additional concepts will be introduced in later chapters when the need arises.

### 2.1 DNA, RNA and proteins

The basic unit of living systems is the cell. For most species, the heritable information that distinguishes one organism from another is stored in the *deoxyribonucleic acid (DNA)* sequences in living cells. Each DNA sequence is a linear chain of nucleotides. There are four types of nucleotides in DNA sequences: adenine (A), cytosine (C), guanine (G) and

thymine (T). A DNA sequence can thus be represented by a string using an alphabet with four characters.

In a cell, DNA sequences are arranged in a double-stranded helical structure, where the nucleotides on the two strands are complementary to each other so that A is paired with T, G is paired with C, and vice versa.

The DNA in a cell can be divided into different parts called *chromosomes*. All chromosomes together form the *genome* of an organism. If an organism has two copies of the set of chromosomes, it is said to be *diploid*. If there is only one copy, it is *haploid*.

For higher organisms such as humans, DNA is stored in the nucleus of a cell. Organisms having cells with a clear nucleus are called *eukaryotes*. Organisms without clear cell nucleus are called *prokaryotes*.

In a DNA sequence, there are parts that can be used as templates to generate products called *ribonucleic acids (RNA)*. These parts are called *genes*, while the other parts are called intergenic regions. The process of generating RNA from DNA is called *transcription*. Like DNA, an RNA sequence is also composed of nucleotides. There are four such types of nucleotides in RNA: A, C, G and Uracil (U). The transcription process ensures that the resulting RNA is complementary to the DNA on the gene, with A being complementary to U in this case.

After transcription, some unwanted parts on the RNA are removed in higher organisms. The corresponding DNA of the retained and removed parts are called *exons* and *introns*, respectively. Most genes do not have RNA as their end products, but rather the RNA is further used as a template to generate *amino acid* chains, which after folding into particular three-dimensional structures are called *proteins*. The process of creating protein from RNA

is called *translation*. The translation of RNA into amino acids is again based on fixed rules. The DNA sequence is read 3 at a time, with each triple (called a *codon*) encoding one of the twenty types of amino acid.

When two amino acids are joined together, a water molecule is expelled from the bonding called peptide bond. Each remaining amino acid is thus given the name of a *residue* (what is left after expelling water). In other words, residues will be used as the basic units of proteins, just as nucleotides are the basic units of DNA and RNA. Although logically the chain of amino acids of a protein should be called an amino acid sequence, it is more commonly called a protein sequence and we will follow this convention. Be cautious that a protein sequence is not a sequence of multiple proteins, but the sequence of multiple amino acid residues of a protein.

The content of DNA is subject to change by mutations. If a DNA region is important, so that mutations in the region could cause serious survival problems, the region in survived organisms all have relatively few mutations and therefore look more similar to each other: the region is said to be more *conserved*. For example, if two species both require a certain protein to survive, then the encoding DNA sequence will be highly conserved across the two genomes. To identify the well-conserved and poorly-conserved regions, the DNA/protein sequence of the same gene/protein can be used to form an *alignment* by using alignment algorithms that try to minimize the number of mutations required to change one to the other. The idea can be generalized to involve more than two sequences, and the resulting alignment is called a *multiple sequence alignment (MSA)*.

Protein sequences have been compared to look for conserved regions. The conserved sequences have been named *motifs* and *domains*, usually for shorter and longer sequences, respectively. They are crucial to the functions and structures of proteins, and their inter-



actions with other biological objects.

Based on the concept of conservation, one may ask which species has a certain gene based on a reference sequence of the gene and a similarity/mutation threshold. By examining multiple species, each gene receives a binary vector with each bit indicating whether the gene is present in a given species. The vector is called a *phylogenetic profile* of the gene.

The cell contains not only the nucleus, but also many other compartments. A protein typically resides in only some of the compartments. A binary vector similar to a phylogenetic profile can be constructed for a protein based on the cell compartments where it can be found. The resulting data is called the *localization profile*.

Another type of large-scale dataset for genes is their RNA expression levels as measured by *microarray* experiments. A microarray consists of many small wells, each containing many copies of one type of target sequence, such as the complementary sequence of a gene region. To measure how much RNA produced by the gene is present in a sample, a small portion of sample is added to the well. The RNA in the sample is hybridized (bound) to the complementary sequences in the well. The amount is detected by fluorescence. By adding a portion of the sample to every well, the resulting measurements tell the relative RNA expression levels of different genes in the sample. The whole set of experiments can also be repeated for other samples, for example from the same cells in different conditions, which would produce data allowing for comparison of the activity of the genes across different conditions.

A gene can be artificially disabled by *knockout experiments* such as mutagenesis. For a diploid organism, it is possible to knock out only one of the copies, or both. The resulting strain of the former is called *heterozygous* and the latter is called *homozygous*.

## 2.2 Biological networks

In this thesis our primary interest is not individual biological objects, but the interaction between different objects in *networks*. There are many different types of biological networks.

A *protein-protein interaction (PPI) network* records which proteins have physical interactions with each other. The edges are undirected. We will assume that the interactions are binary, i.e., involving only two proteins. Real biological systems contain protein complexes that involve multiple proteins (and also multiple copies of the same proteins) physically binding together. Each complex can be represented by a set of binary interactions.

In this thesis, we define an edge of a protein-protein interaction network as two proteins that interact in at least one condition. We do not consider whether the proteins are permanently bound together or just transiently interacting. We also do not consider whether two protein interactions can simultaneously occur.

Protein interactions can be detected by small-scale experiments such as *western blotting*. Large-scale detection methods have also been proposed. The two most popular methods are *yeast-two-hybrid (Y2H)* and *tandem affinity purification with mass spectrometry (TAP-MS)*. The former detects binary interactions happening in the cell nucleus, while the latter pulls down whole complexes without revealing the connections within. Though neither of them provides the complete set of binary interactions and both have high error rates, these experiments are the current state-of-the-art in large-scale detection of protein interactions.

In *gene-regulatory networks*, edges are drawn from a regulator to its target. Transcription is controlled by regulators called *transcription factors (TFs)*. They are proteins that

recognize and bind to certain gene regions, to either *activate* or *suppress* the transcription. In graph-theoretic terms, the edges are directed and signed, where a positive sign means activation and a negative sign means suppression.

There are large-scale experiments for detecting TF binding, including *chromatin immunoprecipitation with microarray (ChIP-chip)* or with sequencing (*ChIP-seq*).

*Metabolic networks* are more commonly called *metabolic pathways*, which involve the conversion of metabolites from one form to another through enzymatic actions. The most common representation has the metabolites as nodes. There is an edge from one node to another if the former can be converted into the latter by the action of an enzyme. The enzyme is used as the label of the directed edge. Some reactions are reversible. In such cases, there are two edges between the nodes, one from the first node to the second, and the other from the second back to the first.

In *co-evolution networks*, each node is a biological object and two nodes are connected if they are evolutionarily linked so that mutations of one would trigger corresponding mutations of the other. Co-evolution networks can be defined at multiple levels, from single nucleotides in DNA, to single residues in the same protein or different proteins, to whole proteins.

The term “*genetic interaction*” sometimes refers generally to the interaction between different genes and their products [70], and sometimes refers specifically to the situation that the absence or change of dosage of the products of two genes together (the *genotypes*) causes some unexpected outcomes (the *phenotypes*) [182]. The most well known example is *synthetic lethality*, in which a cell can survive the deletion of either of two genes, but it cannot survive if both genes are deleted. By having each gene as a node and putting an

edge between two genes that have a certain type of genetic interaction, the resulting genetic interaction network reflects some interesting special relationships between the genes. For example, it has been shown that genes that have synthetic lethality are more likely to be in parallel biological pathways [102].

A related network is the *gene-drug network*, in which there are two sets of nodes, one for genes and one for drugs. An edge is drawn from a drug to a gene if the latter is the target of the former. The network is thus a directed, bipartite graph. Since a drug could cause inhibition of proteins, applying a drug to a cell has a net effect similar to knocking out (or partially knocking out, i.e., *knocking down*) the encoding genes of the proteins. By carefully comparing the genetic interaction network and gene-drug network, one could predict drug targets and get more insights about the biological pathways a protein participates in.

**Part I**

**Reconstructing Biological  
Networks**

## Chapter 3

# Exploiting Data Properties: Training Set Expansion

### 3.1 Introduction

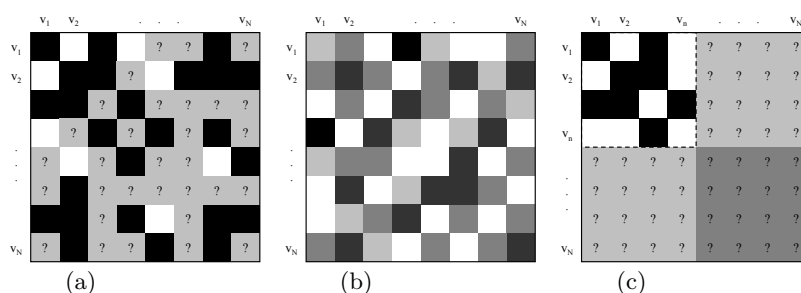
Biological networks offer a global view of the relationships between biological objects. In recent years high-throughput experiments have enabled large-scale reconstruction of the networks. However, as these data are usually incomplete and noisy, they can only be used as a first approximation of the complete networks. For example, a recent study reports that the false positive and negative rates of yeast two-hybrid protein-protein interaction data could be as high as 25%-45% and 75%-90% respectively [90], and a recently published dataset combining multiple large-scale yeast-two-hybrid screens is estimated to cover only 20% of the yeast binary interactome [207]. As another example, as of July 2008, the synthetic lethal interactions in the BioGRID database [29] (version 2.0.42) only involve 2505 yeast

genes, while there are about 5000 non-essential genes in yeast [72]. A large part of the genetic network is likely not yet discovered.

To complement the experimental data, computational methods have been developed to assist the reconstruction of the networks. These methods learn from some example interactions, and predict the missing ones based on the learned models.

This problem is known as supervised network inference [187]. The input to the problem is a graph  $G = (V, E, \bar{E})$  where  $V$  is a set of nodes each representing a biological object (e.g. a protein), and  $E, \bar{E} \subset V \times V$  are sets of known edges and non-edges respectively, corresponding to object pairs that are known to interact and not interact respectively. For each of the remaining pairs, whether they interact is not known (Figure 3.1(a)). A model is to be learned from the data, so that when given any object pair  $(v_i, v_j)$  as input, it will output a prediction  $y \in [0, 1]$  where a larger value means a higher chance of interaction between the objects.

The models are learned according to some data features that describe the objects. For example, in predicting protein-protein interaction networks, functional genomic data are commonly used. In order to learn models that can make accurate predictions, it is usually required to integrate heterogeneous types of data that contain different kinds of information. Since the data are in different formats (e.g. numeric values for gene expression, strings for protein sequences), integrating them is non-trivial. A natural choice for this complex data integration task is kernel methods [164], which unify the data representation as special matrices called kernels and facilitate easy integration of these kernels into a final kernel  $K$  through various means [111] (Figure 3.1(b)). As long as  $K$  is positive semi-definite,  $K(v_i, v_j)$  represents the inner product of objects  $v_i$  and  $v_j$  in a certain embedded space [130], which can be interpreted as the similarity between the objects. Kernel methods then learn the



**Figure 3.1.** The supervised network inference problem. (a) Adjacency matrix of known interactions (black boxes), known non-interactions (white boxes), and node pairs with an unknown interaction status (gray boxes with question marks). (b) Kernel matrix, with a darker color representing a larger inner product. (c) Partially-complete adjacency matrix required by the supervised direct approach methods, with complete knowledge of a submatrix. In the basic local modeling approach, the dark gray portion cannot be predicted.

models from the training examples and the inner products [2]. Since network reconstruction involves many kinds of data, in this study we will focus on kernel methods for learning.

The supervised network inference problem differs from most other machine learning settings in that instead of making a prediction for each input object (such as a protein), the learning algorithm makes a prediction for each pair of objects, namely how likely these objects interact in the biological network. Since there is a quadratic number of object pairs, the computational cost could be very high. For instance, while learning a model for the around 6000 genes of yeast is not a difficult task for contemporary computing machines, the corresponding task for the around 18 million gene pairs remains challenging even for high-end computers. Specialized kernel methods have thus been developed for this learning problem.

For networks with noisy high-throughput data, reliable “gold-standard” training sets



are to be obtained from data verified by small-scale experiments or evidenced by multiple methods. As the number of such interactions is small, there is a scarcity of training data. In addition, the training data from small-scale experiments are usually biased towards some well-studied proteins, creating an uneven distribution of training examples across proteins.

In the next section, we review some existing computational approaches to reconstructing biological networks. One recent proposal is *local modeling* [21], which allows for the construction of very flexible models by letting each object construct a different *local model*, and has been shown promising in some network reconstruction tasks. However, when there is a scarcity of training data, the high flexibility could turn out to be a disadvantage, as there is a high risk of overfitting, i.e., the construction of overly complex models that fit the training data well but do not represent the general trend of the whole network. As a result, the prediction accuracy of the models could be affected.

In this study we propose methods called *training set expansion* that alleviate the problem of local modeling while preserving its modeling flexibility. They also handle the issue of uneven training examples by propagating knowledge from information-rich regions to information-poor regions. We will show that the resulting algorithms are highly competitive with the existing approaches in terms of prediction accuracy. We will also present some interesting findings based on the prediction results.

## 3.2 Related work: existing approaches for network reconstruction

### 3.2.1 The pairwise kernel approach

In the pairwise kernel (Pkernel) approach [17], the goal is to use a standard kernel method (such as SVM) to make the predictions by treating each object pair as a data instance (Figure 3.2(a,b)). This requires the definition of an embedded space for object pairs. In other words, a kernel is to be defined, which takes two pairs of objects and returns their inner product. With  $n$  objects, the kernel matrix contains  $O(n^4)$  entries in total.

One systematic approach to constructing such *pairwise kernels* is to build them on top of an existing kernel for individual objects, in which each entry corresponds to the inner product of two objects. For example, suppose a kernel  $K$  for individual objects is given, and  $v_1, v_2, v_3, v_4$  are four objects, the following function can be used to build the pairwise kernel [17]:

$$K'((v_1, v_2), (v_3, v_4)) = K(v_1, v_3)K(v_2, v_4) + K(v_1, v_4)K(v_2, v_3) \quad (3.1)$$

Loosely speaking, two object pairs are similar if the two objects in the first pair are respectively similar to different objects in the second pair.

### 3.2.2 The direct approach

The direct approach [199] avoids working in the embedded space of object pairs. Instead, only a kernel for individual objects is needed. Given such an input kernel  $K$  and

a cutoff threshold  $t$ , the direct approach simply predicts each pair of objects  $(v_i, v_j)$  with  $K(v_i, v_j) \geq t$  to interact, and each other pair to not interact. Since the example interactions and non-interactions are not used in making the predictions, this method is unsupervised.

The direct approach is related to the pairwise kernel approach through a simple pairwise kernel:

$$K'((v_1, v_2), (v_3, v_4)) = K(v_1, v_2)K(v_3, v_4) \quad (3.2)$$

With this kernel, each object pair  $(v_i, v_j)$  is mapped to the point  $K(v_i, v_j)$  on the real line in the embedded space of object pairs. Thresholding the object pairs at a value  $t$  is equivalent to placing a hyperplane in the embedded space with all pairs  $(v_i, v_j)$  having  $K(v_i, v_j) \geq t$  on one side and all other pairs on the other side. Therefore, if this pairwise kernel is used, then learning a linear classifier in the embedded space is equivalent to learning the best value for threshold  $t$ .

To make use of the training examples, two supervised versions of the direct approach have been proposed. They assume that the sub-network of a subset of objects is completely known, so that a submatrix of the adjacency matrix is totally filled (Figure 3.1(c)). The goal is to modify the similarity values of the objects defined by the kernel to values that are more consistent with the partial adjacency matrix. Thresholding is then performed on the resulting set of similarity values.

The two versions differ in the definition of consistency between the similarity values and the adjacency matrix. In the kernel canonical correlation analysis (kCCA) approach [199], the goal is to identify feature  $f_1$  from the input kernel and feature  $f_2$  from the diffusion kernel derived from the partial adjacency matrix so that the two features have the highest

correlation under some smoothness requirements. Additional feature pairs orthogonal to the previous ones are identified in similar ways, and the first  $l$  pairs are used to redefine the similarity between objects.

In the kernel metric learning (kML) approach [187], a feature  $f_1$  is identified by optimizing a function that involves the distance between known interacting objects. Again, additional orthogonal features are identified, and the similarity between objects is redefined by these features.

### 3.2.3 The matrix completion approach

The em approach [184] (which is theoretically related to the expectation-maximization (EM) framework) also assumes a partially complete adjacency matrix. The goal is to complete it by filling in the missing entries, so that the resulting matrix is closest to a spectral variant of the kernel matrix as measured by KL-divergence. The algorithm iteratively searches for the filled adjacency matrix that is closest to the current spectral variant of the kernel matrix, and the spectral variant of the kernel matrix that is closest to the current filled adjacency matrix. When convergence is reached, the predictions are read from the final completed adjacency matrix.

### 3.2.4 The local modeling approach

A potential problem of the previous approaches is that one single model is built for all object pairs. If there are different subgroups of interactions, a single model may not be able to separate all interacting pairs from non-interacting ones. For example, protein pairs involved in transient interactions may use a very different mechanism than those involved

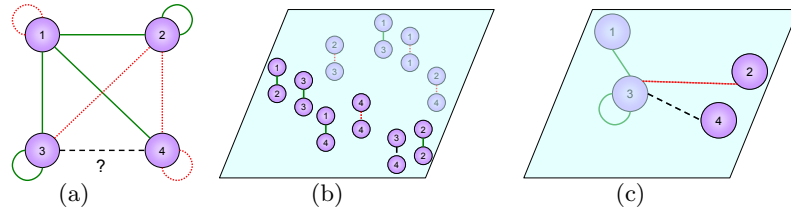
in permanent complexes. These two types of interactions may form two separate subgroups that cannot be fitted by one single model.

A similar problem has been discussed in Myers and Troyanskaya [135]. In this work, the biological context of each gene is taken into account by conditioning the probability terms of a Bayesian model by the biological context. The additional modeling power of having multiple context-dependent sub-models was demonstrated by improved accuracy in network prediction.

Another way to allow for a more flexible modeling of the subgroups is *local modeling* [21]. Instead of building a single global model for the whole network, one local model is built for each object, using the known interactions and non-interactions of it as the positive and negative examples. Each pair of objects thus receives two predictions, one from the local model of each object. In our implementation, the final prediction is a weighted sum of the two according to the training accuracy of the two local models.

Figure 3.2 illustrates the concept of local modeling. Part (a) shows an interaction network, with solid green lines representing known interactions, dotted red lines representing known non-interactions, and the dashed black line representing an object pair of which the interaction status is unknown. Part (b) shows a global model with the locations of the object pairs determined by a pairwise kernel. The object pair  $(v_3, v_4)$  is on the side with many positive examples, and is predicted to interact. Part (c) shows a local model for object  $v_3$ . Object  $v_4$  is on the side with a negative example, and  $(v_3, v_4)$  is predicted to not interact.

Since each object has its own local model, subgroup structures can be readily handled by having different kinds of local models for objects in different subgroups.



**Figure 3.2. Global and local modeling.** (a) An interaction network with each green solid edge representing a known interaction, each red dotted edge representing a known non-interaction, and the dashed edge representing a pair of objects with an unknown interaction status. (b) A global model based on a pairwise kernel. (c) A local model for object  $v_3$ .

### 3.3 Our proposal: the training set expansion approach

Local modeling has been shown to be very competitive in terms of prediction accuracy [21]. However, local models can only be learned for objects with a sufficiently large amount of known interactions and non-interactions. When the training sets are small, many objects would not have enough data for training their local models. Overfitting may occur, and in the extreme case where an object has no positive or negative examples, its local model simply cannot be learned. As to be shown in our empirical study presented below, this problem is especially serious when the embedded space is of very high dimension, since very complex models that overfit the data could be formed.

In the following we propose ways to tackle this data scarcity issue while maintaining the flexibility of local modeling. Our idea is to expand the training sets by generating auxiliary training examples. We call it the *training set expansion* approach. Obviously these auxiliary training examples need to be good estimates of the actual interaction status of the corresponding object pairs, for expanding the training sets by wrong examples could further

worsen the learned models. We propose two methods for generating reliable examples: *prediction propagation* and *kernel initialization*.

### 3.3.1 Prediction propagation (pp)

Suppose  $v_1$  and  $v_2$  are two objects, where  $v_1$  has sufficient training examples while  $v_2$  does not. We first train the local model for  $v_1$ . If the model predicts with high confidence that  $v_1$  interacts with  $v_2$ , then  $v_1$  can later be used as a positive example for training the local model of  $v_2$ . Alternatively, if the model predicts with high confidence that  $v_1$  does not interact with  $v_2$ ,  $v_1$  can be used as a negative example for training the local model of  $v_2$ .

This idea is based on the observation that predictions that a model is most confident with are more likely correct. For example, if the local models are support vector machines, the predictions for objects far away from the separating hyperplane are more likely correct than those for objects falling in the margin. Therefore, to implement the idea, each prediction should be associated with a confidence value obtained from the local model. When expanding the training sets of other objects, only the most confident predictions should be involved.

We use support vector regression [172] to produce the confidence values. When training the local model of an object  $v_i$ , the original positive and negative examples of it are given labels of 1 and -1 respectively. Then a regression model is constructed to find the best fit. Objects close to the positive examples will receive a regressed value close to 1, meaning that they correspond to objects that are likely to interact with  $v_i$ . Similarly, objects close to the negative examples will receive a regressed value close to -1, and hence correspond to objects that are likely to not interact with  $v_i$ . For other objects, the model is less confident in

telling whether they interact with  $v_i$ . Therefore the predictions with large positive regressed values can be used as positive examples for training other local models, and those with large negative regressed values can be used as negative examples, where the magnitudes of the regressed values represent the confidence.

Each time we use  $p\%$  of the most confident predictions to expand the training sets of other objects, where the numbers of new positive and negative examples are in proportion to the ratio of positive and negative examples in the original training sets. The parameter  $p$  is called the *training set expansion rate*.

To further improve the approach, we order the training of local models so that objects with more (original and augmented training examples) are trained first, as the models learned from more training examples are generally more reliable. Essentially this is handling the uneven distribution of training examples by propagating knowledge from the information-rich regions (objects with many training examples) to the information-poor regions (objects with no or few training examples).

Theoretically prediction propagation is related to co-training [22], which uses the most confident predictions of a classifier as additional training examples of other classifiers. The major differences are that in co-training, the classifiers are to make predictions for the same set of data instances, and the classifiers are complementary to each other due to the use of different data features. In contrast, in prediction propagation, each model is trained for a different object, and the models are complementary to each other due to the use of different training examples.

Instead of regression, one can also use support vector classifier (SVC) to determine the confidence values, by measuring the distance of each object from the separating hyperplane.



Since we only use the ranks of the confidence values to deduce the auxiliary examples but not their absolute magnitudes, we would expect the results to be similar. We implemented both versions and tested them in our experiments. The two sets of results are indeed comparable, with SVR having slightly higher accuracy on average, as we will see in the experiment section.

### 3.3.2 Kernel initialization (ki)

The prediction propagation method is effective when some objects have sufficient input training examples at the beginning to start the generation of auxiliary examples. Yet if all objects have very few input training examples, even the object with the largest training sets may not be able to form a local model that can generate accurate auxiliary examples.

An alternative way to generate auxiliary training examples is to estimate the interaction status of each pair of objects by its similarity value given by the kernel. This is in line with the idea of the direct approach, that object pairs with a larger similarity value are more likely to interact. However, instead of thresholding the similarity values to directly give the predictions, they are used only to initialize the training sets for learning the local models. Also, to avoid generating wrong examples, only the ones with the largest and smallest similarity values are used, which correspond to the most confident predictions of the unsupervised direct method.

For each object,  $p\%$  of the objects with the largest/smallest similarity values given by the kernel are treated as positive/negative training examples in proportion to the positive and negative examples in the original training sets. These auxiliary examples are then combined with the original input examples to train the local models.

The kernel initialization method can be seen as adding a special prior to the object pairs, which assigns a probability of 1 to the most similar pairs of each object and 0 to the most dissimilar pairs. We have also tried normalizing the inner products to the  $[0,1]$  range and using them directly as the initial estimate of the confidence of interaction. Yet the performance was not as good as the current method, which could be due to the large variance of confidence values of the object pairs with moderate similarity.

The two training set expansion methods fall within the class of semi-supervised learning methods [34], which make use of both the training examples and some information about all data instances to learn the model. Prediction propagation exploits the information about each object pair produced by other local models to help train the current local model. Kernel initialization utilizes the similarity between objects in the feature space to place soft constraints on the local models, that the objects most similar to the current object should be put in the positive class and those most dissimilar to the current object should be put in the negative class.

### 3.3.3 Combining the two methods (pp+ki)

Since kernel initialization is applied before learning while prediction propagation is applied during learning, the two can be used in combination. In some cases this leads to additional performance gain in our experiments.

## 3.4 Prediction accuracy

### 3.4.1 Data and setup

To test the effectiveness of the training set expansion approach, we compared its prediction accuracy with the other approaches on three protein-protein interaction networks of the yeast *Saccharomyces cerevisiae* from BioGRID [29], DIP [160], MIPS [131] and iPfam [63]. The BioGRID-10 dataset contains all BioGRID interactions of *Saccharomyces cerevisiae* (version 2.0.44) that satisfy the following criteria:

1. Having one of the following physical interaction types:
  - FRET
  - Protein-peptide
  - Co-crystal Structure
  - Co-fractionation
  - Co-purification
  - Reconstituted Complex
  - Biochemical Activity
  - Affinity Capture-Western
  - Two-hybrid
  - Affinity Capture-MS
2. From one of the small-scale studies, defined as studies that report less than 10 physical interactions to BioGRID

3. The proteins/genes involved in the interactions have valid values from all the features for learning

The cutoff (10 physical interactions) was chosen so that the network is large enough to have relatively few missing interactions, while small enough to run the different algorithms in reasonable time. The dataset contains 5,126 interactions that involve 2,328 yeast proteins.

The BioGRID-200 dataset is similar to BioGRID-10, except that small-scale studies are defined as studies that report less than 200 physical interactions to BioGRID. Notice that since the four high-throughput datasets used as data features all have more than 200 interactions, they are not included in this dataset. The dataset contains 12,155 interactions that involve 3,222 yeast proteins.

The DIP\_MIPS\_iPfam dataset contains the union of all interactions from DIP (7 Oct 2007 version), MIPS (18 May 2006 version) and iPfam (version 21 of Pfam) that satisfy the following criteria:

1. For interactions in DIP, only those identified in small-scale experiments or multiple experiments are considered
2. For interactions in MIPS, only the physical, non-Yeast two hybrid and non-TAP-MS ones are considered
3. The involving proteins/genes have valid values from all the features for learning

The dataset contains 3,201 interactions that involve 1,681 yeast proteins.

We use BioGRID-10 as the main dataset for comparison, while DIP\_MIPS\_iPfam represents a high quality but smaller dataset, and BioGRID-200 represents one with few missing

**Table 3.1. List of datasets used in the comparison study. Each row corresponds to a dataset from a publication in the Source column, and is turned into a kernel using the function in the Kernel column, as in previous studies [21, 199].**

Code	Data type	Source	Kernel
phy	Phylogenetic profiles	COG v7 [181]	RBF ( $\sigma=3,8$ )
loc	Sub-cellular localization	[92]	Linear
exp-gasch	Gene expression (environmental response)	[68]	RBF ( $\sigma=3,8$ )
exp-spellman	Gene expression (cell-cycle)	[176]	RBF ( $\sigma=3,8$ )
y2h-ito	Yeast two-hybrid	[97]	Diffusion ( $\beta=0.01$ )
y2h-uetz	Yeast two-hybrid	[185]	Diffusion ( $\beta=0.01$ )
tap-gavin	Tandem affinity purification	[69]	Diffusion ( $\beta=0.01$ )
tap-krogan	Tandem affinity purification	[108]	Diffusion ( $\beta=0.01$ )
int	Integration		Summation

interactions, but is too large that the pairwise kernel method could not be tested as it caused our machine to run out of memory. The three datasets together allow us to show the effectiveness of training set expansion in a wide spectrum of scenarios.

We tested the performance of the different approaches on various kinds of genomic data features, including phylogenetic profiles, sub-cellular localization and gene expression datasets using the same kernels and parameters as in previous studies [21, 200]. We also added in datasets from tandem affinity purification with mass spectrometry using the diffusion kernel, and the integration of all kernels by summing them after normalization, as in previous studies [21, 200]. The list of datasets used is shown in Table 3.1.

We performed ten-fold cross validations and used the area under the receiver operator characteristic curve (AUC) as the performance metric. The cross validations were done in two different modes. In the first mode, as in previous studies [21, 199], the proteins were

divided into ten sets. Each time one set was left out for testing, and the other nine were used for training. All known interactions with both proteins in the training set were used as positive training examples. As required by some of the previous approaches, the sub-network involving the proteins in the training set was assumed completely known (Figure 3.1(c)). As such, all pairs of proteins in the training set not known to interact were regarded as negative examples. All pairs of proteins with exactly one of the two proteins in the training set were used as testing examples (light gray entries in Figure 3.1(c)). Pairs with both proteins not in the training set were not included in the testing sets (dark gray entries in Figure 3.1(c)), as the original local modeling method cannot make such predictions.

Since all protein pairs in the submatrix are either positive or negative training examples, there are  $O(n^2)$  training examples in each fold. In the pairwise kernel approach, this translates to a kernel matrix with  $O(n^4)$  elements. This is of the order of  $10^{12}$  for 1,000 proteins, which is too large to compute and to learn the SVC and SVR models. We therefore did not include the pairwise kernel method in the experiments that used the first mode of cross-validation.

Since some protein pairs treated as negative examples may actually interact, the reported accuracies may not completely reflect the absolute performance of the methods. However, as the tested methods were subject to the same setting, the results are still good indicators of the relative performance of the approaches.

In the second mode of cross-validation, we randomly sampled protein pairs not known to interact to form a negative training set with the same size as the positive set, as in previous studies [17, 151]. Each of the two sets was divided into ten subsets, which were used for left-out testing in turn. The main difference between the two modes of cross-validation is that the train-test split is based on proteins in the first mode and protein pairs

in the second mode. Since the training examples do not constitute a complete submatrix, the kCCA, kML and em methods cannot be tested in the second mode. The second mode represents the more general case, where the positive and negative training examples do not necessarily form a complete sub-network.

We used the Matlab code provided by Jean-Philippe Vert for the unsupervised direct, kCCA, kML and em methods with the first mode of cross-validation. We implemented the other methods with both the first and second modes of cross-validation. We observed almost identical accuracy values from the two implementations of the direct approach in the first mode of cross-validation with the negligible differences due only to random train-test splits, which confirms that the reported values from the two sets of code can be fairly compared. For the pairwise kernel approach, we used the kernel in Equation 3.1.

We used the  $\epsilon$ -SVR and C-SVC implementations of the Java version of libsvm [33]. In a preliminary study, we observed that the prediction accuracy of SVR is not much affected by the value of the termination threshold  $\epsilon$ , while for both SVR and SVC the performance is quite stable as long as the value of the regularization parameter C is not too small. We thus fixed both parameters to 0.5. For *PP* and *KI*, we used a grid search to determine the value of the training set expansion rate  $p$ .

### 3.4.2 Results

Since we use datasets different from the ones used in previous studies, the prediction results are expected to be different. To make sure that our implementations are correct and the testing procedure is valid, we compared our results on the DIP\_MIPS\_iPfam dataset with those reported in Bleakley et al. [21] as the size of this dataset is most similar to the

one used by them. Our results (Table 3.4) display a lot of similarities with those in Bleakley et al. [21]. For example, in the first mode of cross-validation, local modeling outperformed the other previous approaches when object similarity was defined by phylogenetic profiles and yeast two-hybrid data. Also, the em method had the best performance among all previous approaches with the integrated kernel in both studies. We are thus confident that our results represent a reliable comparison between the methods.

The comparison results for our main dataset, BioGRID-10, are shown in Table 3.2. In the table pp, ki and pp+ki are written as local+pp, local+ki and local+pp+ki, respectively, to emphasize that the two training set expansion methods are used on top of basic local modeling. Notice that the accuracies in the second mode of cross-validation are in general higher. We examined whether this is due to the presence of self-interactions in the gold-standard set of the second mode of cross-validation but not in the first mode, by removing the self-interactions and re-running the experiments. The results suggest that the performance gain due to the removal of self-interactions is too small to explain the performance difference between the two modes of cross-validation. The setting in the second mode may thus correspond to an easier problem. The reported accuracies of the two modes should therefore not be compared directly.

From the table, the advantages of the training set expansion methods over basic local modeling are clearly seen. In all cases, the accuracy of local modeling was improved by at least one of the expansion methods, and in many cases all three combinations (pp, ki and pp+ki) performed better than basic local modeling. With training set expansion, local modeling outperformed all the other approaches in all 9 datasets.

Inspecting the performance of local modeling without training set expansion, it is observed that although local modeling usually outperformed the other previous methods,



**Table 3.2. Prediction accuracy (percentage of AUC) of the different approaches on the BioGRID-10 dataset. The best approach for each kernel and each mode of cross-validation is in bold face.**

	phy	loc	exp-gasch	exp-spellman	y2h-ito	y2h-uetz	tap-gavin	tap-krogan	int
Mode 1									
direct	58.04	66.55	64.61	57.41	51.52	52.13	59.37	61.62	70.91
kCCA	65.80	63.86	68.98	65.10	50.89	50.48	57.56	51.85	80.98
kML	63.87	68.10	69.67	68.99	52.76	53.85	60.86	57.69	73.47
em	71.22	75.14	67.53	64.96	55.90	53.13	63.74	68.20	81.65
local SVM	71.53	71.17	70.35	68.98	67.26	67.25	64.59	67.48	74.77
local+pp SVM	72.07	69.64	76.02	73.54	71.50	71.46	74.41	71.09	82.94
local+ki SVM	71.72	71.15	75.84	71.00	69.32	69.03	70.66	71.89	81.75
local+pp+ki SVM	71.78	70.40	76.73	71.37	70.42	70.43	73.49	72.47	83.19
local SVR	71.67	71.41	72.66	70.63	67.27	67.27	64.60	67.48	75.65
local+pp SVR	<b>73.89</b>	<b>75.25</b>	<b>77.43</b>	<b>75.35</b>	<b>71.60</b>	<b>71.51</b>	<b>74.62</b>	71.39	<b>83.63</b>
local+ki SVR	71.68	71.42	75.89	70.96	69.40	69.05	70.53	72.03	81.74
local+pp+ki SVR	72.40	75.19	77.41	73.81	70.44	70.57	73.59	<b>72.64</b>	83.59
Mode 2									
direct	59.99	67.81	66.18	59.22	54.02	54.64	62.28	63.69	72.34
Pkernel	72.98	69.84	78.61	77.30	57.01	54.65	71.16	70.36	87.34
local SVM	76.17	78.68	76.07	73.46	72.26	72.23	68.39	72.48	81.29
local+pp SVM	75.85	73.66	79.71	75.61	74.05	73.80	75.89	75.10	87.80
local+ki SVM	76.06	78.70	79.02	73.32	72.68	72.03	71.22	75.55	85.53
local+pp+ki SVM	76.32	73.73	79.99	75.48	73.58	73.35	74.98	75.87	87.62
local SVR	76.89	78.73	79.72	77.32	72.93	72.89	68.81	73.15	82.82
local+pp SVR	<b>77.71</b>	<b>80.71</b>	<b>82.56</b>	<b>80.62</b>	<b>74.74</b>	<b>74.41</b>	<b>76.36</b>	75.12	<b>88.78</b>
local+ki SVR	76.76	78.73	80.62	76.44	73.39	72.76	72.42	76.22	86.12
local+pp+ki SVR	77.45	80.57	81.93	78.92	74.14	74.01	75.59	<b>76.59</b>	88.56
Mode 3									
direct	57.72	66.69	64.23	56.86	51.36	52.01	60.10	61.60	70.75
Pkernel	72.01	68.89	77.89	76.37	56.24	53.97	71.48	69.67	87.13
local SVM	76.47	78.56	76.27	73.88	72.57	72.54	68.64	72.81	81.39
local+pp SVM	75.84	73.41	79.93	76.16	74.48	74.21	76.38	75.63	87.79
local+ki SVM	76.40	78.57	79.56	73.90	72.92	72.35	71.63	75.94	85.43
local+pp+ki SVM	76.51	73.43	80.32	75.66	73.70	73.60	75.62	76.24	87.62
local SVR	77.17	78.71	79.87	77.56	73.21	73.18	69.05	73.44	82.97
local+pp SVR	<b>78.18</b>	80.44	<b>82.57</b>	<b>80.41</b>	<b>75.05</b>	<b>74.83</b>	<b>76.76</b>	75.70	<b>88.87</b>
local+ki SVR	77.10	78.71	80.74	76.41	73.51	72.97	72.72	76.53	85.96
local+pp+ki SVR	77.52	<b>80.51</b>	81.73	78.51	74.27	74.09	76.10	<b>76.85</b>	88.55

its performance with the integration kernel was unsatisfactory. This is probably due to overfitting. When kernels are summed, the resulting embedded space is the direct product of the ones defined by the kernels [164]. Since the final kernel used for the integrated dataset is a summation of 8 kernels, the corresponding embedded space is of very high dimension. With the high flexibility and the lack of training data, the models produced by basic local modeling were probably overfitted. In contrast, with the auxiliary training examples, the training set expansion methods appear to have largely overcome the problem.

Comparing the two training set expansion methods, in most cases prediction propagation resulted in a larger performance gain. This is reasonable since the input training examples were used in this method, but not in kernel initialization.

The results for BioGRID-200 and DIP\_MIPS\_iPfam are shown in Table 3.3 and Table 3.4, respectively. They exhibit similar patterns as in the case of BioGRID-10, and thus the above discussion also applies to them.

To better understand how the two training set expansion methods improve the predictions, we sub-sampled the gold-standard network at different sizes, and compared the performance of local modeling with and without training set expansion using the second mode of cross-validation. The results for two of the kernels are shown in Figure 3.3, which show the two typical cases observed.

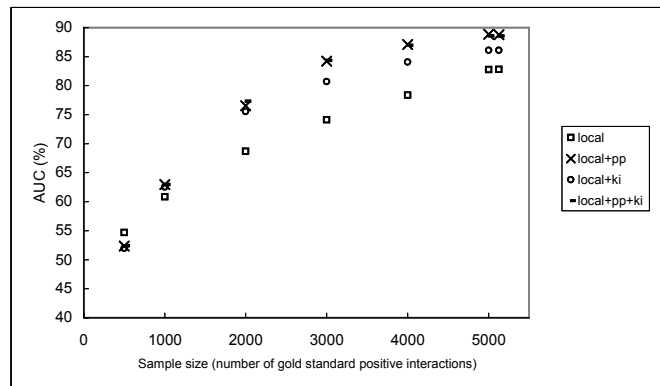
In general training set expansion improved the accuracy the most with moderate gold-standard set sizes, at around 3000 interactions. For prediction propagation, this is expected since when the training set was too small, the local models were too inaccurate that even the most confident predictions could still be wrong, which made propagation undesirable. On the other hand, when there were many training examples, there were few missing in-

**Table 3.3. Prediction accuracy (percentage of AUC) of the different approaches on the BioGRID-200 dataset. The best approach for each kernel and each mode of cross-validation is in bold face.**

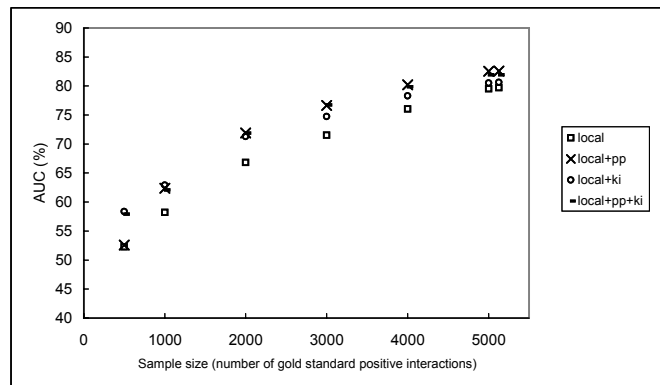
	phy	loc	exp-gasch	exp-spellman	y2h-ito	y2h-uetz	tap-gavin	tap-krogan	int
Mode 1									
direct	58.89	66.32	65.44	59.68	51.87	51.28	63.98	64.56	71.59
kCCA	69.14	66.36	72.30	62.74	53.52	50.85	63.23	58.49	85.73
kML	65.86	68.57	73.79	73.41	55.00	56.12	64.41	62.67	68.82
em	73.60	75.78	68.66	67.55	56.10	53.47	68.76	70.48	80.89
local SVM	76.67	76.78	78.92	77.49	75.08	75.07	71.24	75.34	82.56
local+pp SVM	76.35	75.85	80.02	78.29	75.86	<b>76.48</b>	<b>77.63</b>	76.51	85.36
local+ki SVM	75.88	76.71	80.42	78.04	75.55	75.27	75.15	76.91	85.46
local+pp+ki SVM	76.51	75.73	80.68	78.00	75.91	75.83	76.83	77.10	85.57
local SVR	77.18	76.48	80.23	79.02	75.08	75.07	71.91	75.34	83.09
local+pp SVR	<b>77.60</b>	78.92	<b>81.98</b>	<b>80.59</b>	<b>76.10</b>	76.48	76.67	76.54	<b>85.98</b>
local+ki SVR	75.79	76.50	80.87	78.59	75.59	75.33	75.03	76.96	85.42
local+pp+ki SVR	76.06	<b>78.94</b>	81.71	79.58	75.98	75.94	76.73	<b>77.15</b>	85.83
Mode 2									
direct	60.52	66.81	66.97	61.41	54.01	53.70	65.19	65.81	72.50
local SVM	83.37	83.96	84.94	83.22	81.74	81.75	75.47	81.95	88.76
local+pp SVM	83.26	83.14	86.15	84.23	81.68	81.89	<b>81.34</b>	82.85	91.37
local+ki SVM	81.84	84.00	86.02	82.77	81.30	80.98	78.31	82.63	89.99
local+pp+ki SVM	82.20	83.06	86.17	82.38	81.54	81.54	80.91	82.76	91.16
local SVR	83.88	83.30	86.79	85.54	<b>82.68</b>	82.71	76.36	82.89	89.92
local+pp SVR	<b>84.37</b>	<b>85.62</b>	<b>88.12</b>	<b>87.00</b>	82.43	<b>82.87</b>	80.61	83.65	<b>91.82</b>
local+ki SVR	82.31	83.29	86.93	84.16	82.29	81.99	79.02	83.65	90.17
local+pp+ki SVR	82.63	85.55	87.02	85.03	82.44	82.54	81.09	<b>83.80</b>	91.51
Mode 3									
direct	58.91	65.99	65.61	59.81	52.10	51.79	63.74	64.40	71.37
local SVM	83.63	84.16	85.15	83.55	82.04	82.06	75.72	82.25	88.90
local+pp SVM	83.32	83.70	86.71	84.60	82.33	82.45	<b>81.79</b>	83.59	91.46
local+ki SVM	82.07	84.20	86.54	83.22	81.78	81.49	78.77	83.11	90.05
local+pp+ki SVM	82.50	83.55	86.66	82.75	82.10	82.02	81.64	83.45	91.28
local SVR	84.10	83.51	86.99	85.78	<b>82.99</b>	83.01	76.61	83.20	90.09
local+pp SVR	<b>84.74</b>	85.71	<b>88.21</b>	<b>87.00</b>	82.82	<b>83.37</b>	81.29	<b>84.31</b>	<b>91.88</b>
local+ki SVR	82.49	83.51	87.35	84.27	82.65	82.39	79.41	84.03	90.18
local+pp+ki SVR	82.67	<b>85.74</b>	87.43	85.11	82.79	82.87	81.67	84.25	91.55

**Table 3.4. Prediction accuracy (percentage of AUC) of the different approaches on the DIP\_MIPS\_iPfam dataset. The best approach for each kernel and each mode of cross-validation is in bold face.**

	phy	loc	exp-gasch	exp-spellman	y2h-ito	y2h-uetz	tap-gavin	tap-krogan	int
Mode 1									
direct	63.09	64.23	68.60	62.24	53.40	57.34	63.46	64.58	73.68
kCCA	68.78	62.24	70.93	66.85	55.25	56.70	62.88	62.59	74.45
kML	65.04	67.58	70.09	69.80	58.12	59.90	63.72	61.19	77.58
em	63.22	67.90	65.15	61.74	56.23	58.31	68.02	62.92	78.46
local SVM	72.45	69.90	71.45	69.02	66.56	66.53	64.95	66.92	74.28
local+pp SVM	73.00	70.38	75.69	74.12	<b>72.10</b>	<b>72.10</b>	75.84	71.83	83.26
local+ki SVM	73.67	69.89	76.89	72.01	69.80	69.25	72.75	72.41	82.44
local+pp+ki SVM	72.93	70.76	77.46	72.17	70.86	70.78	74.47	72.81	83.20
local SVR	72.85	70.50	72.89	70.60	66.58	66.56	64.97	66.93	74.76
local+pp SVR	<b>74.48</b>	<b>74.99</b>	78.09	<b>75.89</b>	72.02	72.09	<b>75.88</b>	71.56	<b>83.72</b>
local+ki SVR	74.03	70.47	76.87	72.87	69.88	69.39	72.80	72.43	82.41
local+pp+ki SVR	73.62	74.92	<b>78.35</b>	75.08	70.93	70.97	74.48	<b>73.01</b>	83.39
Mode 2									
direct	67.57	66.48	71.54	66.24	57.74	61.52	67.46	68.86	76.53
Pkernel	73.51	68.24	78.91	77.08	58.10	58.51	72.65	69.98	85.04
local SVM	77.78	77.79	76.67	73.99	72.93	72.98	68.68	73.23	81.10
local+pp SVM	77.42	75.15	79.94	77.10	76.21	76.20	78.45	76.28	87.10
local+ki SVM	78.31	77.80	80.86	75.24	75.52	73.99	74.65	77.51	85.95
local+pp+ki SVM	77.71	74.93	81.38	75.46	75.61	75.83	77.37	78.03	86.75
local SVR	78.78	77.80	79.84	77.38	73.46	73.49	69.01	73.72	82.12
local+pp SVR	<b>79.25</b>	81.65	<b>83.01</b>	<b>81.67</b>	<b>76.76</b>	<b>76.88</b>	<b>79.75</b>	76.99	<b>88.26</b>
local+ki SVR	78.88	77.80	81.55	77.83	76.11	74.62	75.56	78.07	86.51
local+pp+ki SVR	78.78	<b>81.68</b>	82.60	79.90	76.08	76.20	77.79	<b>78.72</b>	87.68
Mode 3									
direct	63.77	64.30	68.19	62.24	52.71	56.94	63.61	65.17	73.78
Pkernel	72.66	66.76	78.42	75.88	57.31	56.90	73.18	69.76	85.63
local SVM	77.91	77.88	76.83	74.23	73.34	73.40	68.58	73.68	81.18
local+pp SVM	77.81	75.17	79.91	76.99	75.93	75.76	78.28	77.45	86.80
local+ki SVM	78.49	77.88	80.82	75.64	74.94	73.45	73.96	77.05	85.30
local+pp+ki SVM	78.17	75.32	81.02	75.28	75.25	75.04	76.87	77.53	86.58
local SVR	78.95	77.89	80.09	77.66	73.95	74.00	69.02	74.28	82.40
local+pp SVR	<b>79.05</b>	<b>80.91</b>	<b>82.42</b>	<b>80.88</b>	<b>76.33</b>	<b>76.30</b>	<b>79.10</b>	77.45	<b>87.54</b>
local+ki SVR	78.86	77.90	81.36	77.39	75.34	73.87	74.60	77.45	85.74
local+pp+ki SVR	78.34	80.85	82.15	79.01	75.39	75.26	77.22	<b>77.92</b>	87.32



(a)



(b)

**Figure 3.3. Prediction accuracy at different gold-standard set sizes. (a) Using the int kernel. (b) Using the exp-gasch kernel.**

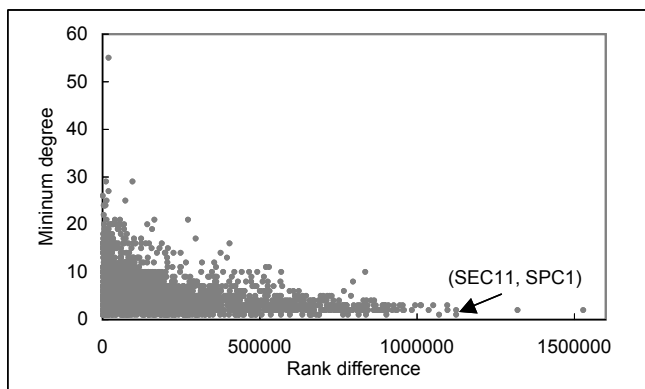
teractions, so that the augmented training examples became relatively less important. The latter argument also applies to kernel initialization, that it resulted in larger performance gain when the gold-standard set was not too large. However, it is surprising to see that using the integrated kernel (Figure 3.3(a)), kernel initialization resulted in a drop in accuracy when there were only 500 interactions. Since the kernel remained the same at different gold-standard set sizes, one would expect to see a stable performance gain for kernel initialization regardless of the size of the gold-standard set. This stable performance gain is indeed observed when the Gasch or phylogenetic profile kernel was used (Figure 3.3(b) and Figure S1). In contrast, prediction propagation, being dependent on the raw accuracy of local modeling, performed poorly when there were only 500 interactions for all 9 datasets. This suggests that when the dataset is expected to contain a lot of missing interactions, kernel initialization is potentially more useful, but it also depends on the feature used in learning. On the other hand, prediction propagation is more useful when the dataset contains enough interactions for local modeling to achieve a reasonable accuracy.

### 3.5 Analysis

With the observed performance gain of training set expansion, we would like to know what kind of correct predictions it could make that were ranked low by other methods. To answer the question, for each known interaction in the gold-standard positive set of BioGRID-10, we computed its rank in the predictions made by local+pp and local+ki using the integrated kernel in the first mode of cross-validation. Then we computed the highest rank of the interaction given by kCCA, kML, em and local, and calculated the difference between the two. If the former is much higher than the latter (i.e., there is a

large rank difference), then the interaction is uniquely identified by training set expansion but not by any of the four other methods.

Among the 2,880 interactions in the gold-standard set that were tested by both local+pp and the four comparing methods, the ranks of 2,121 of them are higher in the predictions made by local+pp than in any of the four methods. For each of them, we computed the minimum degree (number of known interactions in the gold-standard set) of the two interacting proteins as an indicator of the number of available training examples for the pair. Then we correlated the minimum degree with the rank difference. The resulting graph (Figure 3.4) shows a significant negative correlation (Spearman correlation =  $-0.38$ ,  $p < 10^{-16}$ ), which confirms that the correct predictions made by local+pp that were missed by the other four methods correspond to the protein pairs with few known examples. We have also tested the average degree instead of the minimum, and the Pearson correlation instead of Spearman correlation. The results all lead to the same conclusion (Figure S2).



**Figure 3.4. Correlating the number of gold-standard examples and the rank difference between local+pp and the four methods.**

A concrete example of a gold-standard interaction predicted by local+pp but ranked low by the four methods is the one between SEC11 and SPC1. They are both subunits of the signal peptidase complex (SPC), and are reported to interact in BioGRID according to multiple sources. In the BioGRID-10 dataset, SPC1 is the only known interaction partner of SEC11, while SPC1 only has one other known interaction (with SBH2). The extremely small numbers of known examples make it difficult to identify this interaction. Indeed, the best of the four previous methods could only give it a rank at the 74th percentile, indicating that they were all unable to identify this interaction. In contrast, local+pp was able to rank it at the top 7th percentile, i.e., with a rank difference of 67 percentiles (see Figure 3.4). This example illustrates that interactions with very few known examples, while easily missed by the previous methods, could be identified by using prediction propagation.

For local+ki, among the 2,880 commonly tested gold-standard interactions, 2,025 received a higher rank from this method than from any of the four comparing methods. Again, there is a negative correlation between the rank difference and the minimum degree and average degree (Figure S2), which shows that kernel initialization is also able to predict interactions for proteins with few training examples. In addition, there is a positive correlation with moderate significance between the rank difference and the similarity between the interacting proteins according to the kernel (Figure S2, Spearman correlation = 0.04,  $p = 0.04$ ), which is expected as the kernel initialization method uses protein pairs with high similarity as auxiliary positive training examples. Interestingly, for local+pp, a negative correlation is observed between the rank difference and protein similarity (Figure S2), which suggests that the prediction propagation method is able to identify non-trivial interactions, where the two interacting proteins are not necessarily similar according to the kernel.



### 3.6 Discussion

Training set expansion is a general concept that can also be applied to other problems and used with other learning methods. The learning method is not required to make very accurate predictions for all object pairs, and the data features do not need to define an object similarity that is very consistent with the interactions. As long as the *most confident* predictions are likely correct, prediction propagation is useful, and as long as the most similar objects are likely to interact and the most dissimilar objects are unlikely to interact, kernel initialization is useful. In many biological applications at least one of these requirements is satisfied.

In the next chapter, we continue our exploration of the idea of training set expansion. In addition to expanding the training sets of other objects at the same level, we also study ways to expand the training sets of objects at other levels in a natural concept hierarchy of protein interactions.

## Chapter 4

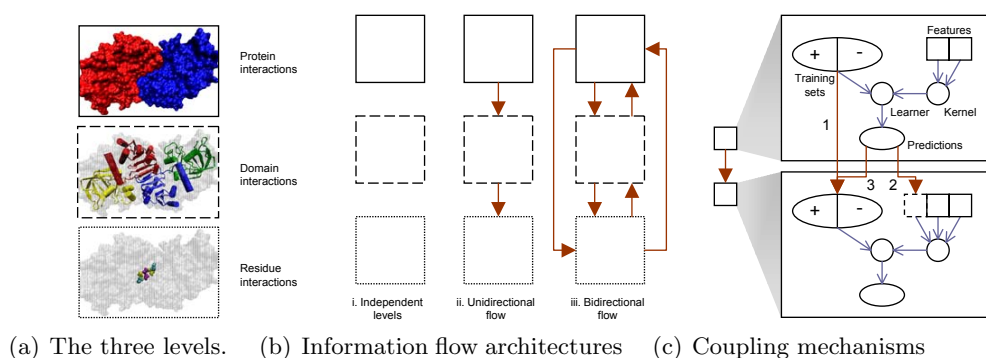
# Utilizing Problem Structures: Multi-level Learning

### 4.1 Introduction

In the previous chapter we described methods for predicting protein interactions, and how we improved prediction accuracy by training set expansion. While some of the methods could predict *which* proteins interact with high accuracy, they do not explain *how* the proteins interact. For instance, if protein A interacts with both proteins B and C, whether B and C could interact with A simultaneously remains unknown, as they may or may not compete for the same binding interface of A. This observation has led to the recent interest in refining PPI networks by structural information about domains [6, 16, 104]. It has also called for the prediction of protein interactions at finer granularities.

Since binding interfaces of proteins are enriched in conserved domains in permanent

interactions [31], it is possible to construct a second-level interaction network with protein interactions annotated by the corresponding domain interactions. An even finer third-level interaction network involves the residues mediating the interactions (Figure 4.1).



**Figure 4.1. Schematic illustration of multi-level learning concepts. (a) The three levels of interactions. Top: the PDB structure 1piw of the homo-dimer yeast NADP-dependent alcohol dehydrogenase 6. Middle: each chain contains two conserved Pfam domain instances, PF00107 (inner) and PF08240 (outer). The interaction interface is at PF00107. Bottom: two pairs of residues predicted by iPfam to interact: 283 (yellow) with 287 (cyan), and 285 (purple) with 285. Visualization by VMD [94]. (b) The three information flow architectures. i: independent levels, ii: unidirectional flow (illustrated by download flow), iii: bidirectional flow. (c) Coupling mechanisms for passing information from one level to another. 1: passing training information to expand the training set of the next level, 2: passing predictions as an additional feature of the next level, 3: passing predictions to expand the training set of the next level.**

As will be described in the next section, some recent studies have started to perform interaction predictions at the domain and residue levels. The data features used by each level are quite distinct. While protein level features are mostly from functional genomic and proteomic data such as gene expression and sub-cellular localization of whole genes and proteins, domain level features are mainly evolutionary information such as phylogenetic-

occurrence statistics of the domain families, and residue level features are largely structural or physical-chemical information derived from the primary sequences.

In the literature of domain-level prediction, the term “domain” is usually used to mean a domain family, which could have multiple occurrences in different proteins. In this study we use the terms “domain family” and “domain instance” to refer to these two concepts respectively, in order to make a clear distinction between them. For example, PF07974 is a domain family from Pfam, where ADP1\_YEAST.PF07974 is a domain instance in the protein ADP1\_YEAST.

Since the data features of the three levels describe very different aspects of the biological objects, potentially they could contribute to the prediction of different portions of the interaction networks. For example, some protein interactions could be difficult to detect using whole-protein level features since they lack fine-grained physical-chemical information. These can be supplemented by the residue level features such as charge complementarity.

Likewise, for the protein interactions that occur within protein complexes, there could be a high correlation between the expressions of the corresponding genes. With proper gene expression datasets included in the protein features, there is a good chance of correctly predicting such protein interactions. Then if one such interaction involves a pair of proteins each with only one conserved domain, it is very likely that the domain instances actually interact.

One may worry that if the predictions at a particular level are inaccurate, the errors would be propagated to the other levels and worsen their predictions. As we will discuss, this issue can be handled algorithmically by carefully deciding what information to propagate and how it is propagated. With a properly designed algorithm, combining the predictions

and utilizing the data features of all three levels can improve the predictions at each level.

In this work we propose a new multi-level machine-learning framework that combines the predictions at different levels. Since the framework is also potentially useful for other problems in computational biology that involve a hierarchy, such as biomedical text mining (a journal contains papers and a paper contains key terms), we start with a high-level description of multi-level learning and discuss three key aspects of it. Then we suggest a practical algorithm for the problem of predicting interactions at the protein, domain and residue levels, which integrates the information of all three levels to improve the overall accuracy. We demonstrate the power of this algorithm by showing the improvements it brings to the prediction of yeast interactions relative to the predictions from independent levels.

## 4.2 Related work

Two main ingredients of protein-protein interaction predictions are the selection of a suitable set of data features, and an appropriate way to integrate them into a learning method. Many kinds of features have been considered [199], including sub-cellular localization [92], gene expression [55, 176], and phylogenetic profiles [146]. With the many different kinds of data features, Bayesian approaches [98] and kernel methods [17, 21, 199] are natural choices for integrating them into a single learning algorithm. The former unifies the whole inference process by a probabilistic framework, while the latter encodes different kinds of data into kernel matrices that can be combined by various means [111].

Predictions of interactions between domain families are related to the more general goal of identifying protein interaction interfaces. While some studies tackle the problem using

features at the domain level only [100], most other work assumes that a set of protein-protein interactions are known a priori, and the goal is to predict either domain family interactions (i.e., which domain families have their instances interact in at least one pair of proteins) or domain-instance interactions (i.e., through which domain instances do proteins interact in known interactions) [3, 18, 23, 37, 48, 58, 78, 79, 82, 83, 96, 100, 113, 120, 127, 137, 138, 156, 161, 178, 190, 191]. The data features are mainly derived from statistics related to the parent proteins. For example, for a pair of domain families, the frequency of co-occurrence in interacting proteins is an informative feature, since a higher frequency may indicate a larger chance for them to be involved in mediating the interactions.

At a finer level, identifying protein interaction interfaces involves the prediction of residue interactions, which could be divided into two sub-tasks: 1) predicting which residues are in any interaction interfaces of a protein [41], and 2) predicting which of these interfaces interact [42]. Data features are mainly derived from the primary protein sequences or from crystal structures if they are assumed available. Docking algorithms [163] represent related approaches, but have a fundamentally different focus: Their goal is to utilize largely physical information to deduce the structure of the complex from the unbound protein structures, a considerably harder problem. Therefore, we do not consider them in this article and focus on large-scale techniques.

From a theoretical perspective, our multi-level learning framework is loosely related to co-training [22] and the meta-learning technique called stacking [195]. We will compare them with our framework after introducing the information flow architectures and the coupling mechanisms in Sections 4.4.1 and 4.4.2 respectively. Also, our framework by nature facilitates semi-supervised learning [34]. We will briefly discuss semi-supervised learning and its relationships with PSI-BLAST [7] in Section 4.4.2.

### 4.3 Problem definition

We now formally describe the learning problem we tackle in this study. The inputs of the problem consist of the following:

- **Objects:** a set of proteins, each containing the instances of one or more conserved domains, each of which contains some residues. Each protein, domain instance and residue is described by a vector of feature values. Some additional features are available for pairs of objects, such as the likelihood for a pair of proteins to interact according to a high-throughput experiment.
- **Gold standard<sup>1</sup> positive sets** of known protein-protein, domain instance-domain instance and residue-residue interactions. The positive sets could be 1) contaminated with false positives, and 2) incomplete, with false negatives, and a pair of upper-level objects in the positive set may not have any corresponding lower-level object pairs known to be in the positive sets.
- **Gold standard negative sets** of non-interactions at the three levels.

We assume no crystal structures are available except for the proteins in the gold-standard positive sets, so that the input features cannot be derived from known structures. This is a reasonable assumption given the small number of known structures as compared to the availability of other data features.

The objective is to use the gold standard sets and the data features to predict whether the object pairs outside the gold standard sets interact or not. Prediction accuracies are

---

<sup>1</sup>As in other studies on protein interaction networks, we use the term “gold standard set” to mean a set of sufficiently reliable data useful for the prediction purpose, instead of a ground-truth set that is absolutely correct.

estimated by cross-validation using holdout testing examples in the gold standard sets not involved in the training process.

In this study we focus on kernel methods [164] for learning from examples and making predictions. The main goal of this study is to explain how the predictions at the different levels can be integrated, and to demonstrate the resulting improvements in accuracy. We do not attempt to boost the accuracy at each individual level to the limit. It may be possible to improve our predictions by using other features, learning algorithms, and parameter values. As we will see, the design of our algorithm provides the flexibility for plugging in other state-of-the-art learning methods at each level. We expect that the more accurate the individual algorithms are, the more benefits they will bring to the overall accuracy through the multi-level framework.

## 4.4 Methods

In order to develop a method for predicting interactions at all three levels in a cohesive manner, we need to define the relationships between the levels, which is the topic of Section 4.4.1. We first describe two information flow architectures already considered in previous studies, and then propose a new architecture that maximally utilizes the available data. In Section 4.4.2 we discuss various possible approaches to coupling the levels, i.e., ways to pass information between levels. In Section 4.4.3 we discuss the data sparsity issue. In particular, we describe the idea of local modeling, which is also useful for network predictions in general. Finally, in Section 4.4.4 we outline the actual concrete algorithm that we have developed and used in our experiments.



#### 4.4.1 Information flow architectures

##### **Architecture 1: independent levels**

A traditional machine-learning algorithm learns patterns from one single set of training examples and predicts the class labels of one single set of testing instances. When there are three sets of examples and instances instead, the most straightforward way to learn from all three levels is to handle them separately and make independent predictions (Figure 4.1bi). We use this architecture to set up the baseline for evaluating the performance of the other two architectures.

##### **Architecture 2: unidirectional flow**

A second architecture is to allow downward (from protein to domain to residue) or upward (from residue to domain to protein) flow of information, but not both (Figure 4.1bii). This architecture is similar to some previous domain-level interaction methods described above, which also use information from the protein level. However, in our case protein interactions are not assumed to be known with certainty. So only the training set and the predictions made from the training set at the protein level can be used to assist the domain and residue levels.

##### **Architecture 3: bidirectional flow**

A third architecture is to allow the learning algorithm of each level to access the information of any other levels, upper or lower (Figure 4.1biii). By allowing both upward and downward flow of information, this new architecture is the most flexible among the three, and is the architecture that we explore in this study. Theoretically, this architecture is loosely related to co-training [22], which assumes the presence of two independent sets of

features, each of which is capable of predicting the class labels of a subset of data instances. Here we have three sets of features, each of which is capable of predicting a portion of the whole interaction network. Practical extensions to the ideal co-training model allow partially dependent feature sets and noisy training examples, which fit our current problem. Learning proceeds by iteratively building a classifier from one feature set, and adding the highly confident predictions as if they were gold-standard examples to train another classifier using the other feature set. The major difference between our bidirectional-flow architecture and co-training is the presence of a hierarchy between the levels in our case, so that each set of features makes predictions at a different granularity.

#### 4.4.2 Different approaches to coupling the levels

To design a concrete learning algorithm, we need to specify what information is to be passed between different levels and how it is passed. Here we suggest several possibilities, and briefly discuss the pros and cons of each of them.

##### **What information to pass**

###### *i. Training data*

One simple idea is to pass training data to other levels (Figure 4.1c, arrow 1). This can be useful in filling in the missing information at other levels. For example, many known protein interactions do not have the corresponding 3D structures available, so there is no information regarding which domain instances are involved in the interactions. The known protein interactions can be used to compute statistics for helping the prediction of domain-level interactions.

###### *ii. Training data and predictions*

The major limitation of passing only training data is that the usually much larger set of data instances not in the training sets (the “unlabeled data”) would not benefit from multi-level learning. In contrast, if the predictions made at a level are also passed to the other levels, much more data instances could benefit (Figure 4.1c, arrow 2 and 3). For instance, if two domain instances are not originally known to interact, but they are predicted to interact by the domain-level features with high confidence, this information directly implies the interaction of their parent proteins.

Algorithms adopting this idea are semi-supervised in nature [34], since they train on not only gold-standard examples, but also predictions of data instances that are originally unlabeled in the input data set. Note that the idea of semi-supervised learning has been explored in the bioinformatics literature. For instance, in the PSI-BLAST method [7], sequences that are highly similar to the query input are iteratively added as seeds to retrieve other relevant sequences. These added sequences can be viewed as unlabeled data, as they are not specified in the original query input.

### **How the information is passed**

#### *i. Combined optimization*

To pass information between levels, a first approach is to combine the learning problems of the different levels into a single optimization problem. The objective function could involve the training accuracies and smoothness requirements of all three levels. This approach enjoys the benefits of being mathematically rigorous, and being backed by the well-established theories of optimization. Yet the different kinds of data features at the different levels, as well as noisy and incomplete training sets, make it difficult to define a good objective function. Another drawback is the tight coupling of the three levels, so that

it is not easy to reuse existing state-of-the-art prediction algorithms for each level.

*ii. Predictions as additional features*

Another approach is to have a separate learning algorithm at each level, and use the predictions of a level as an additional feature of another level (Figure 4.1c, arrow 2). For example, if each pair of proteins is given a predicted probability of interaction, it can be used as the value of an additional feature 'parent proteins interacting' of the domain instance pairs and residue pairs. In this approach the different levels are loosely coupled, so that any suitable learners can be plugged into the three levels independently, and the coupling of the levels is controlled by a meta-algorithm.

A potential problem is the weighting of the additional features from other levels relative to the original ones. If the original set of features is large, adding one or two extra features without proper weighing would have negligible effects on the prediction process. Finding a suitable weight may require a costly external optimization or cross-validation procedure. For kernel methods, an additional challenge is integrating the predictions from other levels into the kernel matrix, which could be difficult as its positive semi-definiteness has to be conserved.

The idea of having a meta-algorithm that utilizes the predictions of various learners is also used in stacked generalization, or stacking [195]. It treats the predictions of multiple learners as a new set of features, and uses a meta-learner to learn from these predictions. However, in our setting, the additional features come from other levels instead of the same level.

*iii. Predictions as augmented training examples*

A similar approach is to add the predictions of a level to the training set of another

level (Figure 4.1c, arrow 3). The resulting training set involves the original input training instances and augmented training data from other levels, with a coefficient reflecting how much these augmented training data are to be trusted according to the training accuracy of the supplying level. This approach also has the three levels loosely coupled.

A potential problem of this training set expansion approach is the propagation of errors to other levels. The key to addressing this issue is to perform soft coupling, i.e., to associate confidence values to predictions, and propagate only highly confident predictions to other levels [203]. For kernel methods, this means ignoring objects falling in or close to the margin. This approach is similar to PSI-BLAST mentioned above, which selectively includes only the most similar sequences in the retrieval process.

In this study, we focus on this third approach. It requires a learning method for each level, while the control of information flow between the different levels by means of training set expansion forms the meta-algorithm. Since each level involves only one set of features and one set of data instances, traditional machine learning methods can be used. We chose support vector regression (SVR) [52], which is a type of kernel method. We used regression instead of the more popular support vector machine classifiers [26] because the former can accept confidence values of augmented training examples as inputs, and produce real numbers as output, which can be converted back into probabilities that reflect the confidence of interactions.

#### 4.4.3 Global vs. local modeling, and data sparsity issues

##### Global modeling

Taking a closer look at the prediction problem at each individual level, one would realize

that applying a traditional learning method is actually non-trivial since we are dealing with network data. In a traditional setting, each training instance has a class label and the job of a learning algorithm is to identify patterns in the feature values for predicting the class label of each unlabeled object. In our current situation, each data instance is a pair of biological objects (proteins/domain instances/residues), with two possible class labels: interacting and non-interacting. In order to construct a learner, one would need features for pairs of objects. A model can then be learned using a traditional machine learning method for all object pairs. We call this ‘global modeling’ since a single regression model is built for all the data instances. Global modeling has a number of major drawbacks:

1. Features for object pairs: it is not easy to construct features for pairs of objects, since most available data features are for single objects. This is particularly a problem for kernel methods, which require a kernel matrix to encapsulate the similarity between each pair of data instances. For network data, this means a similarity value for each pair of object pairs. While methods have been proposed to construct such kernel matrices [17], the resulting kernels, while formally correct, are difficult to interpret.
2. Time complexity: working with pairs of objects squares the time requirement with respect to the number of objects in the dataset. While state-of-the-art implementations of kernel methods could easily handle thousands of proteins, it would still be challenging to deal with millions of protein pairs, let alone the even more daunting numbers of domain instance pairs and residue pairs.
3. Space complexity: the kernel matrix has a size quadratic in the number of data instances. With  $n$  objects at a level, there are  $O(n^2)$  pairs and thus the kernel matrix contains  $O(n^4)$  entries.

4. Sub-clusters: the two classes of data instances may contain many sub-clusters that cannot be handled by one single global model [21, 203]. For instance, proteins involved in permanent complexes may use a very different interaction mechanism from transient interactions in signaling pathways.

### Local modeling

To avoid these problems, one alternative is local modeling [21], which we have described in the previous chapter. Briefly, instead of building one single global model for all object pairs, one local model is built for each object. For example, if the dataset contains  $n$  proteins, then  $n$  models are built, one for each protein, for predicting whether this protein interacts with each of the  $n$  proteins. The advantages of local modeling are obvious: 1) data features are needed for individual objects only, 2) the time complexity is smaller than global modeling whenever the learning method has a super-linear time complexity, 3) much less memory space is needed for the kernel matrix, and 4) each object can have its very specific local model. For all these benefits, in our experiments we only considered local modeling.

Local modeling is also not free from problems, but they are solvable. The most significant problem is data sparsity – some objects may have insufficient training examples (or none at all) for building local models. For example, among the millions of yeast protein pairs, there are only a few thousand known interactions, so many proteins have very few known interactions. An object with zero or few known interaction partners would not have enough training examples for building its local model.

Our proposed solution uses concepts related to semi-supervised learning: use high confidence predictions to augment training sets [203]. Suppose protein A has sufficient known positive and negative examples in the original training sets, and the local model

learned from these examples predicts with high confidence protein B to be an interaction partner with A. Then when building the local model for B, A can be used as a positive training example. Predicted non-interactions can be added as negative examples in a similar way.

This idea is consistent with the training set expansion method proposed above for inter-level communication. As a result, the information flow both between levels and within a level can be handled in a unified framework. The expanded training set of a level thus involves the input training data, highly confident predictions of the local models of the level, and highly confident predictions from other levels.

Practically, training set expansion within the same level requires an ordered construction of the local models. Objects with many (input or derived) training examples should have their local models constructed first, as more accurate models are likely to be obtained from larger training sets. As these objects are added as training examples of their predicted interaction partners and non-partners, they would progressively accumulate training examples for their own local models.

#### 4.4.4 The concrete algorithm

We now explain how we used the ideas described in the previous sections, namely bidirectional information flow, coupling by predictions passing, and local modeling with training set expansion, to develop our concrete learning algorithm for prediction of protein, domain instance and residue interactions. We first give a high-level overview of the algorithm, then explain the components in more detail.

The main steps of the algorithm are:



1. Set up a learning sequence of the levels.
2. Use the model learned for the first level in the sequence to predict interactions at the level.
3. Propagate the most confident predictions to the next level in the sequence as auxiliary training examples.
4. Repeat the previous two steps for the second and third levels, and so on.

#### **Learning at each level**

We use training set expansion with support vector regression (SVR) to perform learning at each level, similar to the idea in [203]. Each pair of objects in the positive and negative training sets is given a class label of 1 and 0, respectively. A SVR model is learned for the object (e.g. protein) with the largest number of training examples (denoted as A). The model predicts a real value for each object, indicating the likelihood that it interacts with A. The ones with the largest and smallest predicted values are treated as the most confident positive and negative predictions, and are used to expand the training set. For example, if B is an object with the largest predicted value, then A and B are predicted to interact, and A is added as an auxiliary positive training example of B. After training set expansion, the next object with the largest number of training examples is re-determined, its SVR is learned, and the most confident predictions are used to expand the training set in the same manner. The whole process then repeats until all models have been learned. Finally, each pair of objects A and B received two predicted values, one from the model learned for A and one from the model learned for B. The two values are weighted according to the training accuracies of A and B to produce the predicted value for the pair. Sorting the predicted values in descending order gives a list of predictions from the pair most likely to interact to

the one least likely. The list can then be used to evaluate the accuracy by metrics such as the area under the receiver operator characteristic curve (AUC) [88].

### **Setting up the learning sequence**

One way to set up the learning sequence is to use the above procedure to deduce the training accuracy of the three levels when treated independently, then order the three levels into a learning sequence according to their accuracies. For example, if the protein level gives the highest accuracy, followed by the domain level, and then the residue level, the sequence would be “PDRPDR...”, where P, D and R stand for the protein, domain and residue levels, respectively. Having the level with the highest training accuracy earlier in the sequence ensures the reliability of the initial predictions of the whole multi-level learning process, which is important since all latter levels depend on them. Notice that after learning at the last level, we feed back the predictions to the first level to start a new iteration of learning.

In our computational experiments we also tested the accuracy when only two levels are involved. In such situations, we simply bypassed the left-out level. For example, to test how much the domain and residue levels could help each other without the protein level, the learning sequence would be “DRDR...”.

### **Propagating predictions between levels**

The mechanism of propagating predictions from a level to another depends on the direction of information flow.

For an upward propagation ( $R \rightarrow D$ ,  $R \rightarrow P$  or  $D \rightarrow P$ ), each object pair in the next level receives a number of predicted values from its children at the previous level. For example, if predictions are propagated from the domain level to the protein level, each pair of domain

instances provides a predicted value to their pair of parent proteins. We tried two methods to integrate these values. In the first method, we normalize the predicted values to the  $[0, 1]$  range as a proxy of the probability of interaction, then use the noisy-OR function [190] to infer the chance that the parent objects interact. Let  $X$  and  $Y$  be the two sets of lower-level objects, and  $p(x, y)$  denotes the probability of interaction between two objects  $x \in X$  and  $y \in Y$ , then the chance that the two parent objects interact is  $1 - \prod_{x \in X, y \in Y} (1 - p(x, y))$ , i.e., the parent objects interact if and only if at least one pair of its children objects interact. In the second method, we simply take the maximum of the values. In the ideal case where all predicted values are either 0 or 1, both methods are exactly the same as taking the OR of the values. When the values are noisy, the former is more robust as it does not depend on a single value. Yet its value is dominantly affected by a large number of fuzzy predicted values with intermediate confidence, and is thus less sensitive. Since in our tests it does not provide superior performance, in the following we report results for the second method.

For a downward propagation ( $P \rightarrow D$ ,  $P \rightarrow R$  or  $D \rightarrow R$ ), we inherit the predicted value of the parent pair as the prior belief that the object pairs from the two parents will interact.

In both cases, after computing the probability of interaction for each pair of objects in the next level based on the predicted values at the current level, we again add the most confident positive and negative predictions as auxiliary training examples for the next level, with the probabilities used as the confidence values of these examples.

In the actual implementation, we used the Java package `libsvm` [33] for SVR, and the Java version of `lapack`<sup>2</sup> for some matrix manipulations.

---

<sup>2</sup><http://www.netlib.org/lapack/>

**Table 4.1. Data features at the protein level.**

Feature	Feature of	Data type	Kernel
COG (version 7) phylogenetic profiles [181]	Proteins	Binary vectors	RBF ( $\sigma = 8$ )
Sub-cellular localization [92]	Proteins	Binary vectors	Linear
Cell cycle gene expression [176]	Proteins	Real vectors	RBF ( $\sigma = 8$ )
Environment response gene expression [68]	Proteins	Real vectors	RBF ( $\sigma = 8$ )
Yeast two-hybrid [97, 185]	Protein pairs	Unweighted graph	Diffusion ( $\beta = 0.01$ )
TAP-MS [69, 108]	Protein pairs	Weighted graph	Diffusion ( $\beta = 0.01$ )

## 4.5 Experiments

We tested the effectiveness of multi-level learning by predicting protein, domain instance and residue interactions of the yeast *Saccharomyces cerevisiae*.

### 4.5.1 Data

#### Protein level

Data features were gathered from multiple sources (Table 4.1), including phylogenetic profiles, sub-cellular localization, gene expression, and yeast two-hybrid and TAP-MS networks. Each of them was turned into a kernel matrix and the final kernel was the summation of them, as in previous studies [21, 199, 203].

A gold standard positive set was constructed from the union of experimentally verified or structurally determined protein interactions from MIPS [131], DIP [160] and iPfam [63] with duplicates removed. The MIPS portion was based on the 18 May 2006 version, and only physical interactions not obtained from high throughput experiments were included. The DIP portion was based on the 7 Oct 2007 version, and only interactions from small-scale experiments or multiple experiments were included. The iPfam portion was based on version 21 of Pfam [64]. A total of 1681 proteins with all data features and at least one

**Table 4.2. Data features at the domain level. \*: These two features were used with the unidirectional and bidirectional flow architectures only since they involve information about the training set of the protein level.**

Feature	Feature of	Data type	Kernel
Phylogenetic tree correlations [76] of Pfam alignments	Domain family pairs	Real matrix	Empirical kernel map [183]
In all species, number of proteins containing an instance of the domain family	Domain families	Integers	Polynomial (d=3)
In all species, number of proteins containing domain instances only from the family	Domain families	Integers	Polynomial (d=3)
Number of domain instances of parent protein	Domain instances	Integers	Polynomial (d=3)
Fraction of non-yeast interacting protein pairs containing instances of the two domains respectively are mediated by the domain instances*	Domain family pairs	Real matrix	Constant shift embedding [157]
Fraction of protein pairs containing instances of the two domains respectively are known to be interacting in the PPI training set*	Domain family pairs	Real matrix	Constant shift embedding

interaction were included in the final dataset, forming 3201 interactions. A gold standard negative set with the same number of protein pairs was then created from random pairs of proteins not known to interact in the positive set [17, 37].

### Domain level

We included two types of features at the domain level: co-evolution and statistics related to parent proteins (Table 4.2). These are similar to the features used by previous studies for domain family/domain instance interaction predictions [100, 178].

The gold standard positive set was taken from iPfam, where two domain instances are defined as interacting if they are close enough in 3D structure and some of their residues are predicted to form bonding according to their distances and chemistry. After intersecting with the proteins considered at the protein level, a total of 422 domain instance interactions were included, which involves 272 protein interactions and 317 domain instances from 223 proteins and 252 domain families. A negative set with the same number of domain instance

**Table 4.3. Data features at the residue level.**

Feature	Feature of	Data type	Kernel
PSI-BLAST profiles	Residues and neighbors	Vectors of real vectors	Summation of linear
Predicted secondary structures	Residues and neighbors	Vectors of real vectors	Summation of linear
Predicted solvent accessible surface areas	Residues and neighbors	Vectors of real numbers	Summation of circular

pairs was then formed from random pairs of domain instances in the positive set. All known yeast Pfam domain instances of the proteins were involved in the learning, many of which do not have any known interactions in the gold standard positive set. Altogether 2389 domain instances from 1681 proteins and 1184 domain families were included.

### Residue level

We used three data features derived from sequences (Table 4.3). Charge complementarity and other features likely useful for interaction predictions are implicit in the sequence profiles. The features are similar to those used in a previous study [42]. However, as we do not assume the availability of crystal structures of unlabeled objects, the secondary structures and solvent accessible surface areas we used were algorithmically predicted from sequence instead of derived from structures. We used SABLE [1] to make such predictions.

In a previous study [42], the feature set of a residue involves not only the features of the residue itself, but also neighboring residues closest to it in the crystal structure, which allows for the possibility that some of them are involved in the same binding site and thus have dependent interactions. In the absence of crystal structures, we instead included a window of residues right before and after a residue in the primary sequence to construct its feature set. We chose a small window size of 5 to make sure that the included residues are physically close in the unknown 3D structures.

The gold standard positive set was taken from iPfam. Since there is a large number of residue pairs, we only sampled 2000 interactions, which involve 228 protein pairs, 327

domain instance pairs and 3053 residues from 195 proteins, 279 domain instances and 224 domain families. Only these 3053 residues were included in the data set. A negative set was created by randomly sampling from these residues 2000 residue pairs that do not have known interactions in iPfam.

### 4.5.2 Evaluation procedure

We used ten-fold cross validation to evaluate the performance of our algorithm. Since the objects in the three levels are correlated, an obvious performance gain would be obtained if in a certain fold the training set of a level contains some direct information about the testing set instances of another level. For example, if a residue interaction in the positive training set comes from a protein pair in the testing set, then the corresponding protein interaction can be directly inferred and thus the residue interaction would create a fake improvement for the predictions at the protein level. This problem was avoided by partitioning the object pairs in the three levels consistently. First, the known protein interactions in iPfam were divided into ten folds. Then, each domain instance interaction and each residue interaction was put into the fold in which the parent protein interaction was assigned. Finally, the remaining protein interactions and all the negative sets were randomly divided into ten folds.

Each time, one of the folds was held out as the testing set and the other nine folds were used for training. We used the area under the ROC (Receiver Operator Characteristics) curve (AUC) [88] to evaluate the prediction accuracies. For each level, all object pairs in the gold standard positive and negative sets were sorted in descending order of the predicted values of interaction they received when taking the role of testing instances. The possible values of AUC range from 0 to 1, where 1 corresponds to the ideal situation where all

positive examples are given a higher predicted value than all negative examples, and 0.5 is the expected value of a random ordering.

We compared the prediction accuracies in three cases: independent levels, unidirectional flow of training information only, and bidirectional flow of both training information and predictions. For the latter two cases, we compared the performance when different combinations of the three levels were involved in training.

For independent levels, we trained each level independently using its own training set, and then used the predictions as initial estimates to retrain for ten feedback iterations. This iterative procedure was to make sure that any accuracy improvements observed in the other architectures were at least in part due to the communications between the different levels, instead of merely the effect of semi-supervised learning at a single level. For unidirectional flow, we focused on downward flow of information. The levels were always arranged with upper levels coming before lower levels.

### 4.5.3 Results

Table 4.4 summarizes the prediction accuracies of the three levels. All numbers correspond to the average results among the ten feedback iterations. Each row represents the results of one level. For unidirectional flow and bidirectional flow, the levels involved in training are also listed. For example, the PR column of unidirectional flow involves the use of the protein-level training sets in setting up the initial estimate of the residue interactions. This has no effect on the predictions at the protein level since information only flows downward. The cell at the row for protein interactions is therefore left blank. The best result in each row is in bold face.



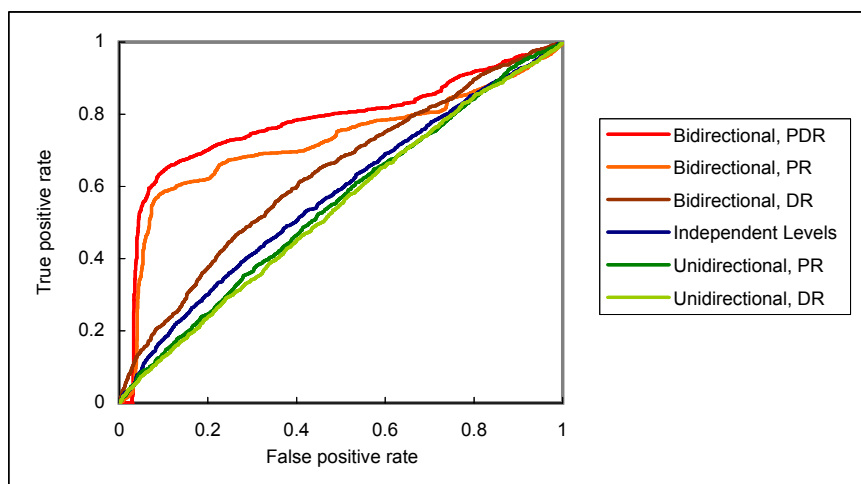
**Table 4.4. Prediction accuracies (AUC) of the three levels with different information flow architectures and training levels.**

Level	Independent levels	Unidirectional flow			Bidirectional flow			
		PD	PR	DR	PD	PR	DR	PDR
Proteins	0.7153				0.7205	0.7227		<b>0.7257</b>
Domains	0.5214	0.5854			<b>0.7015</b>		0.6796	0.6986
Residues	0.5675		0.5296	0.5128		0.6581	0.6182	<b>0.7361</b>

We first notice that the results for independent levels are consistent with our expectations. Having many diverse data features, the protein level has a satisfactory accuracy. On the other hand, the accuracies of the domain and residue levels were relatively low due to their weak and noisy features. Note that we are predicting whether two arbitrary domain instances or two arbitrary residues interact, rather than only those in known interacting protein pairs. This setting is more realistic for organisms with no known protein interaction network, and the problem is significantly harder than when the protein interaction network is available.

Downward flow of training information did help the prediction of domain instance interactions. However, the results of the residue level are quite unsatisfactory, with accuracies even lower than those with independent levels no matter assisted by the training examples of the protein level or domain level.

In contrast, the results for bidirectional flow are encouraging. In all cases, the accuracies are higher than the other two architectures. For example, while using the domain level to help the residue level decreased the accuracy of the latter from 0.5675 to 0.5128 with unidirectional flow, the accuracy was increased to 0.6182 with bidirectional flow. As an illustration of the difference in performance of the three architectures, the various ROC curves of residue interaction predictions are shown in Figure 4.2.



**Figure 4.2. Receiver operator characteristic (ROC) curves of residue interaction predictions with different frameworks and training levels.**

The improvements for both the domain and residue levels are quite dramatic, with maximum AUC gains of more than 0.15. This clearly shows the benefits of passing not only training information, but also highly confident predictions. Consider a domain instance pair in the testing set of a certain fold. Since the corresponding parent protein pair must not be in the training set at the protein level of the fold, the passing of training information does not directly help predict the interaction status of the domain instance pair. On the other hand, if the interaction status of the protein pair is predicted correctly with high confidence, passing this information to the domain level can make a direct influence on the prediction of the domain instance interaction. For instance, if the protein pair is correctly predicted as not interacting, the domain instance pair would probably be correctly predicted as not interacting, too.

In general, it is observed that levels with a higher raw accuracy with independent levels could offer a bigger improvement to the other levels. For example, the protein level

increased the accuracy of the residue interaction predictions from 0.5675 to 0.6581, while the domain level could only increase it to 0.6182.

However, it is also crucial to note that although the domain level has a low accuracy with independent levels, it could still make good improvements to the prediction of residue interactions. This observation supports our design of passing only highly confident predictions in avoiding the propagation of errors.

The combination of all three levels has the potential to further improve accuracy. For both the protein and residue levels, the best results were obtained when all three levels were involved in training. In particular, while each of the protein and domain levels improved the residue level by a certain amount, the combination of them provided yet another significant amount of improvement.

## 4.6 Discussion

The experimental results have demonstrated the great potential of linking up the prediction problems at the different levels. This initial success encourages deeper investigations of the idea along various directions.

Algorithmically, other approaches to combining the different levels, including combined optimization and predictions as extra features, need to be studied. Currently the features at the domain and residue levels are weak, as reflected by their low accuracy when learned independently, and the small improvement they could cause to the protein level. It is interesting to study ways to improve the predictions at these two levels, and more directly extract the complementary information hidden in these levels that are useful for the protein

level.

To predict the whole interaction network, we need to reduce the time and space requirements. One possible way is to intelligently select what data to exclude, such as residues that are predicted to be buried deep inside the core of a protein. Another idea is to group objects into interaction groups so that each cluster can be handled independently.

More insights could be gained by studying some theoretical aspects of multi-level learning, such as the hierarchical structure of the prediction problems, and the issue of noisy and incomplete training sets. With multiple levels, performance evaluation is very tricky. As we discussed, careless definitions of training and testing sets could produce biases to the resulting performance. It is instrumental to study the optimal way of evaluation.

Biologically, there are many interesting follow-up questions to be studied. The intricate interactions between the different levels are not yet clear, and could form a larger study of how the predictions change after receiving information from the other levels. One could compare different kinds of data features at the three levels and identify the ones with the greatest complementary effects. Another direction is to choose different kinds of residue samples (e.g., only charged residues) and inspect the relative improvements they provide to the protein and domain levels, to determine the residues that are more significant in a protein interaction.

## Chapter 5

# Handling Errors in Data: Consistent Prediction of Interactions at Different Levels

### 5.1 Introduction

In the previous chapter we study the interactions between proteins by analyzing the corresponding interactions at the domain and residue levels. In this chapter we concentrate on the protein and domain levels, and study a topic that is both interesting within the context of multi-level learning and in machine learning in general: handling errors in training sets. In particular, we study how errors at the protein level would affect the inference at the domain level, and how we can handle such errors to improve prediction accuracy.

It has been shown that the binding interfaces in obligatory interactions are more con-

served than the rest of the protein surface [31]. Various groups have thus attempted to explain protein interactions by the corresponding interactions of their conserved regions [171], with the conserved regions defined in domain or motif databases [64, 93, 95, 119, 150, 152, 194], or directly discovered from the proteins [11, 41, 42]. For simplicity, in the following we will use the term “domain” to mean all different types of conserved subsequences.

Different methods have been used to infer the domain-domain interactions (DDI) between interacting proteins. The association methods [3, 58, 127, 137, 178] look for domains that could distinguish interacting proteins from non-interacting ones, for example by computing the fraction of protein pairs that interact among all protein pairs that a domain pair occurs. The idea is extended by the lowest p-value method [138], which performs statistical tests with a null hypothesis that a domain pair does not affect a protein interaction. A separate study using statistical tests demonstrates the importance of negative sets [83]. Maximum likelihood methods search for a set of interacting domains (or domain occurrences) that could maximize the likelihood of the observed protein interactions [48, 78, 79, 161, 190]. Variations of the maximum likelihood methods involve the integration of information from multiple species [113, 120], and the use of likelihood difference before and after disallowing a pair of domains to interact as an assessment of its chance of mediating a protein interaction [156, 191]. In some studies, co-evolution signals are used to predict [100] and analyze [201] domain interactions. There are also methods that use maximum parsimony through probabilistic linear programming [82], message passing formalism [96] and standard machine learning methods such as SVM [23] and random decision forest [37] to predict domain interactions. Methods have also been proposed to elucidate domain interactions within complexes instead of simple binary interactions [18].

Most of these methods require as input a protein-protein interaction (PPI) network,

either from high-throughput assays such as Y2H or TAP-MS, or from large-scale databases of protein interactions [4, 10, 29, 103, 131, 148, 160, 211]. The accuracy of the inferred domain interactions depends on the quality of the input PPI network. False positives and false negatives in the network could lead to biased or wrong inference of domain interactions. In the extreme case, such errors could result in a mathematical system with no solutions [82, 96]. For example, if a pair of domains simultaneously occurs in a pair of interacting proteins with no other domains, and in a pair of non-interacting proteins, then the two protein pairs would draw contradictory conclusions regarding whether the domains interact.

In the previous studies, errors at the protein level have been dealt with passively during the inference of domain interactions. The probabilistic model in Wang et al. [190] allows a domain occurrence to be inactive, and a protein interaction to be explained by a “spurious binding” variable unrelated to domain interactions, to handle false negatives and false positives in the input protein network, respectively. Their later work also retain the concept of spurious protein interactions [191]. The linear programming method in Guimarães et al. [82] randomizes the linear constraints, so that in each problem instance only a subset of the protein interactions need to be explained. The model in Iqbal et al. [96] softens the Heaviside  $\theta$ -functions, so that protein interactions not explained by any domain interactions still have a non-zero likelihood equal to the value of a parameter  $\epsilon$ .

None of the above work studies extensively the effect of errors at the input PPI network to the inference of DDI. Also, none of the algorithms actively avoids errors at the protein level. In this study we first show the performance degradation caused by false positives and false negatives in the PPI network, and then propose methods for actively handling the errors.

## 5.2 How is DDI inference affected by noise in PPI network?

To study the effect of noise in the PPI network to the inference of DDI, we create noisy networks by first obtaining a high-confidence network with a low expected noise level, then introducing different amounts of controlled errors. For each resulting network, we apply a number of DDI prediction methods and observe their performance change with respect to the noise level.

**PPI network:** We take the BioGRID-10 dataset [203] as the high-confident gold-standard PPI network, which consists of all yeast protein interactions in BioGRID [29] reported by studies that report no more than 10 interactions. The threshold ensures a good coverage of PPIs while keeping false positives at a low level. Keeping only proteins with at least one Pfam-A [64] domain and all data features (see Section 5.3.2), the network contains 3,543 interactions between 1,677 proteins.

**Controlled errors:** We generate false negatives by randomly removing interactions from the positive set. False positives are then generated by adding random pairs of proteins residing in different cellular compartments [92], which are unlikely to interact [98].

**Inferring DDI from PPI:** We test several methods that infer DDI from a given PPI network.

The basic expectation maximization (EM) method [48] defines the likelihood of the observed PPI network based on the probabilities of interaction of the corresponding domain pairs. An EM algorithm is used to search for the interaction status of domain pairs in order to reach a local maximum of the likelihood function.

The InSite method [191] defines a probabilistic model for computing the probability



of the observed PPI network based on the probabilities of interaction of the corresponding domain pairs. The method caters for possible unexplained PPIs due to noise by adding in a spurious binding variable so that models with unexplained PPIs still have non-zero likelihood. Also, instead of using the estimated binding affinities as a direct indicator of how likely two domains interact, the method tests the likelihood difference when the domains are not allowed to interact. A large likelihood drop would indicate that the interaction of the domains is crucial for the PPI.

The message passing method [96] uses belief propagation (BP) to identify potential interacting domains. Again, some probabilities are assigned to the likelihood of each PPI that cannot be explained by DDIs.

**Performance evaluation:** Ideally, the performance of DDI inference methods should be evaluated by known DDIs. However, since there are relatively few known DDIs, we also use the accuracy of inferred PPIs as an auxiliary measure.

The gold-standard DDIs are taken from the crystal structures in iPfam [63]. It contains 337 interactions between 252 domain families. For comparison purpose, we also construct a set of non-interacting domain families, formed by the same number of random domain family pairs not known to interact.

We use the area under the receiver operator characteristic (ROC) curve [88], AUC, as the performance metric. Since the input PPI network contains no direct information about domain interactions, we simply use the whole PPI network to predict DDIs, and compare the predictions against the gold-standard.

The gold-standard PPIs are taken from the BioGRID-10 dataset, and a negative set with the same number of protein pairs is generated from protein pairs in different cellular

compartments. This negative set is generated independent of the false positive examples.

We use 10-fold cross-validation to evaluate the performance of the methods in inferring back the PPI network. For the BioGRID-10 PPI network without controlled errors, we hide one-tenth of the gold-standard positives and negatives as the holdout testing set. The resulting partial PPI network is sent to a DDI prediction method. For InSite, PPI probabilities are provided as the program output. For EM and BP, we use the noisy-OR function [190] to back-infer the probability of interaction for each pair of proteins  $i$  and  $j$  based on the domain pairs  $\mathcal{D}(i, j)$  that occur between them:

$$Pr(i, j) = 1 - \prod_{(m,n) \in \mathcal{D}(i,j)} (1 - \lambda_{mn}), \quad (5.1)$$

where  $\lambda_{mn} = Pr(m, n)$  is the probability for domains  $m$  and  $n$  to interact.

The function assumes independence between DDIs, so that the probability for proteins  $i$  and  $j$  to interact is equal to one minus the probability that none of their domain pairs interacts.

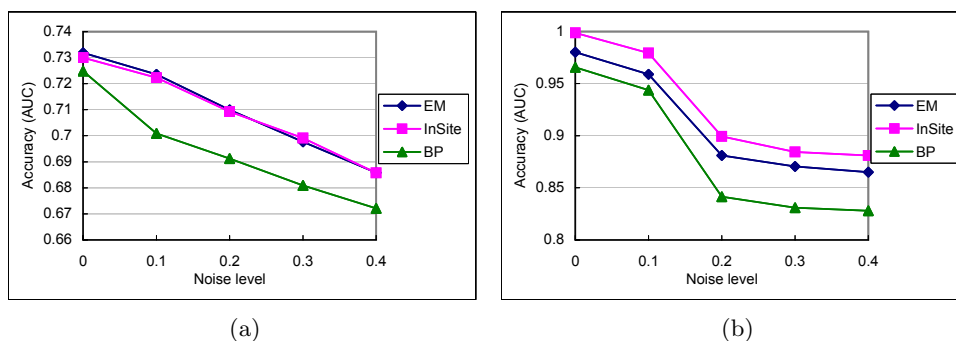
The PPI probabilities of the different folds are combined to compute the AUC value of the predicted PPI network.

For the networks with controlled errors, the error-containing training sets are again cut into ten folds. Each time nine folds are used for training and inferring PPI probabilities. However, instead of evaluating the accuracy against the error-containing holdout set, the AUC value is computed based on gold-standard positives and negatives not in the training fold. In other words, in all experiments, the predictions are evaluated against the same

gold-standard PPIs in the original BioGRID-10 dataset and the fixed set of gold-standard negatives.

### Results:

Figure 5.1 shows the accuracy of the three methods as the noise level increases.



**Figure 5.1. Accuracy of the three methods as the noise level increases. (a) Accuracy of PPI prediction. (b) Accuracy of DDI prediction.**

It is clearly seen that the prediction accuracies at both the protein and domain levels drop as the noise level increases. The accuracies at noise level 0.4 are substantially lower than when the datasets contain no controlled errors. As the false positive and negative rates of high-throughput PPI networks can be as high as 25%-45% and 75%-90% respectively [90], while small-scale experiments have covered only a small portion of the real interactions (approximately 20% for the BioGrid-10 dataset), the prediction of PPI and DDI networks could be seriously hampered by the noise in the input PPI network based on the results above.

We notice that although InSite and BP have built-in mechanisms for handling false positives and false negatives in the input PPI network, and EM also has a global estimate

of the false positive and false negative rates in the input PPI network, their performance is still substantially affected by noise. This observation motivates the development of new methods for handling errors in the PPI network.

### 5.3 Handling errors in the PPI network

It is not easy to detect errors in the input PPI network using only the information of domain occurrence at different proteins, as the numbers of domain pairs between protein pairs are usually large enough to explain erroneous PPI networks without causing contradictions. On the other hand, some protein features are very useful in identifying errors. For instance, if the expression patterns of two genes are highly correlated, and they co-occur in the same subset of genomes, they are likely to interact. It would thus be dubious to find the pair in the negative training set. As such, it would be useful to incorporate protein-level features in the process of DDI inference.

We propose a method to extend the EM algorithm to use protein level features. Before explaining the method, we first describe the original EM algorithm in more detail.

#### 5.3.1 The original EM algorithm

The original EM algorithm [48] takes as input a set  $T^+$  of observed interacting protein pairs and a set  $T^-$  of observed non-interacting protein pairs. The two sets together constitute the observed protein pairs  $T = T^+ \cup T^-$ . Each observation is subject to noise: an interacting pair can be in the negative set with a certain false negative rate, and a non-interacting pair can be in the positive set with a certain false positive rate. Let  $O_{ij}$  be a

Boolean variable denoting whether the protein pair  $(i, j)$  is in the positive set ( $O_{ij} = 1$ ) or the negative set ( $O_{ij} = 0$ ), and  $P_{ij}$  be the actual interaction status, then  $O_{ij}$  and  $P_{ij}$  are related by the false positive rate  $fp$  and false negative rate  $fn$  as follows:

$$Pr(O_{ij} = 1) = Pr(P_{ij} = 1)(1 - fn) + Pr(P_{ij} = 0)fp \quad (5.2)$$

Each pair of proteins  $(i, j)$  contains a corresponding set of domain pairs  $\mathcal{D}(i, j)$ . Let Boolean variable  $D_{mn}$  denotes whether domains  $m$  and  $n$  interact, and  $\lambda_{mn} = Pr(D_{mn} = 1)$  be the probability for the domains to interact. Then  $P_{ij}$  and  $D_{mn}$  are related by the noisy OR function stated in Equation 5.1.

The false positive rate  $fp$  is common to all protein pairs  $(i, j)$ , and is estimated as follows:

$$\begin{aligned} fp &= Pr(O_{ij} = 1 | P_{ij} = 0) \\ &= \frac{Pr(O_{ij} = 1, P_{ij} = 0)}{Pr(P_{ij} = 0)} \\ &\leq \frac{Pr(O_{ij} = 1)}{Pr(P_{ij} = 0)} \\ &= \frac{|T^+|}{\text{total no. of protein pairs} - \text{no. of real interaction pairs}}, \end{aligned} \quad (5.3)$$

where the total number of protein pairs is  $\binom{1667}{2}$  in our case and we approximate the number of real interaction pairs as 15,000 based on a recent large-scale experimental study [207].

The false negative rate  $fn$  is again common to all protein pairs  $(i, j)$ , and is estimated

as follows:

$$\begin{aligned}
 fn &= Pr(O_{ij} = 0 | P_{ij} = 1) & (5.4) \\
 &= 1 - \frac{Pr(P_{ij} = 1, O_{ij} = 1)}{Pr(P_{ij} = 1)} \\
 &\geq 1 - \frac{Pr(O_{ij} = 1)}{Pr(P_{ij} = 1)} \\
 &= 1 - \frac{|T^+|}{\text{no. of real interaction pairs}}
 \end{aligned}$$

The goal is to find the values of  $\lambda_{mn}$  in the parameter vector  $\lambda$  such that the likelihood of the observed data is maximized:

$$L(\lambda) = \prod_{(i,j) \in T} [Pr(O_{ij} = 1 | \lambda)]^{O_{ij}} [1 - Pr(O_{ij} = 1 | \lambda)]^{1 - O_{ij}} \quad (5.5)$$

The values of  $\lambda_{mn}$  are estimated by the EM algorithm. The observed data consist of the pairs in the positive and negative sets  $\{O_{ij}\} = T$ . The hidden data consist of the variables  $D_{mn}^{ij}$ , which denote whether domains  $m$  and  $n$  interact between proteins  $i$  and  $j$ . The actual protein interaction status  $P_{ij}$  is also hidden, but it is fully determined by  $D_{mn}^{ij}$ :  $P_{ij} = \vee_{(m,n) \in \mathcal{D}(i,j)} D_{mn}$ , and thus does not need to be separately handled. The observed and hidden data together form the complete data.

The EM algorithm starts by having an initial estimate of the parameters  $\lambda_{mn}$ , by counting the fraction of times domains  $m$  and  $n$  participate in interacting protein pairs:

$$\lambda_{mn}^{(0)} = \frac{|\mathcal{P}(m, n) \cap T^+|}{|\mathcal{P}(m, n)|}, \quad (5.6)$$

where  $\mathcal{P}(m, n)$  denotes the set of protein pairs that contain  $m$  and  $n$  respectively.

In the E-step, the expectation of the hidden data is determined based on the previous estimation of the parameter values:

$$\begin{aligned} E[D_{mn}^{ij} | O_{ab}, \forall a, b, \lambda^{(t-1)}] &= Pr(D_{mn}^{ij} = 1 | O_{ab}, \forall a, b, \lambda^{(t-1)}) & (5.7) \\ &= Pr(D_{mn}^{ij} = 1 | O_{ij}, \lambda^{(t-1)}) \\ &= \frac{Pr(O_{ij} | D_{mn}^{ij} = 1, \lambda^{(t-1)}) Pr(D_{mn}^{ij} = 1 | \lambda^{(t-1)})}{Pr(O_{ij} | \lambda^{(t-1)})} \\ &= \frac{Pr(O_{ij} | D_{mn}^{ij} = 1) Pr(D_{mn}^{ij} = 1 | \lambda^{(t-1)})}{Pr(O_{ij} | \lambda^{(t-1)})} \\ &= \frac{(1 - fn)^{O_{ij}} fn^{1-O_{ij}} \lambda_{mn}^{(t-1)}}{Pr(O_{ij} | \lambda^{(t-1)})}, \end{aligned}$$

where the denominator can be computed from Equations 5.1 and 5.2.

The M-step finds the maximum likelihood estimator (MLE) of the parameters of the expected log-likelihood of the complete data over all possible values of  $D_{mn}^{ij}$ . Denote  $D^{ij} = \{D_{mn}^{ij} : (m, n) \in \mathcal{D}(i, j)\}$  as the set of domain pair variables for protein pair  $(i, j)$ ,  $D = \{D_{mn}^{ij}\}$  as the whole set of domain pair variables, and  $O = \{O_{ij}\}$  as the whole set of protein interaction variables, the expected log-likelihood is expressed as

$$\begin{aligned}
 E_d[\ln L] &= E_d[\ln Pr(O, D|\lambda)] & (5.8) \\
 &= E_d[\ln Pr(O|D, \lambda)Pr(D|\lambda)] \\
 &= E_d[\ln \prod_{(i,j) \in T} Pr(O_{ij}|D^{ij})Pr(D^{ij}|\lambda)] \\
 &= E_d[\sum_{(i,j) \in T} \ln Pr(O_{ij}|D^{ij})Pr(D^{ij}|\lambda)] \\
 &= \sum_{(i,j) \in T} E_d[\ln Pr(O_{ij}|D^{ij}) + \ln Pr(D^{ij}|\lambda)],
 \end{aligned}$$

where the expectation is taken over all possible values of  $D$ .

To find the MLE, we differentiate the expected log-likelihood with respect to each parameter  $\lambda_{mn}$ . Since the first log term is independent of all  $\lambda_{mn}$  and for the second log term, only  $Pr(D_{mn}^{ij})$  depends on it, we have

$$\begin{aligned}
 \frac{\partial E_d[\ln L]}{\partial \lambda_{mn}} &= \frac{\partial \sum_{(i,j) \in \mathcal{P}(m,n)} E_d[\ln Pr(D_{mn}^{ij}|\lambda)]}{\partial \lambda_{mn}} & (5.9) \\
 &= \frac{\partial \sum_{(i,j) \in \mathcal{P}(m,n)} [prob \ln Pr(D_{mn}^{ij} = 1|\lambda_{mn}) + (1 - prob) \ln Pr(D_{mn}^{ij} = 0|\lambda_{mn})]}{\partial \lambda_{mn}} \\
 &= \frac{\partial \sum_{(i,j) \in \mathcal{P}(m,n)} [prob \ln \lambda_{mn} + (1 - prob) \ln(1 - \lambda_{mn})]}{\partial \lambda_{mn}} \\
 &= \sum_{(i,j) \in \mathcal{P}(m,n)} \left( \frac{prob}{\lambda_{mn}} - \frac{1 - prob}{1 - \lambda_{mn}} \right),
 \end{aligned}$$

where  $prob$  is the probability  $Pr(D_{mn}^{ij} = 1|O_{ab}, \forall a, b, \lambda^{(t-1)})$  we obtained in the E-step.

Now, setting the equation to zero, we have



$$\begin{aligned}
 \sum_{(i,j) \in \mathcal{P}(m,n)} \left( \frac{prob}{\lambda_{mn}} - \frac{1-prob}{1-\lambda_{mn}} \right) &= 0 & (5.10) \\
 \Rightarrow \sum_{(i,j) \in \mathcal{P}(m,n)} (prob - \lambda_{mn}) &= 0 \\
 \Rightarrow \lambda_{mn} &= \frac{\sum_{(i,j) \in \mathcal{P}(m,n)} prob}{\sum_{(i,j) \in \mathcal{P}(m,n)} 1} \\
 \Rightarrow \lambda_{mn} &= \frac{\sum_{(i,j) \in \mathcal{P}(m,n)} prob}{|\mathcal{P}(m,n)|},
 \end{aligned}$$

which is the fraction of protein pairs containing the domain pair that interacts.

The two steps are repeated until convergence, and the final probability for a pair of domains  $(m, n)$  to interact is read off from the final estimate of  $\lambda_{mn}$ .

### 5.3.2 Incorporating protein features

We modify the EM algorithm by incorporating protein features. Instead of treating protein pairs in the training sets  $T^+$  and  $T^-$  as ground truth, we use them only to estimate the likelihood of protein features.

Let  $F_{ij}$  be the vector of feature values for protein pair  $(i, j)$ , we define the new likelihood as follows:

$$\begin{aligned}
 L(\lambda) &= \prod_{(i,j) \in T} Pr(F_{ij} | \lambda) & (5.11) \\
 &= \prod_{(i,j) \in T} [Pr(F_{ij}, O_{ij} = 1 | \lambda) + Pr(F_{ij}, O_{ij} = 0 | \lambda)] \\
 &= \prod_{(i,j) \in T} [Pr(F_{ij} | O_{ij} = 1) Pr(O_{ij} = 1 | \lambda) + Pr(F_{ij} | O_{ij} = 0) Pr(O_{ij} = 0 | \lambda)]
 \end{aligned}$$

In the E-step, we need to determine the expectation of the missing data  $D_{ij}^{mn}$ :

$$\begin{aligned}
 & E[D_{mn}^{ij}|F_{ab}, \forall a, b, \lambda^{(t-1)}] & (5.12) \\
 = & Pr(D_{mn}^{ij} = 1|F_{ab}, \forall a, b, \lambda^{(t-1)}) \\
 = & Pr(D_{mn}^{ij} = 1|F_{ij}, \lambda^{(t-1)}) \\
 = & Pr(D_{mn}^{ij} = 1, O_{ij} = 1|F_{ij}, \lambda^{(t-1)}) + Pr(D_{mn}^{ij} = 1, O_{ij} = 0|F_{ij}, \lambda^{(t-1)}) \\
 = & Pr(D_{mn}^{ij} = 1|O_{ij} = 1, \lambda^{(t-1)})Pr(O_{ij} = 1|F_{ij}, \lambda^{(t-1)}) + \\
 & Pr(D_{mn}^{ij} = 1|O_{ij} = 0, \lambda^{(t-1)})Pr(O_{ij} = 0|F_{ij}, \lambda^{(t-1)}) \\
 = & \frac{Pr(O_{ij} = 1|D_{mn}^{ij} = 1)Pr(D_{mn}^{ij} = 1|\lambda^{(t-1)})Pr(F_{ij}|O_{ij} = 1)}{Pr(F_{ij}|\lambda^{(t-1)})} + \\
 & \frac{Pr(O_{ij} = 0|D_{mn}^{ij} = 1)Pr(D_{mn}^{ij} = 1|\lambda^{(t-1)})Pr(F_{ij}|O_{ij} = 0)}{Pr(F_{ij}|\lambda^{(t-1)})} \\
 = & \frac{\lambda_{mn}^{(t-1)}}{Pr(F_{ij}|\lambda^{(t-1)})} [(1 - fn)Pr(F_{ij}|O_{ij} = 1) + fnPr(F_{ij}|O_{ij} = 0)] \\
 = & \frac{\lambda_{mn}^{(t-1)} [(1 - fn)Pr(F_{ij}|O_{ij} = 1) + fnPr(F_{ij}|O_{ij} = 0)]}{Pr(O_{ij} = 1|\lambda^{(t-1)})Pr(F_{ij}|O_{ij} = 1) + Pr(O_{ij} = 0|\lambda^{(t-1)})Pr(F_{ij}|O_{ij} = 0)},
 \end{aligned}$$

We use a simple model to estimate the likelihood  $Pr(F_{ij}|O_{ij})$ . We assume the features are independent given the real observation, as in a Naive Bayes classifier. The likelihood is estimated by the MLE of the Gaussian given the training data:

$$\begin{aligned}
 Pr(F_{ij}|O_{ij} = o_{ij}) &= \prod_k Pr(F_{ijk}|O_{ij} = o_{ij}) & (5.13) \\
 &= \prod_k \frac{1}{\sigma_{ko_{ij}}(2\pi)^{1/2}} \exp\left[-\frac{1}{2}\left(\frac{F_{ijk} - \mu_{ko_{ij}}}{\sigma_{ko_{ij}}}\right)^2\right]
 \end{aligned}$$

where  $k$  is the feature index,  $\mu_{ko_{ij}}$  and  $\sigma_{ko_{ij}}$  are the sample mean and standard deviation of the values of feature  $k$  of the training data of class  $o_{ij}$ :

$$\mu_{k1} = \frac{\sum_{(a,b) \in T^+} F_{abk}}{|T^+|} \quad (5.14)$$

$$\mu_{k0} = \frac{\sum_{(a,b) \in T^-} F_{abk}}{|T^-|} \quad (5.15)$$

$$\sigma_{k1} = \left[ \frac{\sum_{(a,b) \in T^+} (F_{abk} - \mu_{k1})^2}{|T^+|} \right]^{1/2} \quad (5.16)$$

$$\sigma_{k0} = \left[ \frac{\sum_{(a,b) \in T^-} (F_{abk} - \mu_{k0})^2}{|T^-|} \right]^{1/2} \quad (5.17)$$

Estimating the likelihood by the MLE Gaussian results in a posterior classifier  $\frac{Pr(O_{ij}=o_{ij}|F_{ij}=1)}{Pr(O_{ij}=o_{ij}|F_{ij}=0)}$  equivalent to one obtained by logistic regression [133].

In the M-step, we re-estimate the parameters  $\lambda_{mn}$ . The expected log-likelihood is defined as follows:

$$\begin{aligned} E_d[\ln L] &= E_d[\ln Pr(F, O, D|\lambda)] \quad (5.18) \\ &= E_d[\ln Pr(F|O)Pr(O|D)Pr(D|\lambda)] \\ &= E_d[\ln \prod_{(i,j) \in T} Pr(F_{ij}|O_{ij})Pr(O_{ij}|D^{ij})Pr(D^{ij}|\lambda)] \\ &= E_d\left[ \sum_{(i,j) \in T} \ln Pr(F_{ij}|O_{ij})Pr(O_{ij}|D^{ij})Pr(D^{ij}|\lambda) \right] \\ &= \sum_{(i,j) \in T} E_d[\ln Pr(F_{ij}|O^{ij}) + \ln Pr(O_{ij}|D^{ij}) + \ln Pr(D^{ij}|\lambda)] \end{aligned}$$

Again, only the term  $\ln Pr(D^{ij}|\lambda)$  depends on  $\lambda$ , so the MLE for each  $\lambda_{mn}$  remains the same as in Equation 5.10.

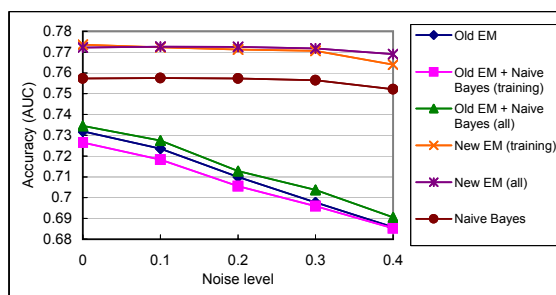
A variation of this method is to consider not only protein pairs in the training set  $T$ , but all pairs of proteins. On the positive side, this approach utilizes also the features of protein pairs outside the training set, which could potentially discover more domain interactions. On the negative side, if the protein-level predictions are not too accurate, this approach might introduce more noise to the DDI inference. We study the performance of the new EM algorithm and this variation empirically in the next section.

### 5.3.3 Empirical study

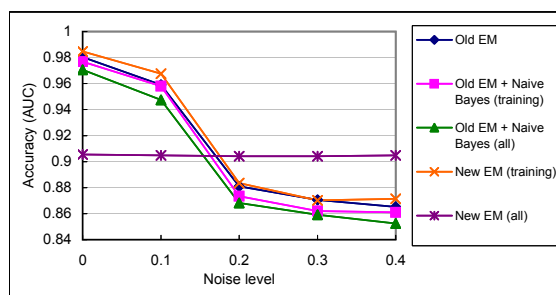
We use the new EM algorithm and its variation to predict DDI and PPI as before, and compare the results with those of the other methods. Protein features include phylogenetic profiles, gene expression data and high-throughput data we used in the training set expansion study. Cell localization is not included as it is used to define the gold-standard negative set. We take the protein kernels constructed in the training set expansion study, and use their elements as the feature values of protein pairs.

We would also want to study if any potential performance gain of the new algorithms can be trivially achieved by first running a protein-level classifier to predict PPI from the training set, and then use the results to infer DDI. To this end, we also include an approach that uses a Naive Bayes classifier to predict protein interactions, and then use the predicted probabilities to initialize the value of  $\lambda_{mn}$  in the original EM method. Again, we have two variations here, one uses only the predicted probabilities of the protein pairs in the training set  $T$ , and the other uses all predicted probabilities.

Figure 5.2 shows the prediction accuracy of the various methods.



(a)



(b)

**Figure 5.2. Comparing the accuracy of the new methods with the original EM algorithm. (a) Accuracy of PPI prediction. (b) Accuracy of DDI prediction. Old EM: the original EM algorithm by Deng et al. Old EM + Naive Bayes (training): the original EM algorithm, with initial parameter values estimated by the Naive Bayes predictions of the protein pairs in the training set. Old EM + Naive Bayes (all): the original EM algorithm, with initial parameter values estimated by the Naive Bayes predictions of all protein pairs. New EM (training): the new EM algorithm, with variables defined for protein pairs in the training set only. New EM (all): the new EM algorithm, with variables defined for all protein pairs. Naive Bayes: the Naive Bayes predictions.**

The figure shows that the new EM algorithm predicts protein interactions with a higher accuracy than the original EM algorithm when the training set is error free. It is also much less sensitive to errors in the training set. The performance gain is not only due to a more accurate PPI network input, as initializing the original EM algorithm with Naive

Bayes predictions results in only a small accuracy improvement. Furthermore, the new EM algorithm is more accurate than Naive Bayes alone in predicting protein interactions. This result suggests that the new EM algorithm is able to utilize the information from both the protein and domain levels to make more accurate predictions.

The two variations have very similar performance, with the one considering all protein pairs having slightly higher accuracy at high noise level.

The DDI performance is intriguing. When only protein pairs in the training set are considered, the new EM algorithm is slightly more accurate than the original EM algorithm, but is still very sensitive to noise. However, when all protein pairs are considered, the new EM algorithm has a very stable accuracy regardless of the noise level. It appears that by considering all protein pairs, this approach is dominantly affected by the initial likelihood estimations that are based on information at the protein level only.

This result suggests that using only information at the protein level to estimate feature likelihood could make it difficult to predict some domain interactions. We would want to devise a method to estimate feature likelihood using also information at the domain level.

#### 5.3.4 Constrained likelihood estimation

Intuitively, if two protein pairs share a large number of common domain pairs, it is desirable to predict the interaction status of the two protein pairs consistently, so that if the first pair has a large likelihood, the second pair should also have a large likelihood. This idea can be formally described as a constrained optimization problem over a graph. Consider a graph in which each node represents a pair of proteins, labeled with its feature likelihood. An edge is drawn from a node  $p$  to a node  $q$  if the latter pair of proteins shares

some domain pairs as the former pair. The weight of the edge is equal to the fraction of common domain pairs:

$$w_{pq} = \frac{|\mathcal{D}(p) \cap \mathcal{D}(q)|}{|\mathcal{D}(p)|} \quad (5.19)$$

We would like to assign a new set of labels to the nodes, so that 1) nodes connected by edges with large weights have similar labels, and 2) the new node labels do not deviate too much from the original labels. The first criterion originates from the idea that protein pairs that share common domain pairs should have similar feature likelihood, while the second criterion ensures that information at the protein level continues to play a role in the final likelihood estimations. These two criteria can be formulated mathematically as follows. Let  $x$  be the vector of original labels and  $y$  be the vector of new labels. Define the following objective function  $f$ :

$$\begin{aligned} f(y) &= \frac{1}{n} \sum_p (y_p - x_p)^2 + \alpha \frac{2}{n(n-1)} \sum_{p < q} w_{pq} (y_q - y_p)^2 \\ &= \frac{1}{n} \|y - x\|^2 + \frac{2\alpha}{n(n-1)} y^T (I - W)y, \end{aligned} \quad (5.20)$$

where  $n$  is the number of protein pairs,  $\alpha$  is a tradeoff parameter between the two criteria,  $I$  is the identity matrix, and  $W$  is the weight matrix defined as  $W_{pq} = w_{pq}$ . Again, the summations can be taken over only protein pairs in the training set  $T$ , or all protein pairs.

To minimize the objective function, we differentiate it with respect to  $y$ :

$$\begin{aligned}
 \frac{\partial f(y)}{\partial y} &= \frac{2}{n}(y-x) + \frac{2\alpha}{n(n-1)}[(I-W)^T + (I-W)]y \\
 &= \frac{2}{n}(y-x) + \frac{4\alpha}{n(n-1)}(I-W^*)y, \\
 &= \frac{2}{n}[(n-1+2\alpha)I - 2\alpha W^*]y - \frac{2}{n}x
 \end{aligned} \tag{5.21}$$

where  $W^*$  is the symmetrized weight matrix  $W^*_{pq} = \frac{w_{pq}+w_{qp}}{2}$ . By setting the equation to zero, the analytical solution of the  $y$  that minimizes  $f(y)$  is  $[(n-1+2\alpha)I - 2\alpha W^*]^{-1}x$ . Since  $W$  is an  $n \times n$  matrix where  $n$  is the number of protein pairs, which is of the order of millions, taking the inverse directly is infeasible. Instead, the equation can be solved by Jacobi iterations [44]. The initial estimate of  $y$  is simply  $x$ :

$$y^{(0)} = x \tag{5.22}$$

Subsequent approximations are based on this update formula:

$$y^{(t)} = \frac{n-1}{n-1+2\alpha}x + \frac{2\alpha}{n-1+2\alpha}W^*y^{(t-1)} \tag{5.23}$$

The value of a particular component  $p$  of the vector is:



$$\begin{aligned}
 y_p^{(t)} &= \frac{n-1}{n-1+2\alpha}x + \\
 &\quad \frac{2\alpha}{n-1+2\alpha} \sum_{(m,n) \in \mathcal{D}(p)} \left\{ \sum_{q \in \mathcal{P}(m,n), q \neq p} \frac{1}{2} \left[ \frac{1}{|\mathcal{D}(p)|} + \frac{1}{|\mathcal{D}(q)|} \right] y_q^{(t-1)} \right\}
 \end{aligned} \tag{5.24}$$

Brute-force summation is still infeasible if some domain pairs are shared by a large number of protein pairs. However, the term in curly brackets can be rewritten as

$$\begin{aligned}
 &\sum_{q \in \mathcal{P}(m,n), q \neq p} \frac{1}{2} \left[ \frac{1}{|\mathcal{D}(p)|} + \frac{1}{|\mathcal{D}(q)|} \right] y_q^{(t-1)} \\
 &= \frac{1}{2|\mathcal{D}(p)|} \sum_{q \in \mathcal{P}(m,n)} y_q^{(t-1)} + \\
 &\quad \frac{1}{2} \sum_{q \in \mathcal{P}(m,n)} \frac{y_q^{(t-1)}}{|\mathcal{D}(q)|} - \\
 &\quad \frac{y_p^{(t-1)}}{|\mathcal{D}(p)|},
 \end{aligned} \tag{5.25}$$

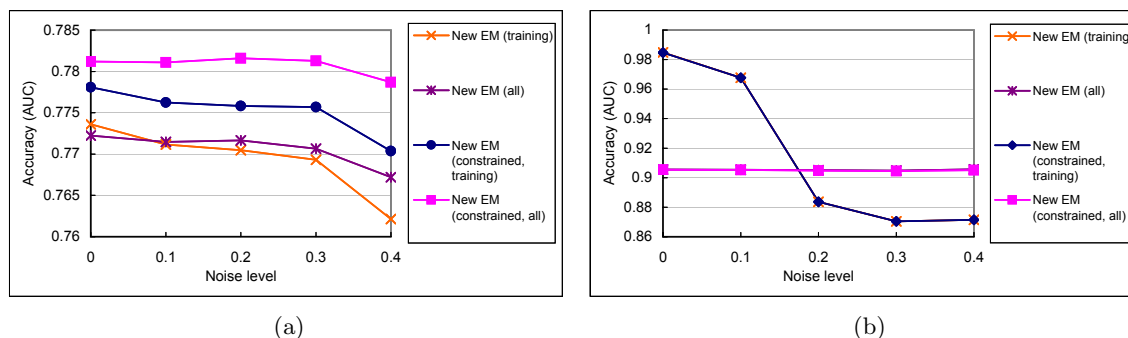
where the first two terms can be pre-computed for each domain pair and the last term can be obtained in constant time.

By considering different possible values of the tradeoff parameter  $\alpha$ , we can obtain an intuitive interpretation of the update formula in Equation 5.23. The estimation of any component  $y_p$  of  $y$  at time  $t$  involves the initial estimate  $x_p$  and a weighted sum of the neighbors of  $p$ . If  $\alpha = 0$ , the formula reduces to  $x$ , which corresponds to the case that the estimation is made using protein-level information only. When  $\alpha = 0.5$ , the weight of  $x$  is  $n - 1$  times the weight of each neighbor. Since there are at most  $n - 1$  neighbors,

and each element in  $W^*$  is no larger than 1, this value of  $\alpha$  is the upper threshold that guarantees protein-level information would have an effect at least as strong as the domain-level information. When  $\alpha = \frac{n-1}{2}$ , the weight of  $x$  is the same as any of the neighbors. Finally, when  $\alpha \gg \frac{n-1}{2}$ , the second term dominates and the value of  $y$  is totally determined by the labels of the neighbors according to domain-level information.

As long as  $\alpha \leq \frac{n-1}{2}$ , the matrix  $(n-1+2\alpha)I - 2\alpha W^*$  remains diagonally dominant, which ensures the convergence of the Jacobi iterations.

We use this method to compute constrained likelihood for the new EM algorithm with several values of  $\alpha$ . The prediction accuracy at the different values is similar, and the results for  $\alpha = 100$  are shown in Figure 5.3.



**Figure 5.3. Comparing the accuracy of the new EM algorithm with constrained and unconstrained likelihood. (a) Accuracy of PPI prediction. (b) Accuracy of DDI prediction.**

The performance of EM is observed to improve slightly when predicting PPI with constrained likelihood, and remain largely the same when predicting DDI.

## 5.4 Discussion

Since some DDI prediction algorithms have built-in error detection mechanisms, another way to handle errors in the PPI network is to actively correct the training sets by detecting dubious training examples, and either removing them from the training sets, or swapping them from the positive set to the negative set, or vice versa.

We have tried this approach by identifying protein pairs that have the largest difference between the input and predicted probabilities of interaction. For example, if a protein pair is in the positive training set, and an algorithm predicts the two proteins to have a very low probability of interaction according to their domain information, this protein pair is a potential false positive. We tested if the prediction accuracy of InSite and BP could be improved by removing these examples or swapping to the opposite training set.

The results suggest that statistically false positives do tend to be predicted with a smaller probability of interaction than true positives according to a Wilcoxin rank sum test of the probabilities of the two set. Similarly, false negatives do tend to be predicted with a larger probability of interaction than true negatives. Yet the precision of error detection is not high enough to be useful in correcting errors. Swapping is observed to be not feasible as it would create even more errors. For example, when the positive training set contains 10% false positives, among the protein pairs in the positive set with the lowest predicted probabilities of interaction, the percentage of real false positives is between 15%-20%. While this percentage is higher than the average false positive rate in the whole positive set, the remaining 80%-85% are true positives. Adding the detected dubious protein pairs to the negative would thus increase the error rate. On the other hand, while removing the examples is guaranteed to reduce the average error rate, it also reduces the size of the training set,

so that less domain interactions would find evidence from the positive protein pairs.

It thus seems more effective to deal with PPI errors by working on the input PPI network before it is used in DDI inference. We have shown that incorporating protein features is one way to reduce noise. Another way is to enforce PPI predictions to consider domain occurrence. The constrained likelihood method is one possible approach, yet more work is still needed to improve its effectiveness.

In general, it is advantageous to incorporate more data if they contain some new information. The main challenge is finding a proper way to extract such knowledge and integrate with existing data in learning. In the next chapter we describe a study in which we successfully incorporated new data into our learning method and outperformed other prediction algorithms.

## Chapter 6

# Adding New Perspectives to Existing Problems: Discovering New Information in New Data

### 6.1 Introduction

In this chapter we switch our focus from the protein interaction network to the gene regulatory network. The expression of genes is tightly controlled by the regulatory machinery in the cell, by regulator proteins called transcription factors (TFs). Taking the simplified view that each gene encodes for a protein, transcription regulation can be modeled as a directed graph with each node representing a gene and its encoded protein, and an edge from one node to another if the former is a regulator of the latter. In addition to the directionality, the edges are also signed, with a positive sign indicating a positive regulation

(activation) and a negative sign indicating a negative regulation (suppression).

Methods have been proposed for computationally reconstructing regulatory networks. One common approach is to use differential equations to model how the expression levels of genes change according to the abundance of their regulator proteins over time [25, 67, 129, 188]. Since it has only recently been possible to quantitatively measure the abundance of proteins in each cell for many proteins simultaneously by flow cytometry [43], protein abundance has been approximated in two ways: 1) the expression level of mRNA has been used as a proxy of the quantity of the corresponding protein; 2) a multi-cell average has been used as a proxy of the quantity in individual cells. With the use of mRNA level to approximate protein abundance, both the data for estimating the expression level of a gene and the activity of its regulators are obtained from the same mRNA microarray assays. Each set of experiments involves an initial experimental condition (e.g., an environmental perturbation such as a heat shock), which affects the expression levels of some genes reacting to the condition. Then additional expression profiles are obtained at different time points as a measure of the changing internal state of the cell.

In the resulting dataset, each data point measures the expression level of a gene in a specific condition at a certain time point. Each such observed value is a mixture of many different factors, including the previous expression level of the gene, the activity of its regulators, decay of mRNA transcripts, randomness, and measurement errors. The many entangled parameters make it difficult to reconstruct the regulatory network based on this type of data alone.

To decode this kind of complex systems, one would want to reduce it to a series of subsystems with manageable sizes by keeping the values of most parameters constant and varying only a small number of them. Thanks to the creation of large-scale deletion li-

libraries [72], it is now possible to carry out this divide-and-conquer strategy. A deletion library contains different strains of a species (e.g. yeast), each of which has one of the genes of the species disabled – completely (*knocked out*) by mutagenesis [72] or partially (*knocked down*) by RNA interference (RNAi) [101]. Profiling the expression of each gene in a deletion strain allows one to study the sub-network that is affected by the deleted gene. For instance, if the deleted gene encodes for a protein that is the only activator of another gene, then the expression level of the latter would be dramatically decreased in the deletion strain of the former as compared to the wild-type strain in which the regulator gene is intact.

While deletion data is good for detecting simple, direct regulatory events, they may not be sufficient for decoding those that are more complicated. For example, if a gene is up-regulated by two TFs in the form of an OR circuit, so that the gene is expressed normally as long as one of the TFs is active, these edges in the regulatory network cannot be uncovered by single-gene deletion data. In such a scenario, traditional time course data could supplement the deletion data in detecting the missing edges. For instance, if at a certain time point both the TFs have a low abundance and the expression rate of the gene is observed to be impaired, this observation could help reconstruct the OR circuit if it provides a good fit to a differential equation model.

In this study we demonstrate how these two types of data can be used in combination to reconstruct regulatory networks. We propose methods for predicting regulatory edges from each type of data, and a meta-method for combining their predictions. Using a set of fifteen benchmark datasets, we show the effectiveness of our approach, which led our team to get the first place in the public challenge of the third Dialogue for Reverse Engineering Assessments and Methods (DREAM) [50]. We will also discuss potential weaknesses of our approach, and directions for future studies.

## 6.2 Problem definition

We first formally define our problem of reconstructing regulatory networks. The target network is a directed graph with  $n$  nodes. The edges are completely unobserved, and we are to predict them from the data features alone. In other words, this is an unsupervised learning setting. The edges are signed, but these signs are not considered in our experimental evaluation. The goal is thus to learn a model from the data features, such that given an ordered pair of two genes  $(i, j)$ , one can predict whether  $i$  is a regulator of  $j$ .

We use two types of data features: perturbation data and deletion data. Deletion data are further sub-divided into homozygous deletion and heterozygous deletion.

In a perturbation time series dataset, an initial perturbation is performed at time 0 by setting the expression levels of each gene to a certain level. Then the regulatory system is allowed to adjust the internal state of the cell by up- and down-regulating genes according to the abundance of the TFs. The expression level of each gene is taken at subsequent time points. Thus, for each perturbation experiment, each gene is associated with a vector of real numbers that correspond to its expression level at different time points after the initial perturbation. If there are  $m$  perturbation experiments and the  $i$ -th one involves  $t_i$  time points, then each gene is associated with a vector of  $\sum_{i=1}^m t_i$  expression values.

In a deletion dataset, a gene is deleted (knocked-out or knocked-down), and the resulting expression level of each gene at steady state is measured. By deleting each gene one by one, and adding the wild-type (no deletion) as control, each gene is associated with a vector of  $n + 1$  values, corresponding to its steady-state expression level in the  $n + 1$  strains. For diploid organisms (with two copies of each gene in the genome), the deletion can be homozygous (with both copies deleted, i.e., “null mutant”) or heterozygous (with only one



copy deleted).

We assume both types of deletion data, as well as perturbation data, are available, although it is trivial to modify our algorithm by simply removing the corresponding sub-routines if any kind of data is missing.

## 6.3 The learning method

Our basic strategy is to learn the simple regulation cases from deletion data using noise models, and to learn the more complex ones from perturbation data using differential equation models. We first describe the two kinds of models and how we learn the parameter values from data, and then discuss our method to combine the two lists of predicted edges into a final list of predictions.

### 6.3.1 Learning noise models from deletion data

We consider a simple noise model for deletion data, that each data point is the superposition of the real signal and a reasonably small Gaussian noise independent of the gene and the time point. The Gaussian noise models the random nature of the biological system, and the measurement error. Based on this model, the larger is the change of expression of a gene  $a$  from wild type to the deletion strain of a gene  $b$ , the more unlikely that the deviation is due to the Gaussian noise only, and thus the larger chance that  $a$  is directly or indirectly regulated by  $b$ .

Notice that the regulation could be direct ( $b$  regulates  $a$ ) or indirect ( $b$  regulates  $c$  that directly or indirectly regulates  $a$ ). There are studies that try to separate the direct

regulation from the indirect ones using methods such as graph algorithms [189] and conditional correlation analysis [155]. In this study we do not attempt to distinguish direct and indirect regulation, and show that even assuming all significant deviation in deletion data to be direct regulation could already provide substantial performance improvements over approaches that focus on perturbation data only.

Given the observed expression level  $x_a^b$  of a gene  $a$  in the deletion strain of gene  $b$ , and its real expression level in wild type,  $x_a^{wt*}$ , we would like to know whether the deviation  $x_a^b - x_a^{wt*}$  is merely due to noise. To answer this question, we would need to know the variance  $\sigma^2$  of the Gaussian, assuming the noise is non-systematic and thus the mean  $\mu$  is zero. If the value of  $\sigma^2$  is known, then the probability for observing a deviation as large as  $x_a^b - x_a^{wt*}$  due to random chance only is simply  $2[1 - \Phi(\frac{|x_a^b - x_a^{wt*}|}{\sigma})]$ , where  $\Phi$  is the cumulative distribution function of the standard Gaussian distribution. The complement,  $p_{b \rightarrow a} = 1 - 2[1 - \Phi(\frac{|x_a^b - x_a^{wt*}|}{\sigma})] = 2\Phi(\frac{|x_a^b - x_a^{wt*}|}{\sigma}) - 1$  is the probability that the deviation is due to a regulation event. One can then rank all the gene pairs  $(b, a)$  in decreasing order of  $p_{b \rightarrow a}$ .

To implement the above procedure, it is necessary to estimate  $\sigma^2$  from data, which is standardly done by using the non-biased sample variance of data points that are not affected by the deleted gene from the wild-type expression. However, this involves two difficulties. First, the set of genes not affected by the deleted gene is unknown and is exactly what we are trying to learn from the data. Second, the observed expression value of a gene in the wild-type strain  $x_a^{wt}$  is also subjected to random noise, and thus cannot be used as the gold-standard reference point  $x_a^{wt*}$  in the calculations.

We propose an iterative procedure to progressively refine our estimation of  $p_{b \rightarrow a}$ . We start by assuming the observed wild-type expression levels  $x_a^{wt}$  are reasonable rough esti-

mates of the real wild type expression levels  $x_a^{wt*}$ . Using them as the initial reference points, we repeat the following three steps for a number of iterations:

1. Calculate the probability of regulation  $p_{b \rightarrow a}$  for each pair of genes  $(b, a)$  based on the current reference points  $x_a^{wt}$ . Then use a p-value of 0.05 to define the set of potential regulation: if the probability for the observed deviation from wild type of a gene  $a$  in a deletion strain  $b$  to be due to random chance only is less than 0.05, we treat  $b \rightarrow a$  as a potential regulation. Otherwise, we add  $(b, a)$  to the set  $P$  of gene pairs for refining the error model.
2. Use the set  $P$  to re-estimate the variance of the Gaussian noise,  $\sigma^2 = \frac{\sum_{(b,a):P} (x_a^b - x_a^{wt})^2}{|P|-1}$ .
3. For each gene  $a$ , we re-estimate its wild-type expression level by the mean of its observed expression levels in strains in which the expression level of  $a$  is unaffected by the deletion:  $x_a^{wt} := \frac{x_a^{wt} + \sum_{b:(b,a) \in P} x_a^b}{1 + |b:(b,a) \in P|}$ .

After the iterations, the probability of regulation  $p_{b \rightarrow a}$  is computed using the final estimate of the reference points  $x_a^{wt}$  and the variance of the Gaussian noise  $\sigma^2$ .

Notice that we have chosen to use a “conservative” p-value of 0.05 in the following sense: when the number of genes in the network,  $n$ , is sufficiently large (e.g.  $n \geq 10$ ) and there are relatively few regulatory edges, there is a large number of gene pairs for estimating the parameters such that missing some of them would not seriously affect the estimation. It would thus be good to add to  $P$  only gene pairs that are very unlikely to contain regulatory edges. This is achieved by using a large (i.e., conservative in this context) p-value to define the potential regulatory edges.

The above iterative procedure can be applied to both homozygous and heterozygous

deletion data, although the regulation signals are expected to be less clear in the heterozygous case since deleting only one copy of a regulator gene may induce only a mild effect to its targets. The final p-values computed from homozygous data are thus expected to be more reliable. Yet the ones learned from heterozygous data can still be useful references in resolving ambiguous cases, as we will discuss in more detail when describing our approach to combining the predictions learned from the different types of data.

### 6.3.2 Learning differential equation models from perturbation time series data

For time series data after an initial perturbation, we use differential equations to model the gene expression rates. The general form is as follows:

$$\frac{dx_i}{dt} = f_i(x_1, x_2, \dots, x_n), \quad (6.1)$$

where  $x_i$  represents the expression level of gene  $i$  and  $f_i$  is a function that explains how the expression rate of gene  $i$  is affected by the expression level of all the genes in the network, including the level of gene  $i$  itself. Various types of function  $f_i$  have been proposed. We consider three of them. The first one is a linear model [67]:

$$\frac{dx_i}{dt} = a_{i0} - a_{ii}x_i + \sum_{j \in S} a_{ij}x_j, \quad (6.2)$$

where  $a_{i0}$  is the basal expression rate of gene  $i$  in the absence of regulators,  $a_{ii}$  is the decay rate of the mRNA transcripts of  $i$ , and  $S$  is the set of potential regulators of  $i$ . In theory,  $S$  could be set as  $[n] = \{1, 2, \dots, n\}$ , the whole set of genes in the network, as the regulators of  $i$  are unknown. However, for performance reasons,  $S$  is usually restricted to some small sets of genes. Our choice of  $S$  will be discussed below. For each potential regulator  $j$ ,  $a_{ij}$  explains how the expression of  $i$  is affected by the abundance of  $j$ . A positive  $a_{ij}$  indicates that  $j$  is an activator of  $i$ , and a negative  $a_{ij}$  indicates that  $j$  is a suppressor of  $i$ .

The linear model assumes a linear relationship between the expression level of the regulators and the resulting expression rate of the target. It is a rough first approximation of the expression rate. An advantage of the model is the small number of parameters ( $|S| + 2$ ), yet real biological regulatory systems seem to exhibit non-linear characteristics. The second model we consider assumes the more realistic sigmoidal relationship between the regulators and the target [188]:

$$\frac{dx_i}{dt} = \frac{b_{i1}}{1 + \exp(-a_{i0} - \sum_{j \in S} a_{ij}x_j)} - b_{i2}x_i, \quad (6.3)$$

where  $b_{i1}$  is the maximum expression rate of  $i$  and  $b_{i2}$  is its decay rate. This model involves  $|S| + 3$  parameters.

The third model we consider has a multiplicative form, with each factor capturing the relationship between the target and one of its regulators [129]:

$$\frac{dx_i}{dt} = a_{i0} \prod_{j_1 \in S_1} \frac{b_{ij_1}}{x_{j_1}^{c_{ij_1}} + b_{ij_1}} \prod_{j_2 \in S_2} \frac{x_{j_2}^{c_{ij_2}}}{x_{j_2}^{c_{ij_2}} + b_{ij_2}} - a_{i1}x_i, \quad (6.4)$$

where  $S_1$  and  $S_2$  represent the sets of suppressors and activators, respectively,  $a_{i1}$  is the decay rate,  $b_{ij_1}$  and  $b_{ij_2}$  are rate constants, and  $c_{ij_1}$  and  $c_{ij_2}$  are sigmoidicity constants. This model involves  $2|S_1| + 2|S_2| + 2$  parameters.

In our actual implementation, the exponent terms in the third model sometimes caused numerical instability when the base was close to zero. We therefore based our predictions on the first two models.

Our goal is to try different possible regulator sets  $S$  (or  $S_1$  and  $S_2$ ) and identify the ones that predict the observed expression levels well in the least-square sense:

$$g_i(\theta) = \sum_t (x_{it} - \hat{x}_{it})^2, \quad (6.5)$$

where  $\theta$  denotes the set of parameters ( $a$ ,  $b$  and  $c$ ),  $x_{it}$  is the expression level of gene  $i$  at time point  $t$ , and  $\hat{x}_{it}$  is the corresponding predicted level of a model. The summation is taken over all time points of all perturbation experiments.

The objective function is not convex with respect to the parameters. We use Newton's method [27] to find local minima of the objective function  $g_i(\theta)$  with 100 random initial values of  $\theta$ , and adopt the one that provides the best fit with the smallest  $g_i(\theta)$ . The expression vector  $\hat{x}$ , gradient  $\nabla \hat{x}$  and Hessian  $\nabla^2 \hat{x}$  are estimated by using the closed-form formulas provided by the second order Runge-Kutta method [44].

We try two types of regulator sets. The first type involves single regulators, in which we try each gene  $j$  as the potential regulator of gene  $i$  in turn, and compare the least square errors of their best-fit models. The second type involves high-confidence potential regulators, plus one extra regulator to be tested. As we will see in the next section, the high-confidence potential regulators are obtained from the predictions of the noise models learned from the deletion data, as well as those predicted by the single-regulator differential equation models. We call such models the “guided models” since the construction of the regulator sets is guided by previous predictions. The full detail of the resulting algorithm will be given in the next subsection.

We also tried double regulator sets with all pairs of potential regulators. Yet the resulting models did not appear to provide much additional information on top of the single regulator set models, while requiring much longer computational time. We therefore decided to consider only the single regulator sets and guided single regulator sets.

For a regulator set  $S$  and a target gene  $i$ , the value of the objective function of the best model indicates how likely  $i$  is regulated by the members of  $S$ . The values are thus used to rank the likelihood of existence of the regulatory edges.

### 6.3.3 Combining the predictions of the models

Our main idea for combining the predictions of the different models learned from deletion and perturbation data is to rank the predictions according to our confidence that they are correct. Specifically, we make predictions in batches, with the first batch containing the most confident predictions, and each subsequent batch containing the most confident predictions that have not been covered by the previous batches. Within each batch, the

predictions are ordered by the confidence of the models, which corresponds to the probability of regulation  $p_{b \rightarrow a}$  for noise models, and negated objective score  $-g_i(\theta)$  for differential equation models. We define the batches as follows:

- Batch 1: all predictions with a probability of regulation larger than 0.99 according to the noise model learned from homozygous deletion data
- Batch 2: all predictions with an objective score two standard deviations below the average according to all types of differential equation models learned from perturbation data
- Batch 3: all predictions with an objective score two standard deviations below the average according to all types of guided differential equation models learned from perturbation data, where the regulator sets contain regulators predicted in the previous batches, plus one extra potential regulator
- Batch 4: as in batch 2, but requiring the predictions to be made by only one type of the differential equation models as opposed to all of them
- Batch 5: as in batch 3, but requiring the predictions to be made by only one type of the differential equation models as opposed to all of them
- Batch 6: all predictions with a probability of regulation larger than 0.95 according to both the noise models learned from homozygous and heterozygous deletion data, and have the same edge sign predicted by both models
- Batch 7: all remaining gene pairs, with their ranks within the batch determined by their probability of regulation according to the noise model learned from homozygous deletion data



In general, we put the greatest confidence in the noise model learned from homozygous deletion data as the signals from this kind of data are clearest among the three types of data. We are also more confident with predictions that are consistently made, either by the different types of differential equation models (batches 2 and 3 vs. batches 4 and 5) or by the noise models learned from homozygous and heterozygous deletion data (batch 6).

## 6.4 Performance study

### 6.4.1 Datasets and performance metrics

We used the algorithm described above to take part in the third Dialogue for Reverse Engineering Assessments and Methods Challenge (DREAM3) [51] on regulatory network reconstruction. The challenge involves fifteen benchmark datasets, five of which have 10 genes, five have 50 and five have 100. The networks are constructed based on parameters extracted from modules in real biological networks [124]. At each size, two of the networks are based on parameters from the regulatory network of *E. coli*, and three are based on yeast.

The predictions are compared against the actual edges in the networks by the DREAM organizer using four different metrics for evaluating the accuracy:

- AUPR: The area under the precision-recall curve
- AUROC: The area under the receiver-operator characteristics curve
- pAUPR: The p-value of AUPR based on the distribution of AUPR values in 100,000 random network link permutations

- pAUROC: The p-value of AUROC based on the distribution of AUROC values in 100,000 random network link permutations

These metrics are further aggregated into an overall p-value for each size using the geometric mean of the five p-values from the five networks, and finally an overall score equal  $-0.5 \log_{10}(p_1 p_2)$ , where  $p_1$  and  $p_2$  are the geometric means of pAUPR and pAUROC respectively.

### 6.4.2 Results

The challenge of size 10 has attracted 29 teams to participate, the one of size 50 has 27 teams and the one of size 100 has 22 teams. The large number of participants makes the challenge currently the largest benchmark for gene network reverse engineering [51].

Our algorithm ended in first place on all three network sizes. The complete set of performance scores for all teams can be found at the DREAM3 web site [51]. Below we summarize our prediction results, and discuss some interesting observations.

Table 6.1 and Table 6.2 show the AUROC and pAUROC values of our predictions reported by the DREAM organizer, respectively. From the p-values, we see that our predictions are consistently significantly better than random. In general, we observe that our method performed better on the E. coli networks, but is relatively unaffected by the network size, as evaluated by AUROC.

We notice that in some cases our first predictions are already very close to the actual network. Figure 6.1(a) shows the actual network of the Yeast1-size10 network, where an arrowhead represents an activation and a blunt-end represents a suppression. Figure 6.1(b)

**Table 6.1. AUROC of our predictions.**

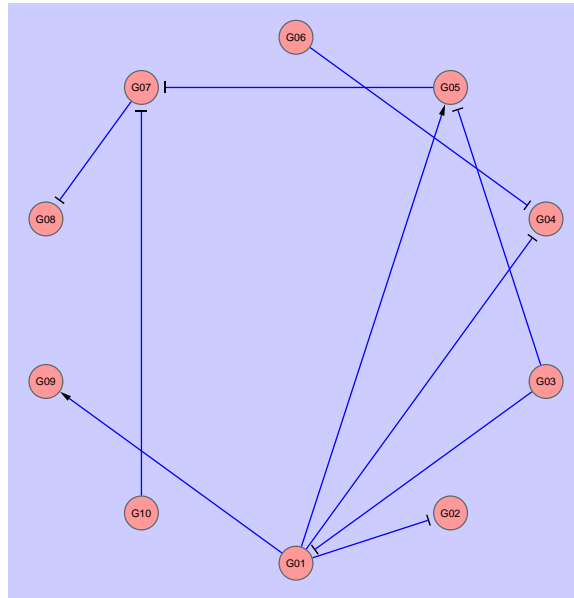
	Ecoli1	Ecoli2	Yeast1	Yeast2	Yeast3
Size 10	0.928	0.912	0.949	0.747	0.714
Size 50	0.930	0.924	0.917	0.792	0.805
Size 100	0.948	0.960	0.915	0.856	0.783

**Table 6.2. pAUROC of our predictions.**

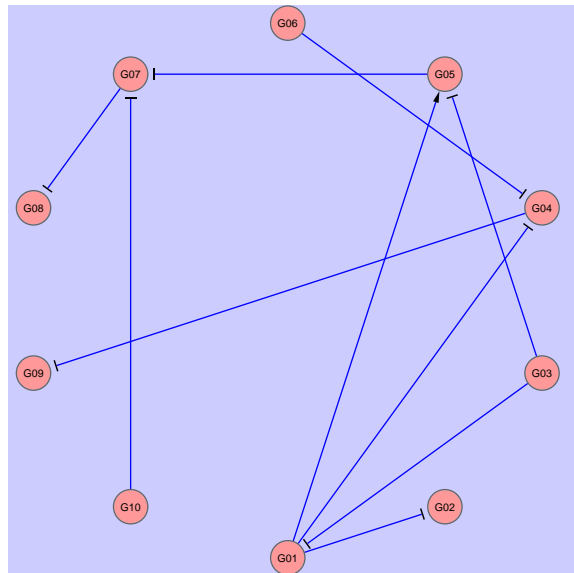
	Ecoli1	Ecoli2	Yeast1	Yeast2	Yeast3	Overall AUROC
Size 10	9.771e-07	2.629e-07	9.941e-07	2.931e-04	1.046e-03	9.523e-06
Size 50	2.396e-27	4.328e-31	1.477e-25	1.808e-21	1.386e-29	5.210e-27
Size 100	1.226e-52	5.876e-42	4.087e-70	5.755e-99	1.722e-92	3.112e-71

shows our top 10 predictions. There is only one false positive (G01 activates G09) and one false negative (G04 suppresses G09). Interestingly, these two edges are tightly related. Since in the actual network G01 suppresses G04 and G04 suppresses G09, G01 can be viewed as indirectly activating G09. Our method thus correctly identified this relationship, yet it failed to distinguish between the direct and indirect regulation. We will discuss the issue of indirect regulation more in the next section.

The overall scores are 5.124, 39.828, and 10e10, respectively, for the size 10, 50 and 100 networks. As a comparison, the scores for the first runners-up are 3.821, 31.341 and 45.443, respectively. We hypothesize that the performance difference is at least partially attributed to our emphasis on the use of deletion data, as it appears that some other high-ranked teams put most of their concentration on building differential equation models from perturbation data (personal communications during the DREAM conference). To demonstrate the effectiveness of the noise models learned from deletion data, we analyze the number of predictions made in each batch, and the number of which are actually correct. The results for the size 10, 50 and 100 networks are shown in Table 6.3, Table 6.4 and Table 6.5, respectively.



(a)



(b)

Figure 6.1. The Yeast1-size10 network. (a) The actual network. (b) Our top 10 predictions.

**Table 6.3. Prediction accuracy per batch on the size 10 networks.**

Batch	Ecoli1		Ecoli2		Yeast1		Yeast2		Yeast3	
	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct
1	11	7	16	12	11	9	13	9	12	8
2	6	1	4	0	5	0	5	1	5	4
3	0	0	1	1	3	0	1	0	1	0
4	5	1	8	0	7	0	4	2	4	0
5	4	0	8	1	6	0	10	3	5	1
6	1	1	0	0	0	0	0	0	0	0
7	63	1	53	1	58	1	57	10	63	9
Total	90	11	90	15	90	10	90	25	90	22

**Table 6.4. Prediction accuracy per batch on the size 50 networks.**

Batch	Ecoli1		Ecoli2		Yeast1		Yeast2		Yeast3	
	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct
1	96	52	133	69	145	57	176	83	201	100
2	76	2	85	1	80	8	87	12	102	16
3	77	0	78	1	69	1	56	1	64	2
4	196	0	153	1	185	1	156	5	113	3
5	178	1	169	1	167	2	177	6	149	2
6	5	0	16	0	9	0	11	0	6	0
7	1822	7	1816	9	1795	8	1787	53	1815	50
Total	2450	62	2450	82	2450	77	2450	160	2450	173

**Table 6.5. Prediction accuracy per batch on the size 100 networks.**

Batch	Ecoli1		Ecoli2		Yeast1		Yeast2		Yeast3	
	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct
1	410	101	377	108	483	118	656	257	710	302
2	387	11	319	1	317	20	282	22	311	31
3	162	0	198	0	129	0	145	3	135	3
4	650	0	685	1	575	2	604	12	638	13
5	683	1	656	2	746	3	739	10	667	24
6	53	0	72	0	82	2	67	0	59	2
7	7555	12	7593	7	7568	21	7407	85	7380	176
Total	9900	125	9900	119	9900	166	9900	389	9900	551

**Table 6.6. Prediction of the first two batches on the size 10 networks when their orders are swapped.**

Batch	Ecoli1		Ecoli2		Yeast1		Yeast2		Yeast3	
	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct	Predicted	Correct
1	6	1	5	1	5	0	5	1	5	4
2	11	7	15	12	11	9	13	9	12	8

As hypothesized, the noise models learned from homozygous deletion data made very accurate predictions. In many cases, most actual edges were already predicted correctly in batch 1. Also, if an actual edge is not predicted in batch 1, it is also likely missed by subsequent batches. For instance, for the 173 actual edges in the Yeast3-size50 network, 100 are detected in batch 1, and among the remaining 73, only 21 are detected in batches 2 to 6.

While the above results suggest the importance of the noise models learned from homozygous data, it is still not clear whether these models are indeed more effective than the other models. It could still be the case that other models could also make the same predictions made in batch 1, just that as these predictions had already been covered in batch 1 that subsequent batches were not allowed to make the same predictions again. To verify if this was the case, we swapped the order of the first two batches for the size 10 networks, so that the first batch is composed of predictions made by the differential equation models and the second batch is composed of predictions made by the noise model learned from homozygous deletion data and not covered by the first batch. The results are shown in Table 6.6.

Comparing Table 6.6 and the first two batches of Table 6.3, it is seen that the number of predictions made by the models almost remained unchanged when the order of the two batches are swapped. In fact, by checking the predicted edges, it is observed that most pre-

dictions previously made by the noise model were not predicted by the differential equation models, even they were given the chance to freely make the predictions. Only one extra correct prediction could be made by the differential equation models for the Ecoli2 network.

This analysis reveals two interesting observations. First, as the noise model learned from deletion data gave higher accuracy than the differential equation models, our decision to use the former to make the first batch predictions is justified. Second, while the differential equation models had a lower accuracy, they made some unique correct predictions that were missed by the noise models. The results thus suggest that the two types of models, based on two different types of data, are complementary to each other and are able to make some orthogonal contributions to the overall predictions.

## 6.5 Discussion and future directions

Our prediction results demonstrate the advantage of combining multiple types of data. While the perturbation data allow the learning of differential equation models that could capture complex interactions in the regulatory network, deletion data also facilitate the detection of some simple interactions using only very basic noise models. As technological advancements are made rapidly, new data types are expected to come out from time to time. For method developers who try to improve existing prediction methods, besides deriving more advanced algorithms using the same data, it is also rewarding to investigate what kinds of information emerging data could provide, and how such information can be extracted to supplement existing methods.

As mentioned earlier, in this study we did not attempt to address the issue of indirect regulation. Indeed we observed that indirect regulation is one of the factors that confounded

our method and caused it to make some wrong predictions. We expect that in a complete network with thousands of nodes, long regulation chains exist and the problem of indirect regulation would be more serious. It is therefore interesting to see if filtering indirect regulation (for example by some existing techniques [155, 189]) could further improve the performance of the method.

In this study, we adopt an unsupervised learning setting, in compliance with the setup of the DREAM3 challenge. For organisms with some known regulation edges as domain knowledge, they can be used as training examples to train a supervised learner, or be used to transform the existing method into a semi-supervised one [34]. For example, known examples can be used to setup p-value cutoffs in defining the potential regulation set  $P$  when learning the noise models. They can also help examine the validity of a particular differential equation model formulation, by checking if the squared errors of their best models are indeed smaller than average.

One issue that we have not touched on is the computational cost. Using a high-end cluster, the predictions for networks of size 10, 50 and 100 took about 2 minutes, 13 hours, and 78 hours, respectively. While there is room for optimizing our code, fitting the differential equation models intrinsically requires a lot of computational power. Given that most correct predictions are made by the noise models, which only took a tiny portion of the computational time, when working on complete networks it is possible to tradeoff some accuracy for much shorter running time.



## Part II

# Data Integration

# Chapter 7

## Semantic Web

### 7.1 Introduction

The web has become instrumental to many facets of research in the life sciences domain. Nowadays, researchers can easily have Internet access to a large quantity and variety of biological data using their web browsers running on local desktop computers. As the number of these web resources continues to increase, it is important to address the problem of interoperability. Currently, it is a challenging problem for the following reasons.

1. It is difficult to automatically identify web sites that contain relevant and interoperable data, as there is a lack of widely accepted standards for describing these web sites as well as their contents. Although approaches like the HTML meta tag (<http://www.htmlhelp.com/reference/html40/head/meta.html>) can be used to annotate a web page through the use of keywords, they are problematic in terms of sensitivity and specificity. In addition, these approaches are neither supported nor used widely

by existing web search engines. Most web search engines rely on using their own algorithms to index individual web sites based on their contents.

2. Different resources provide their data in heterogeneous formats. For example, while some data are represented in HTML format that is interpretable by the web browser, other data formats including the text format (e.g., tab-delimited files) and binary format (e.g., images) are used. Such heterogeneity in data formats makes interoperability difficult if not impossible.
3. Data interoperability involves both syntactic and semantic translation. Both types of translation are hindered by the lack of standard data models, formats, and vocabulary/ontology.

The semantic web research community addresses these problems by seeking methods to facilitate machine-based identification and semantic interoperability of web resources. Crucial to the semantic web approach is the design and development of ontologies (semantic part) that are represented in computer-readable formats (syntactic part). The eXtensible Markup Language (XML) has become a standard syntax for expressing data that are exchanged between applications. In the past several years, a large collection of XML-based formats has emerged for representing different types of biological data. Examples include mzXML [145] for standardizing the representation of mass spectrometry (MS) data generated by different MS instruments, BioML [62] for representing biopolymer data, MAGE-ML [175] for representing microarray gene expression data, SBML [91] for representation and exchange of biochemical network models, and ProML [87] for specifying protein sequences, structures and families. In addition, since XML is widely used there are a large number and variety of open source software tools for processing it.

While these XML formats facilitate data exchange between applications, they do not adequately address semantics and lack expressivity for knowledge representation and inference [46]. In addition, there is a proliferation of semantically-overlapping XML formats in the life sciences domain, making syntactic and semantic data translation more complex and difficult. For example, AGAVE (<http://www.agavexml.org>) and BSML (<http://www.bsml.org>) are different XML formats for describing sequence annotation. SBML, PSI-MI [89], BIND XML [4], and BioPax (<http://www.biopax.org/>) are examples of pathway/network data formats. Efforts have been underway to unify some of these XML formats. For example, MAML and GEML, which were two separate microarray gene expression data formats, were consolidated into MAGE-ML.

The Resource Description Framework (RDF) is a standardized XML format designed to describe web resources. The RDF structure is generic in the sense that it is based on the directed acyclic graph (DAG) model. RDF is a model for defining statements about resources and relationships among them. Each statement is a triplet consisting of a subject, a property, and a property value (or object). For example, <“Protein” “Name” “P53”> is a triple statement expressing that the subject “Protein“ has “P53” as the value of its “Name” property. RDF also provides a means of defining classes of resources and properties. These classes are used to build statements that assert facts about resources. Each resource possesses one or more properties. While the grammar for XML documents is defined using DTD or XSchema, RDF uses its own syntax (RDF Schema or RDFS) for writing a schema for a resource. RDFS is expressive and it includes subclass/superclass relationships as well as constraints on the statements that can be made in a document conforming to the schema. Unlike the order of elements in XML, the order of RDF properties does not matter, thereby giving more flexibility to web programmers in developing their applications. While RDF

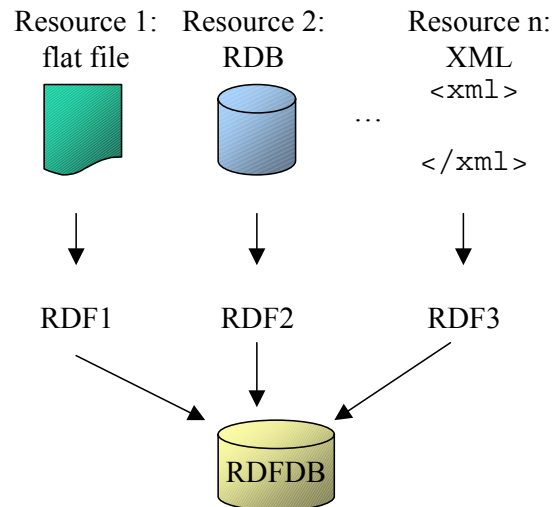
can be serialized to a standard XML format, other representations such as Notation3 also exist.

The generic structure of RDF makes data interoperability and evolution easier to handle as different types of data can be represented using the common graph model. RDF extensions such as the Web Ontology Language (OWL) support more sophisticated knowledge representation and inference. Such languages allow data semantics to be defined declaratively (not procedurally) and can be used as a common model for expressing different types of biological data that are currently defined using different XML syntaxes. There are already some biological data that are expressed in RDF format. Examples include Gene Ontology [12], NCI thesaurus [77], and UniProt [9].

As RDF is gaining more attention in the bioinformatics community and more RDF-related tools and technologies are becoming available, it is important to find new use cases of RDF in the life sciences domain (<http://www.w3.org/2004/07/swls-ws.html>). To this end, our paper demonstrates how to use RDF metadata/data standards (e.g., RDF Site Summary or RSS) and RDF-based technologies (e.g., native RDF database) to facilitate integration of diverse types of genome data provided by multiple web resources in heterogeneous formats. This builds upon our previous work on using XML to interoperate heterogeneous genome data [38].

## 7.2 RDF data warehouse

Figure 7.1 gives a system overview of our semantic web approach to data integration. It entails the following steps.



**Figure 7.1. System overview.**

1. Describing and downloading the contents (as tab-delimited or RDF files) from individual genome web sites.
2. Converting the downloaded data into our RDF format if these data are in tab-delimited format. If the data files are in RDF format (even though they are different from our RDF format), no conversion is required.
3. Loading the RDF-formatted data files into an RDF-native database for data storage, management, and retrieval. Once the data are stored in a repository, (web-enabled) applications can be written to allow users to access, query, and analyze the data.

For data that are already stored in relational databases, we explore a relational-database-to-RDF mapping method, D2RQ (<http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rq>), which allows existing (or legacy) relational databases to publish data in RDF format

via a high-level mapping specification language.

### 7.2.1 RDF data stores

While relational database management systems (RDBMS) are the predominant platform for storing, managing, and querying biological data, they do not directly fit the RDF structure that is based on the DAG model. Mapping methods or new database engines are needed to handle RDF datasets efficiently. Given the growing use of RDF, specialized data storage methods (called “triple stores”) have been developed to efficiently store, manage, and query RDF-formatted data. Representative approaches include: Sesame (<http://www.openrdf.org>), Kowari (<http://www.kowari.org>), Joseki (<http://www.joseki.org>), and Triplestore (<http://triplestore.aktors.org>).

Some data store approaches (e.g., Sesame) use or provide the option to use a relational database (e.g., Oracle, MySQL, and Postgres) as the underlying persistent store. Others (e.g., Kowari) allow a repository to be created directly on top of the RDF files without the need of using a relational database. Many of these RDF database systems come with their own implementation of RDF query languages (e.g., SeRQL is implemented by Sesame and iTQL by Kowari).

A scalability report on existing RDF data stores has been published (<http://simile.mit.edu/reports/stores/>). In the report, Sesame and Kowari are rated high in terms of their performance, ease of use, and deployment. Based on this report, we have made the decision to use Sesame to implement the data warehouse. In addition, Sesame is the only system that allows main memory, relational database, and file approaches to be used to construct a repository. This lets us compare these underlying storage approaches.

### 7.2.2 Metadata and data

In our system, each resource has two RDF files created and associated with it, metadata and data. Figure 7.2 shows the first step of entering information needed to generate the metadata. Based on the information entered, our system will generate metadata in RDF format. The RDF format that we use is based on the RDF Site Summary (RSS; <http://web.resource.org/rss/1.0/>), which is a standard format between web sites. In RSS terms, each resource is known as a channel. The basic idea of RSS is that each news web site will publish (or syndicate) its headline and description of its contents as an RSS feed; applications such as aggregators can spider these RSS-enabled sites and assemble their feeds. We use a similar idea to create and store the genome-oriented RSS feeds centrally. Our data warehouse system can be considered as an aggregator that integrates the data that are described in the RSS feeds.

The RSS format we use incorporates different sets (or modules) of vocabularies including the Dublin Core Metadata (DCM) vocabulary (<http://dublincore.org/documents/dcmi-terms/>). We use the following DCM terms/properties.

1. *Source URL* gives the web address or UR which the original data resource (or channel) can be accessed. In our case a resource or channel is a particular data file.
2. *Format* indicates the format of the original data file: tab-delimited and RDF.
3. *Title* is a descriptive name given to a resource.
4. *Type of resource* is a list of types that can be used to categorize the nature of the content of the resource.
5. *Language* indicates the language in which the resource contents are published.



**Part 1: metadata**

[Dublin Core Standard Terms](#); (\*=Mandatory fields)

\* [Source](#) (URL):

\* [Format](#):

\* [Title](#):

[Resource type](#):

[Language](#):

\* [Description](#):

[Creator](#):

[Publisher](#):

[Created](#) (date published):

[Contributor\(s\)](#):  (comma-separated list)

[Bibliographic citation](#):

Load data into repository

(For tab-delimited files:)

Column header row  (first row count as 1, leave it blank if the file does not have a header row)

\*First data row  (first row count as 1)

Figure 7.2. Metadata generation step.

6. *Description* gives an account of the resource content.
7. *Identifier* is used to identify a resource uniquely. Our system generates this identifier automatically and assigns it to the identifier property.
8. *Creator* indicates the entity (e.g., a person, organization, or service) that is responsible for making the original resource available.
9. *Publisher* indicates the entity (e.g. a person, organization, or service) that makes the resource that is derived from the original resource available.
10. *Created* indicates the date on which the original resource is created.
11. *Contributor* identifies the individual(s) who makes contribution to the content of the resource.
12. *Bibliographic citation* gives a bibliographic reference to the resource.

While *title*, *description*, *identifier* (generated by the system) and *source URL* are mandatory, the other properties are optional. By using these standard properties, we hope to broaden the utility and sharing of metadata. Figure 7.3 gives an example of the metadata represented using these properties in RSS format.

If the source data file is in RDF format, the user just needs to provide the URL of the corresponding schema file. If the source data file is in tabular format, the user needs to indicate whether the data file contains column headers and if so, at which line they occur. Also, the user needs to indicate the line number of the first data row. In addition to data conversion, the user is offered the option to load the converted dataset into the RDF repository for storage and later query retrieval; queries can be done not only for the just

```

<?xml version="1.0" ?>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rss="http://purl.org/rss/1.0/">
  <rss:channel rdf:about="http://twiki.med.yale.edu/kei_web/yeasthub/mips_lethal_genes2.txt">
    <rss:link>http://twiki.med.yale.edu/kei_web/yeasthub/mips_lethal_genes2.txt</rss:link>
    <dc:identifier>58639</dc:identifier>
    <dc:format>tabDelimited</dc:format>
    <rss:title>MIPS genes</rss:title>
    <dc:type>Yeast genome data</dc:type>
    <dc:language>English</dc:language>
    <rss:description>MIPS essential genes</rss:description>
    <rss:items>
      <rdf:Seq>
        <rdf:li rdf:resource="http://mcd750.med.yale.edu/yeasthub/data/datanull.rdf" />
      </rdf:Seq>
    </rss:items>
  </rss:channel>
  <rss:item rdf:about="http://mcd750.med.yale.edu/yeasthub/data/datanull.rdf">
    <rss:link>http://mcd750.med.yale.edu/yeasthub/data/datanull.rdf</rss:link>
    <rss:title>MIPS genes</rss:title>
  </rss:item>
</rdf:RDF>

```

**Figure 7.3. Metadata encoded in RSS 1.0 format.**

stored dataset, but also integrated queries over all resources stored in the repository can be done.

During the second step of data registration - data generation (as shown in Figure 7.4), the user needs to provide information on how the RDF data format should be generated based on the tabular structure (as shown on top of Figure 7.4). This is divided into two parts.

1. The first part requires the user to indicate the type of genome objects and the organism involved. In addition, the user needs to enter the default namespace for the properties to which the file columns (headers) are mapped (see below). Finally, the user indicates which column (if any) is the ID column by entering the corresponding URL, which includes the string pattern “[ID]” that will be replaced by the actual ID value.
2. In the second part, the property name is entered for each file column selected by the user. If the source file contains column headers, the header labels will be used as the default property names (which can later be edited by the user). It is possible that the properties may have been defined in schemas identified by different namespaces. Therefore, the interface provides the user with the option to enter a namespace for each property. In addition, the interface lets the user indicate whether a single column entry contains multiple values (e.g., gene synonyms separated by “—”). If so, the user has to indicate the delimiter (e.g., comma or space) that is used to separate the values. In this case, the corresponding RDF output will have multiple property-value pairs. This may simplify data querying later. Finally, the interface allows the user to replace a substring pattern with another substring pattern when converting column values to property values. For example, a GO ID in one resource may contain a colon (e.g.,

Row	ORF	Gene Name	Gene Synonyms
1	YOR122C	PFY1	profilin
2	YOR143C	THI80	thiamin pyrophosphokinase
3	YOR157C	PUP1	20S proteasome subunit (beta2)

...

**Part 2: data**

Please supply the following information for converting the tab-delimited file to RDF format.

Each row represents a  of

\*ID column:  (first column count as 1)

\*ID URI:  (e.g. "http://foo.org/bar?[ID]", where the [ID] symbol will be replaced by real values in the ID column)

\*Default namespace: (for columns without specific namespaces)

---

Include column 1

\*Name:

Namespace:

This field contains multiple values. Value separator:

Search each value for  and replace all occurrences with  [help on regular expression](#)

---

Include column 2

\*Name:

Namespace:

This field contains multiple values. Value separator:

Search each value for  and replace all occurrences with  [help on regular expression](#)

---

Include column 3

\*Name:

Namespace:

This field contains multiple values. Value separator:

Search each value for  and replace all occurrences with  [help on regular expression](#)

Figure 7.4. Tabular-to-RDF data conversion.

GO:12345), while in another resource it has no colon (e.g., GO12345). Such a substring replacement function helps reduce data variability between resources, thereby easing data integration.

Currently, our RDF conversion applies only to data that are represented in tab-delimited format. In addition to converting tab-delimited files into RDF format, our system generates the corresponding RDF schema. Figure 7.5 depicts the RDF schema generally. In the figure, there is a class named genome object that is associated with a collection of individual genome objects (a collection is a special type of RDF container). Also, genome object has the properties, object type and organism which describe the type of the genome objects involved (e.g., genes, markers, or proteins) and the organism of interest (e.g., yeast, human, or mouse). Each genome object in the collection can be described by a set of properties that can be user-defined or derived from existing standard vocabularies. Different collections of genome objects (obtained from different sources) may involve different sets of properties.

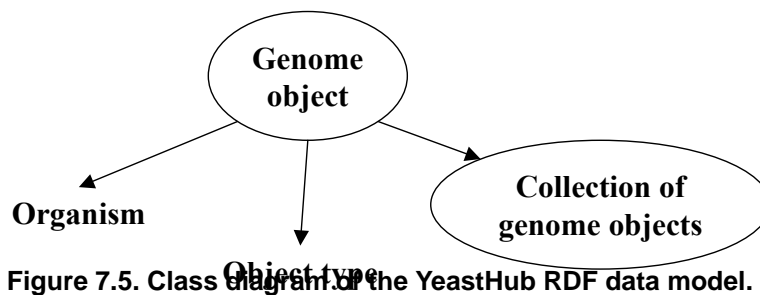


Figure 7.5. Class diagram of the YeastHub RDF data model.

Figure 7.6 illustrates how a collection of yeast genes is expressed in our RDF/XML format. In this example, the description of each yeast gene includes the standard open reading frame (ORF) name, common gene name, and synonyms. Each gene is identified by a URL that takes the ORF name as a parameter and returns the detailed description of the

gene from MIPS.

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:yh="http://mcd750.med.yale.edu/yeasthub/schema/yeasthub_schema.rdf"
xmlns:ns0="http://mcd750.med.yale.edu/yeasthub/schema/schema58639.rdf">
  <rdf:Description rdf:about="http://twiki.med.yale.edu/kei_web/yeasthub/mips_lethal_genes2.txt">
    <yh:object_type>gene</yh:object_type>
    <yh:organism>yeast</yh:organism>
    <yh:genome_objects rdf:parseType="Collection">
      <rdf:Description
rdf:about="http://mips.gsf.de/genre/proj/yeast/searchEntryAction.do?text=YAL001C&db=CYGD">
        <ns0:row_count>1</ns0:row_count>
        <ns0:orf>YAL001C</ns0:orf>
        <ns0:gene_name>TFC3</ns0:gene_name>
        <ns0:gene_synonyms>TFIIIC (transcription initiation factor) subunit, 138 kD</ns0:gene_synonyms>
      </rdf:Description>
      <rdf:Description
rdf:about="http://mips.gsf.de/genre/proj/yeast/searchEntryAction.do?text=YAL003W&db=CYGD">
        <ns0:row_count>2</ns0:row_count>
        <ns0:orf>YAL003W</ns0:orf>
        <ns0:gene_name>EFB1</ns0:gene_name>
        <ns0:gene_synonyms>translation elongation factor eEF1beta</ns0:gene_synonyms>
      </rdf:Description>
      ...
    </yh:genome_objects>
  </rdf:Description>
</rdf:RDF>
```

**Figure 7.6.** An example collection of yeast essential genes represented in RDF/XML format.

The link connecting the generated RDF metadata file, data file, and the schema file is via a common system-generated identifier that is stored in the property identifier in the metadata file. We create a RDF repository for each file type.

### 7.3 Biological use case: YeastHub

To demonstrate how to use semantic web techniques to integrate diverse types of genome data in heterogeneous formats, we have developed a prototype application called “YeastHub”. In this application, a data warehouse has been constructed using Sesame to store and query a variety of yeast genome data obtained from multiple sources. For perfor-

mance reasons, we create the RDF repository using main memory. The application allows the user to register a dataset and convert it into RDF format if it is in tabular format. Once the datasets are loaded into the repository, they can be queried in the following ways.

1. *Ad hoc queries.* This allows the user to compose RDF-based query statements and issue them directly to the data repository. Currently, it allows the user to use the following query languages: RQL, SeRQL, and RDQL. This requires the user to be familiar with at least one of these query syntaxes as well as the structure of the RDF datasets to be queried. SQL users should find it easy to learn RDF query languages.
2. *Form-based queries.* While ad hoc RDF queries are flexible and powerful, users who do not know RDF query languages would prefer to use an alternative method to pose queries. Even users who are familiar with RDF query languages might find these languages arcane to use. To this end, the application allows users to query the repository through web query forms (although they are not as flexible as the ad hoc query approach). To create these query forms, YeastHub provides a query template generator. Figure 7.7 shows the web pages that allow the user to perform the steps involved in generating and saving the query form. First, as shown in Figure 7.7(a), the user selects the datasets and the properties of interest. After the selection, the user proceeds to specify how to generate the query form template, as shown in Figure 7.7(b). This page requires the user to indicate which properties are to be used for the query output (select clause), search Boolean criteria (where clause), and join criteria. In addition, the user is given the option to create a text field, pull-down menu, or select list (in which multiple items can be selected) for each search property. Once the entry is complete, the user can go ahead to generate the query form by saving it with a name (all this information is stored as metadata in a MySQL database). The user



**Table 7.1. Types of databases and data distribution formats.**

	Tabular	XML	RDF	Rel. DB
Global Databases (GB/TB)		BIND		UniProt
Boutique Databases (MB/GB)	SGD, YGDP, MIPS	GO		TRIPLE
Local Databases (KB/MB)	Protein Chips, Protein-Protein Interactions			

can then use the generated query form, as shown in Figure 7.7(c), to perform Boolean queries on the selected datasets. Notice that the user who generates the query is not necessarily the same person who uses the form to query the repository. Some users may just use the query form(s) generated by someone else to perform data querying. These users may not have the need to create query forms themselves.

Presently, both types of queries return results in HTML format for display to the human user. Other formats (e.g., RDF format) can be provided.

### 7.3.1 Example queries

Our example queries involve integrating datasets obtained from different web-accessible databases. Table 7.1 lists these databases. In addition to showing the data distribution formats, it categorizes the databases into the following types.

1. *Global databases* represent very large repositories typically consisting of gigabytes or terabytes of data. These databases are widely accessed by researchers from different countries via the Internet. The example here is the yeast portion of UniProt in RDF format.
2. *Boutique databases* are large databases with typical sizes ranging from several megabytes to hundreds of megabytes (or even several gigabytes). Examples include SGD, YGDP,

Data source	Description	Select data source	Properties
GO	Gene ontology annotation	<input checked="" type="checkbox"/>	accession synonym name
SGD Gene Assoc.	SGD Gene Association	<input checked="" type="checkbox"/>	DB_Object_Symbol Outlier GO_ID

(a)

Template name: test23

**Properties to display:**

[SGD Gene Assoc.]DB\_Object\_Symbol  
 [SGD Gene Assoc.]GO\_ID  
 [GO]accession  
 [GO]name

**Properties to search:**

[SGD Gene Assoc.]DB\_Object\_Symbol Textfield  
 [GO]name Textfield  
 [SGD Gene Assoc.]DB\_Object\_Symbol Textfield  
 [SGD Gene Assoc.]DB\_Object\_Symbol Textfield

**Properties to be joined:**

[SGD Gene Assoc.]GO\_ID = [GO]accession

(b)

Template name: test23

**Properties to search:**

[GO]name =   
 [SGD Gene Assoc.]DB\_Object\_Symbol =

**Properties to display:**

[SGD Gene Assoc.]DB\_Object\_Symbol  
 [SGD Gene Assoc.]GO\_ID  
 [GO]accession  
 [GO]name

Query language:  RQL  SeRQL

(c)

Figure 7.7. (a) Selection of data sources and properties for creating a query template. (b) Query template generation. (c) Generated query form template.

MIPS, BIND, GO, and TRIPLES. While SGD and MIPS datasets are typically available in tabular format, GO and BIND are available in XML format. TRIPLES is a relational database.

3. *Local databases* are relatively small databases that are typically developed and used by individual laboratories. These databases may range from several kilobytes to several (or tens of) megabytes in size. Examples include a protein-protein interaction dataset extracted from BIND and a protein kinase chip dataset. While global and boutique databases are mostly Internet-accessible, some local databases may be network-inaccessible and may involve proprietary data formats.

Example Query 1: Figure 7.8 shows a query form that allows the user to simultaneously query the following yeast resources: a) essential gene list obtained from MIPS, b) essential gene list obtained from YGDP, c) protein-protein interaction data [208], d) gene and GO ID association obtained from SGD, e) GO annotation and, f) gene expression data obtained from TRIPLES [110]. Datasets (a)-(d) are distributed in tab-delimited format. They were converted into our RDF format. The GO dataset is in an RDF-like XML format (we made some slight modification to it to make it RDF-compliant). TRIPLES is an Oracle database. We used D2RQ to dynamically map a subset of the gene expression data stored in TRIPLES to RDF format.

The example query demonstrates how an integrated query can be used to correlate between gene essentiality and connectivity derived from the interaction data. The hypothesis is that the higher its connectivity, the more likely that the gene is essential. This hypothesis has been investigated in other work [80, 197]. In the query form shown in Figure 7.8, the user has entered the following Boolean condition: `connectivity = 80, expression_level = 1,`

**Template name: triples\_test4**

**Properties to search:**

[TRIPLES Expr.]expression\_level = 1

[TRIPLES Expr.]clone\_id = Y182B10

[TRIPLES Expr.]growth\_condition = vegetative

[Prot-Port Int.]connectivity = 80

[Prot-Port Int.]orf =

**Properties to display:**

- [Prot-Port Int.]orf
- [Prot-Port Int.]connectivity
- [YGDP Genes]orf
- [MIPS genes]orf
- [SGD Gene Assoc.]GO\_ID
- [SGD Gene Assoc.]DB\_Object\_Synonym
- [GO]accession
- [GO]name
- [TRIPLES Expr.]growth\_condition
- [TRIPLES Expr.]clone\_id
- [TRIPLES Expr.]orf
- [TRIPLES Expr.]expression\_level

Query language:  RQL  SeRQL

Figure 7.8. Example integrated query form.

growth\_condition = vegetative, and clone\_id = V182B10. Such Boolean query joins across six resources based on common gene names and GO IDs. Figure 7.9 shows the corresponding SeRQL query syntax and output. The query output indicates that the essential gene (YBL092W) has a connectivity equal to 80. This gene is found in both the MIPS and YGDP essential gene lists. This gives a higher confidence of gene essentiality as the two resources might have used different methods and sources to identify their essential genes. The query output displays GO annotation (molecular function, biological process, and cellular component) and TRIPLES gene expression.

Example Query 2: This query demonstrates how to integrate the UniProt dataset with the yeast protein kinase chip dataset that captures the number of substrates that each kinase phosphorylates with an expression level  $> 1$ . Figure 7.10 shows the RQL query syntax and the output that gives the number of substrates phosphorylated by kinase “YBL105C” (level  $> 1$ ) as well as the functional annotation of the kinase. This protein is listed as essential in both MIPS and YGDP. In addition to connectivity, we might hypothesize that the more the number of substrates a kinase phosphorylates at a high level, the more likely that the kinase is essential.

### 7.3.2 Performance

Sesame allows a repository to be created using a database (e.g., MySQL), native disk, or main memory. We evaluate the performance of these approaches using example query 1 described previously. We run the same query twice against main memory, mySQL, and native disk repositories. Each repository stores the identical datasets with a total of  $\sim 800K$  triple statements.

```

SELECT DISTINCT ns0orf,ns0connectivity,ns4accession,ns4name,ns5growth_condition,
ns5clone_id, ns5expression_level
FROM
{source58640} ns1:orf {ns1orf},
{source58639} ns2:orf {ns2orf},
{source58638} ns3:DB_Object_Synonym {ns3DB_Object_Synonym},
{source58638} ns3:GO_ID {ns3GO_ID},
{source58636} ns4:name {ns4name},
{source58636} ns4:accession {ns4accession},
{source55396} ns5:orf {ns5orf},
{source55396} ns5:growth_condition {ns5growth_condition},
{source55396} ns5:expression_level {ns5expression_level},
{source55396} ns5:clone_id {ns5clone_id},
{source58642} ns0:connectivity {ns0connectivity},
{source58642} ns0:orf {ns0orf}
WHERE
ns0connectivity="80"
AND ns5expression_level="1"^^<http://www.w3.org/2001/XMLSchema#longInteger>
AND ns5clone_id="V182B10"^^<http://www.w3.org/2001/XMLSchema#string>
AND ns5growth_condition="vegetative"^^<http://www.w3.org/2001/XMLSchema#string>
AND ns0orf=ns1orf
AND ns1orf=ns2orf
AND ns2orf=ns3DB_Object_Synonym
AND ns3DB_Object_Synonym=ns5orf
AND ns3GO_ID=ns4accession
USING NAMESPACE
ns2=<http://mcd750.med.yale.edu/yeasthub/schema/schema58639.rdf> ,
ns3=<http://mcd750.med.yale.edu/yeasthub/schema/schema58638.rdf> ,
ns1=<http://mcd750.med.yale.edu/yeasthub/schema/schema58640.rdf> ,
ns0=<http://mcd750.med.yale.edu/yeasthub/schema/schema58642.rdf> ,
ns5=<http://mcd750.med.yale.edu/yeasthub/schema/schema_triples.rdf#> ,
ns4=<http://139.91.183.30:9090/RDF/VRP/Examples/schema_go.rdf>

```

ns0orf	ns0connectivity	ns4accession	ns4name	ns5growth_condition	ns5clone_id	ns5expression_level
YEL092W	80	GO.0005842	cytosolic large ribosomal subunit (sensu Eukaryota)	vegetative	V182B10	1
YEL092W	80	GO.0003735	structural constituent of ribosome	vegetative	V182B10	1
YEL092W	80	GO.0006412	protein biosynthesis	vegetative	V182B10	1

Figure 7.9. Syntax and output of example query 1.

```

SELECT kinase,ct, function
FROM {Prot} uniprot1:gene {gene1}, uniprot1:Gene {gene2},
      {gene2} uniprot1:locusName {kinase}, {Prot} uniprot1:annotation {annot1},
      uniprot2:Function_Annotation {annot2}, {annot2} rdfs:comment {function},
      {Kinase} ns0:Orf {orf}, {Kinase} ns0:Phos_count {ct}
WHERE gene1=gene2 and annot1=annot2 and kinase=orf and kinase="YBL105C"
USING NAMESPACE
uniprot1=urn:lsid:uniprot.org:ontology: ,
uniprot2=urn:lsid:uniprot.org:ontology: ,
rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# ,
rdfs=http://www.w3.org/2000/01/rdf-schema# ,
ns0 = http://mcdb750.med.vale.edu/veasthub/schema/schema14469.rdf

```

kinase	ct	function
"YBL105C"	"2"	"Required for cell growth and for the G2->M transition of the cell division cycle. Mediates a protein kinase cascade; it activates BCK1 which itself activates MKK1/MKK2."

Figure 7.10. Syntax and output of example query 2.

Table 7.2. Query performance.

Query run	Memory	MySQL	File
1	312ms	308ms	9929ms
2	306ms	44ms	11045ms

Table 7.2 shows the amount the time (in milliseconds) it takes for query execution for each repository type. Both the main memory and MySQL approaches take about the same amount of time on the first query run ( $\sim 300$ ms). On the second query run, the MySQL approach is 7 times faster than the main memory one due to a cache effect (the speed difference, however, is only a fraction of a second). The file-based approach takes the longest query execution time.

Table 7.3 shows the amount of time (in seconds) it takes to load an RDF-formatted UniProt data file, which contains yeast data only, into the three repositories. The file size is about 63 MB ( $\sim 1.4$  million triple statements). As shown in Table 7.3, the main memory approach has the best data loading performance, while the MySQL approach has the worst performance due to the overhead involved in creating data indexes. In conclusion, the main memory approach gives the best overall performance.

**Table 7.3. UniProt data loading performance.**

Load run	Memory	MySQL	File
1	38ms	651ms	262ms
2	40ms	646ms	275ms

### 7.3.3 Implementation

YeastHub is implemented using Sesame 1.1. We use Tomcat as the web server. The web interface is written using Java servlets. The tabular-to-RDF conversion is written using Java. To access and query the repository programmatically, we use Sesame's Sail API that is Java-based. We use MySQL as the database server (version 3.23.58) to store information about the correspondences between the resource properties and the query form fields. Such information facilitates automatic generation of query forms and query statements. We also use the database server to create an RDF repository for performance benchmark as described previously. YeastHub is currently running on a Dell PC server that has dual processors of 2 GHz, 2 GB main memory, and a total of 120 GB hard disk space. The computer operating system is Red Hat Enterprise Linux AS release 3 (Taroon Update 4).

## 7.4 Discussion

Although the tab-delimited format is popularly used in distributing life sciences data, there are other data distribution formats such as the record format (or the attribute-value pair format), XML format, other proprietary formats. It would be logical to incorporate these formats into our RDF data conversion scheme. In the process of our RDF data conversion, we generate the corresponding RDF schemas. While our approach to generating new schemas allows existing properties that are defined in other schemas to be reused,



there is a need to perform schema mapping at a later stage, as new standard RDF schemas will emerge. How to translate one RDF schema into another RDF schema would be an interesting semantic web research topic.

While URL's are commonly used as a means to identify resources on the web, they have the following problems.

1. The web server referenced by the URL may be broken or become unavailable. Also, when a new server replaces the old one, the URL may need to be changed.
2. The syntax of the URL may change over time as the underlying data retrieval program evolves. For example, parameter names may be changed and additional parameters may be required.
3. The data returned by a URL may change over time as the underlying database contents change. This creates a problem for researchers when they want to exactly reproduce any observations and experiments based on a data object.

To address these problems, the Life Science Identifier project (<http://www-124.ibm.com/developerworks/oss/lcid/>) has proposed a standard scheme to reference data resources. Every LSID consists of up to five parts: the Network Identifier (NID); the root DNS name of the issuing authority; the namespace chosen by the issuing authority; the object id unique in that namespace; and finally an optional revision id for storing versioning information. For example, “urn:lsid:ncbi.nlm.nih.gov:pubmed:12571434” is an LSID that references a PubMed article. Each part is separated by a colon to make LSIDs easy to parse. The specific details of how to resolve the LSID to a given data object is left to an LSID issuing authority. In our case, we can potentially implement an LSID resolution

server (or LSID issuing authority) for referencing data objects stored in our semantic web data warehouse.

To increase the performance of data querying and loading, we use the main memory approach to build the RDF repository. For large amounts of data, we may use the relational database or native disk repository for data archival purposes and load the datasets of interest from the archival repository into the main memory repository for speed performance. Also, if we have a computer cluster, a parallel main memory architecture may be used to allow multiple main memory repositories to be queried concurrently.

While RDF-based query languages are SQL-like, there are SQL features that have not been implemented in Sesame yet. For example, not all RDF query languages (e.g., RQL) support outer-join like queries. In other words, if any of the properties included in a join query have no values, all the corresponding triple statements will be omitted from the query results. To get around this problem, our RDF data format includes property tags that have no data values.

Also, it would be useful to implement the aggregate functions (e.g., sum and average using GROUP BY). Sesame currently does not support delete and update queries, although these operations can be performed using some programmatic graph interfaces. Another limitation is that Sesame does not have a way to identify the source of triples (statements) once they are loaded into the repository. This makes removal of triples from a repository difficult if the triples come from different RDF files and have overlapping namespaces. SPARQL is a new RQL standard addressing these issues (<http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/>).

To enhance the knowledge representation and inference capability of the semantic

web, OWL (<http://www.w3.org/TR/owl-xmlsyntax/>), which is an extension of RDF, has emerged as an XML-based web ontology language. Support of reasoning using OWL is being incorporated into some RDF stores (e.g., Sesame and Tucana). This allows such RDF stores to transit from being data stores to becoming knowledge stores. There are questions (e.g., planning, explanation, and prediction) that cannot be answered by traditional database queries. However, they can be addressed by the kind of representation and reasoning provided by an ontological language such as OWL. This has been demonstrated in the context of reasoning about signaling network data [14]. Our work also represents a step in this direction.

# Chapter 8

## Web 2.0

### 8.1 Introduction

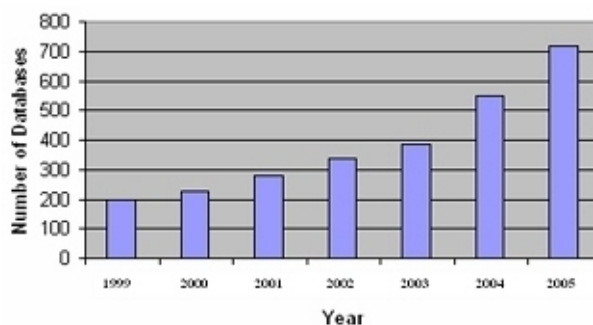
Web 2.0 refers to a second generation of Internet-based services - such as social networking sites, wikis, communication tools, and folksonomies - that emphasize online collaboration and sharing among users (<http://www.paulgraham.com/web20.html>). If the first generation Web has revolutionized the way people access information on the Internet, Web 2.0 has revolutionized the way people communicate across the Internet. Web 2.0 has transformed the Web into an environment that provides richer user experiences by allowing for the combination of disparate information in a variety of data formats, the facilitation of interaction between multiple parties, and the collaboration and sharing of information. Web 2.0 consists of a variety of applications implemented using diverse technologies. In general, the variety of Web 2.0 applications can be classified as follows:

- *Rich Internet applications.* These applications behave very much like desktop applications, and are easy to install and easy to use. In particular, they provide a dynamic interface with interactive features like point-and-click/drag-and-drop. These interfaces are achieved with technologies such as Ajax (Asynchronous JavaScript and XML) (<http://en.wikipedia.org/wiki/AJAX>), and mini plug-in programs known variously as widgets, gadgets and snippets, which create a programming environment within the browser and allow the user to easily combine information and create a variety of graphical presentations. As a result of this progress, the gap between Web programming and desktop programming has been diminishing ([http://blogs.adobe.com/shebanation/2007/02/desktop\\_application\\_programmin.html](http://blogs.adobe.com/shebanation/2007/02/desktop_application_programmin.html)).
- *Collaboration tools.* These include asynchronous collaboration tools such as wikis and blogs, to which users do not need to be simultaneously connected at any given time to collaborate. This category also includes synchronous, real-time (or near real-time) collaboration enablers, such as leading-edge instant messaging tools.
- *User-contributed content databases.* These are large-scale environments - such as YouTube, a video posting Web site, and Flickr, a photo-sharing site - in which users share content in multimedia format.
- *Integrative technologies enabling the Web as a platform.* There are abundant services and data sources scattered over the Internet. While they may be accessed independently, it has been exceedingly challenging to integrate Web-based services to create novel functionality. Web 2.0 mashup offers a solution to this problem. Mashup tools like Yahoo! Pipes (<http://pipes.yahoo.com/pipes/>) offer a graphical workflow editor that allows the user to pipe Web resources together easily. Other tools like Dapper (<http://www.dapper.net>) provide an easy way for users to ex-

tract (or scrape) Web contents displayed in heterogeneous formats and output the extracted contents in a standard format such as tab-delimited values and XML. Data visualization tools like Google Maps (<http://maps.google.com>) and Google Earth (<http://earth.google.com>) offer a GIS (Geographic Information System) interface for displaying and combining geographically related data. Despite their different functionalities, these tools may interoperate. For example, the output of Dapper may be fed into Yahoo! Pipes, and Yahoo! Pipes in turn can be linked to Google Map to process and display geographical data.

The popularity of the Web [19] and the success of the Human Genome Project (HGP) [32] have led to an abundance and diversity of biomedical data available via the Web. Figure 8.1 indicates the rate of growth in the number of Web-accessible biological databases that were published in the annual Database Issue of Nucleic Acids Research (NAR) between 1999 and 2005. These databases (which only represent a small portion of all biomedical databases in existence today) play an indispensable role in modern Health Care and Life Sciences (HCLS) research. They facilitate data mining and knowledge discovery [60]. The benefits for integrating these databases include the following:

- HCLS data are more meaningful in context, while no single database supplies a complete context for a given HCLS research study.
- New hypotheses are derived by generalizing across a multitude of examples from different databases.
- Integration of related data enables validation and ensures consistency.



**Figure 8.1. Number of databases published in the NAR Database Issues between 1999 and 2005.**

Via a Web browser, an HCLS researcher may easily access diverse information including DNA sequences, biochemical pathways, protein interactions, functional domains and annotations, gene expression data, disease information, and public health data. Integrating such data from diverse sources, however, remains challenging. Researchers wishing to analyze their own experimental data in combination with publicly available data face the cumbersome tasks of data preprocessing and cleaning [61], which includes scraping Web pages, converting file formats, reconciling incompatible schemas, and mapping between inconsistent naming systems. Even experienced programmers find such data integration tasks daunting and tedious.

A variety of approaches, including data warehousing [114, 166], database federation [84, 166], and Web services [179, 193], have been developed to facilitate data integration in the context of HCLS. One problem with these approaches is that they require their developers to have significant database/programming expertise. Moreover, these systems may not be able to anticipate or offer the flexibility needed by the end users (who may themselves not be well versed programmers). Furthermore, it is difficult if not impossible for these systems

to keep up with the growth of Web data sources. There are very few such systems that allow the user to add new external data sources easily, especially ones that do not conform to standard data formats.

To address these problems, Web 2.0 mashups have emerged. A mashup is a Web application that combines multiple third-party services over the Web. Numerous mashup examples are available from [www.programmableWeb.com](http://www.programmableWeb.com). Most of the current mashups are for non-scientific use. The potential of data mashup in the HCLS domains has only recently been demonstrated by using Google Earth to geographically integrate and visualize different types of data, including epidemiological and public health data, to help track the global spread of avian influenza [30]. In this study we provide more use cases to demonstrate how Web 2.0 mashups can be of potential use to HCLS researchers.

## 8.2 Mashup scenarios

We provide three scenarios that illustrate the use of several Web 2.0 mashup tools and sites to implement data integration in the HCLS domains. The first scenario, within a life sciences context, shows how to use Dapper and Yahoo! Pipes to integrate diverse data such as microarray measurements and gene annotation data. The second and third scenarios, within public health contexts, demonstrate how to geographically correlate cancer data with environmental data using Yahoo! Pipes, Google Maps, and GeoCommons (<http://www.geocommons.com/>), and to use these tools to predict the risk of West Nile Virus infection in different locations at different time, respectively.



### 8.2.1 Life sciences scenario: annotating microarray data

Figure 8.2 shows the workflow of a typical research study featuring the use of a spotted microarray, one kind of microarray technology. As shown in the figure, two biological samples (normal vs. disease), which consist of quantitatively distinct distributions of mRNA sequences, are labeled with fluorescent dyes. Sequences transcribed from the disease sample mRNA are labeled with the red fluorescent dye and sequences transcribed from the normal sample mRNA are labeled with the green fluorescent dye. Next, the two labeled samples are mixed in equal total amount, and that mixture is allowed to “hybridize” (bind) to the affixed reference sequences that have been deposited on the surface of a chemically treated microscopic glass slide. Each spot on the slide contains many strands of the DNA sequence corresponding to one specific gene. A large number of spots, and therefore many gene sequences, may be featured on a given slide.

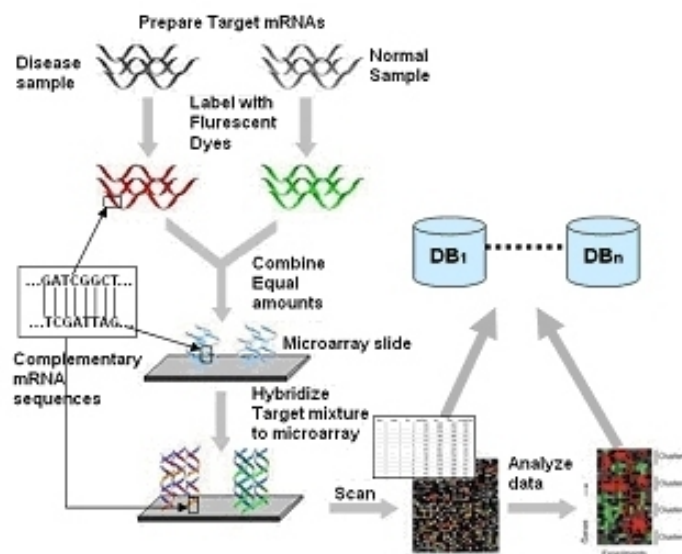


Figure 8.2. A typical research workflow that involves the use of microarrays.

After hybridization is complete, the slide is scanned by a laser scanner that measures the amount of each dye at the scale of 5-10  $\mu\text{m}$  pixels. Associated image processing software assembles the pixels into an image consisting of spots whose average pixel intensity values convey levels of gene expression. The color of a spot indicates how much the corresponding gene expresses in the disease sample relative to the normal sample. For example, a red or green spot means, respectively, that the gene is primarily expressed in the diseased or normal sample. If a spot is yellow, it means that the gene is equally expressed in both samples. If a spot is black, it means that the gene is not expressed or only meagerly expressed in both samples.

The imaging software processes the image data to produce a spreadsheet file of quantitative measurements of the image. This file, which contains rows corresponding to genes and columns corresponding to different types of measurements such as red intensities, green intensities and ratios, may be subjected to data analyses for statistical interpretation of the results. Such interpretation gains dramatically more meaning if the numerical output is integrated with known biological knowledge (e.g., gene annotation); yet such knowledge is frequently provided by diverse continuously-updated databases that are difficult to couple to the image outputs.

In our scenario, we integrated data from two Web sites, one hosted at Yale University, and the other at the BROAD Institute [53]. The Yale site provides microarray data generated from microarray experiments studying the gene expression profiling of *Neurospora crassa*, a red bread mold. The data are presented in the form of a tab-delimited file, with the columns describing different properties of the spots of a microarray slide, including their locations, gene identifiers, and mRNA sequences. To find current information about each of the genes listed in this file, one may go to the BROAD Institute site to search for the

gene annotation in its *N. crassa* database. An example search and the corresponding search results are illustrated in Figure 8.3, where the gene identifier NCU06658.1 was used as the search term. The search result is a page containing assorted annotations of the gene, such as its name, chromosome number, and exact location in the genome.

The figure illustrates a web mashup. The top browser window displays a table of microarray data with the following columns: Plate Name, Row, Column, Sequence, Name, MCU, and counterpart (5' - 3'). The data includes rows for various NCU identifiers and their corresponding sequences. The bottom browser window shows the Broad Institute's Feature Search interface. The search term 'NCU06658.1' is entered in the 'Text Search' field. The search results page for 'NCU06658: predicted protein' is shown, providing detailed annotations such as 'Chromosome/Linkage: V', 'Gene Symbol', 'Synonyms', 'Gene Name: predicted protein', 'Gene Product Names: predicted protein', and 'Location: Contig 31: 290221-291064'. A 'Protein Families' section shows a 'SET domain' with a 'Start Step' table.

**Figure 8.3. Microarray data and gene annotation provided by two sites.**

Currently the most common way to perform this kind of data mashup is to write scripts (in languages such as Perl) to:

1. Parse the tab-delimited file and extract the gene identifiers.
2. For each identifier, construct a URL that corresponds to the search result page of the

gene, and retrieve the content of the page.

3. Parse the result page to extract the data fields of interest.
4. Merge the extracted data fields with the original tab-delimited file to produce the integrated dataset.

This traditional approach has a number of shortcomings:

- Parsing HTML pages, especially those with potentially minor formatting discrepancies, is difficult and error-prone.
- The scripts may not be easily updated when there are changes to the data sources.
- It is difficult to reuse and share the scripts among different researchers. For instance, it is very common that when a graduate student or a postdoctoral fellow leaves a laboratory, the scripts written by him/her are not sufficiently documented for others to understand. In many cases other members of the laboratory resort to rewriting the scripts from scratch when the old ones fail to work due to changes at the data source side.

As we will discuss later in this chapter, an ultimate solution to these problems involves standardizing data formats and adding semantic annotations, so that machines could process the data in a largely automated way. Yet before such semantically rich data are widely available, it is desirable to have some semi-automatic tools that facilitate data integration while minimizing the above issues. We have found that some Web 2.0 tools, such as Dapper and Yahoo! Pipes, serve this purpose well. Here we describe how such tools were used to perform the above data mashup task easily.

The parsing of HTML pages was handled by the Web tool Dapper. Use of the tool consisted of two phases: learning and applying. In the learning phase, Dapper took the search result pages of some genes as input (Figure 8.4, step 1), and asked the human trainer to mark on the screen the parts of the content that corresponded to the data fields of interest (step 2, with the Gene Name field selected as an example). The gene identifier was set as a query parameter that would be changed dynamically for different genes (step 1, green box). Using some machine learning algorithms, the back-end system of Dapper then learned the locations of the data fields in the HTML pages from the examples. The resulting product, called a “dapp”, was the data extraction proxy of the BROAD Institute site. In the applying phase, when the dapp was presented a new gene identifier, it extracted the corresponding data values of the gene from the site and output them in standard XML format (Figure 8.4, step 3).

The dapp was then used as a data source to be integrated with the Yale tab-delimited file using Yahoo! Pipes, which is a tool that treats data as “water” flowing in “pipes”. It allows users to use different widgets to process their data, connecting the widgets using metaphorical pipes.

As shown in Figure 8.5, the Yahoo! Pipes tool has three panels: library, canvas, and debugger. The library panel lists categories of widgets that allow functions such as data fetching, filtering, and manipulation. The canvas panel allows the selected widgets to be placed, moved, and connected. The debugger panel is below the canvas panel, and it displays the output or error messages when the pipe is executed. The specific pipe used for our data mashup task is shown in the canvas panel. It starts with a “Fetch CSV” widget to fetch the tab-delimited data table from the Yale site. The output of the widget is piped to a “Truncate” widget for limiting the total number of rows in the result, which we set as 10 for

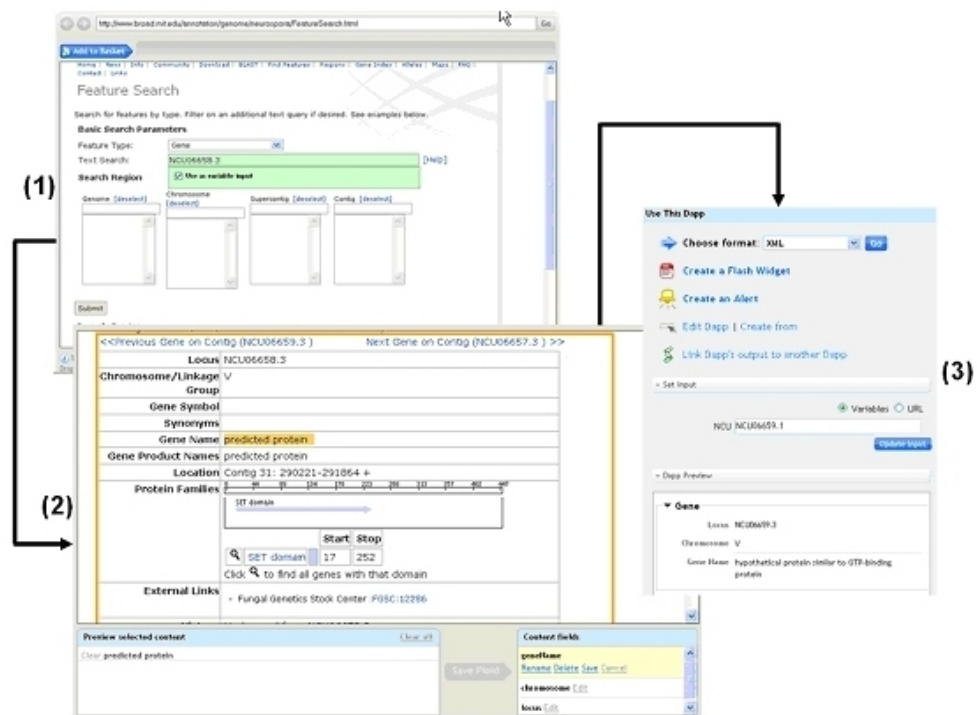
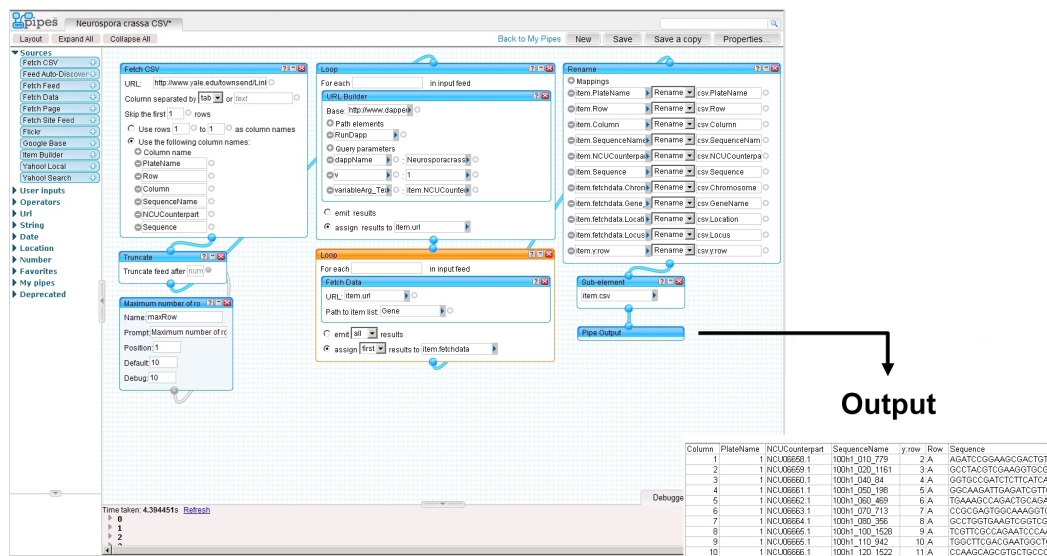


Figure 8.4. A Dapper interface for querying and retrieving gene annotation.

demonstration. Then we used a “Loop” widget to iterate through each row to construct a URL to the dapp using the gene identifier, and another “Loop” widget to actually retrieve the content of the dapp output. Finally, all unwanted fields were filtered and the dataset was output as a comma-separated-value (CSV) file.



(a)

(b)

Column	PlateName	NCUCounterpart	SequenceName	y.row	Row	Sequence
1	1	NCU06658	100h_010_779	2	A	JAGTCCGGGAAGCGACTGT
2	1	NCU06659	100h_020_1161	3	A	GCCTACGTCGAAGGTGTCG
3	1	NCU06660	100h_040_04	4	A	GDTCCGATCTTCATCA
4	1	NCU06661	100h_050_198	5	A	GGCAAGATTAGATCCTTC
5	1	NCU06662	100h_080_469	6	A	TGAAGGTCAGAGTCGAGA
6	1	NCU06663	100h_070_715	7	A	CCCGCAAGTGGCAAGGTG
7	1	NCU06664	100h_080_356	8	A	GCCTGGAAGTCCSTCC
8	1	NCU06665	100h_100_1520	9	A	TGTTTCGCCAAGTCCAA
9	1	NCU06666	100h_110_942	10	A	TGGTTCGAGATGCTTC
10	1	NCU06666	100h_120_1522	11	A	CCAAGCAGCGTCTGCC

**Figure 8.5. (a) A Yahoo! Pipe for mashup of microarray data and gene annotation and (b) integrated output.**

The whole mashup process did not involve any coding. The user interfaces of the two tools were simple and intuitive enough for non-programmers to use. The difficult task of HTML parsing was handled by dedicated learning algorithms of Dapper, which, compared to most custom scripts, requires much less work by the user.

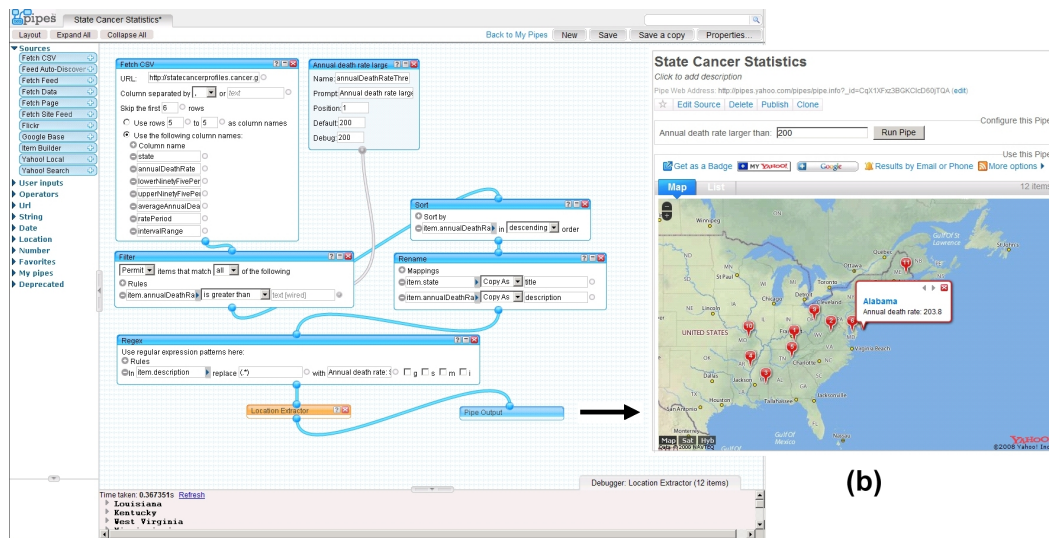
### 8.2.2 Public health scenario 1: correlating cancer and environmental factors

Environmental health epidemiologists study the association between human diseases (e.g., cancer) and environmental factors. Such studies often require the integration of disparate data sources such as population census, air quality and environmental pollution release, and health care utilization data. These different data streams are typically produced by different agencies. Automated integration of data from these agencies is limited due to a variety of political and technological challenges. Web 2.0 mashups offer the potential for automating the integration of disparate health care data to enhance environmental health research. As an example, we demonstrate how to use Yahoo! Pipes and a Web 2.0 site called “GeoCommons” to geographically correlate cancer data with water pollution data in the United States.

First, we identified a cancer profile dataset at the State Cancer Profiles Web site (<http://statecancerprofiles.cancer.gov/map/map.noimage.php>) developed by the National Cancer Institute (<http://gis.cancer.gov>). This tabular dataset contains annual death rates for all types of cancers in different US states (the year of this data collection is 2004). We created a pipe as shown in Figure 8.6(a) to fetch this cancer data table and applied a user-defined threshold against the annual death rates. The filtered output was fed to a “location extractor” widget that allows the states that have annual cancer death rates above the specified threshold to be displayed via Google Maps, as shown in Figure 8.6(b). The map was then exported to a KML file (a standard XML format for Google Maps/Earth).

We uploaded the KML file to the GeoCommons Web site (<http://www.geocommons.com>). This site allows users to annotate and publish their uploaded maps as well as mashup



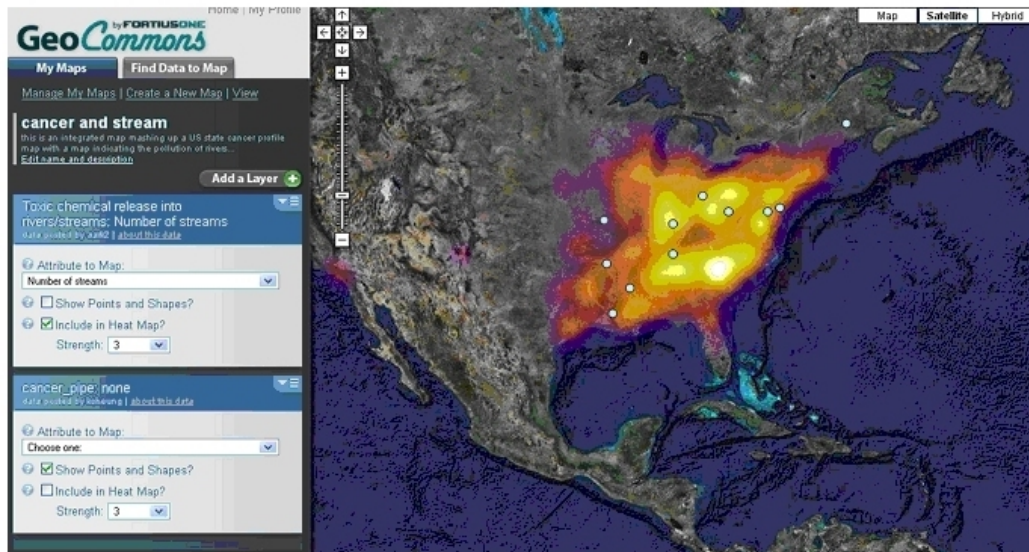


(a)

(b)

Figure 8.6. (a) A Yahoo! Pipe for filtering US state cancer profile data and (b) display the results using Google Maps.

the digital maps uploaded by other users. In this example, we found a “heat” map that details the number of polluted rivers/streams in the US. In a heat map, a brighter color corresponds to a higher number of polluted rivers/streams. Figure 8.7 shows a GeoCommons interface that allows the state cancer profile map to be superimposed with the water pollution map. We can see that most of the states with high cancer death rates are in the fire zone.



**Figure 8.7.** A mashup of the state cancer profile map and water pollution map.

### 8.2.3 Public health scenario 2: predicting West Nile Virus cases

We attempted to reimplement the work of Zou et al. [212] for predicting high risk WNV areas using a Web 2.0 approach and degree-day temperature calculations based on the single sine method [5]. Briefly, WNV is a mosquito-borne flavavirus that was originally discovered in the United States after an outbreak in New York City in 1999 [107, 139, 154]. The

infection can cause illness and death in humans and animals, including crows and horses. Transmission of the virus increases during the summer months since mosquito activity peaks in warmer weather [107]. Many different types of data streams are used for surveillance of West Nile Virus including human case reports, mosquito testing, dead bird sightings, dead bird or other wildlife testing, land use data, and temperature data. In this case report, we use Yahoo! Pipes, Dapper, GeoCommons [154], and Google Earth to create a grid-like application for predicting West Nile Virus (WNV) risk in humans.

We used publicly available data from three websites and integrated that data using two different web services to create the application. Human and animal WNV data were taken from the CDC ArboNet website (<http://diseasemaps.usgs.gov>). The ArboNet website provides the number of WNV cases in five types of organisms at the county level, as well as statewide accumulated totals. We focused on bird and human cases aggregated at the statewide level and built a dapp to locate the total number of WNV cases on the USGS site. Then, using a list of state abbreviations as well as latitudes and longitudes from GeoCommons, we built a Yahoo! Pipe to loop over the states, extract the accumulated totals, and combine them with the geographical locations to produce an output file in Keyhole Markup Language (KML). Temperature data were taken from the National Climate Data Center (NCDC, <http://www.ncdc.noaa.gov/oa/ncdc.html>). The NCDC provides monthly climatic data from each weather station; the stations are broken down into four lists on the NCDC web site. We built a Yahoo! Pipe to aggregate the station IDs and locations into a single list. Then, we extracted a sub-list of stations by state and downloaded the climate data from each station for a particular year (Figure 8.8). Calculation of transmission risk was refined based on degree-day calculations. The degree-day is a measurement of heating or cooling in a given area and is calculated as the difference between the mean daily

temperature and a pre-defined baseline temperature [57, 149]. For vector-borne diseases like WNV, it is used to determine the temperature threshold for which viral transmission can occur. Degree-day calculations were performed by a web service published by the UC Davis Statewide Integrated Pest Management Program (IPM, <http://www.ipm.ucdavis.edu/>). A customized accumulator was built to process the IPM output file and to perform a sliding-window accumulation to account for the limited infection period of a mosquito. Then, the maximum of the accumulated degree-days was computed by Yahoo! Pipes and compared to the threshold required for median viral transmission in order to predict WNV risk at the station for that year. These predictions were then combined with the geographical information to produce a KML file for all the stations in a particular state. We chose two states, Delaware and New Jersey, to use in our case report; any state with potential for WNV transmission could have been included.

For visualization, each state has a separate KML file that can then be overlaid into Google Earth. This allows public health researchers to visualize the predictions from the temperature data alongside the actual number of human and animal cases. We repeated this procedure for years 2006 and 2007.

During development, we had to write computer programs to address performance issues with the Web 2.0 applications. First, we built a server-side cache to store the content from the different data sources in order to avoid overloading the servers. This process is similar to the caching systems of web browsers, which store web content on the local disk and serve users with these cached copies instead of repeatedly fetching from the Web. With multiple users potentially running our pipeline and accessing the same web pages, it is more effective to have a second-level caching at the server side. Yahoo! Pipes does have its own internal caching system, but since we had no control of its properties, including expiration time

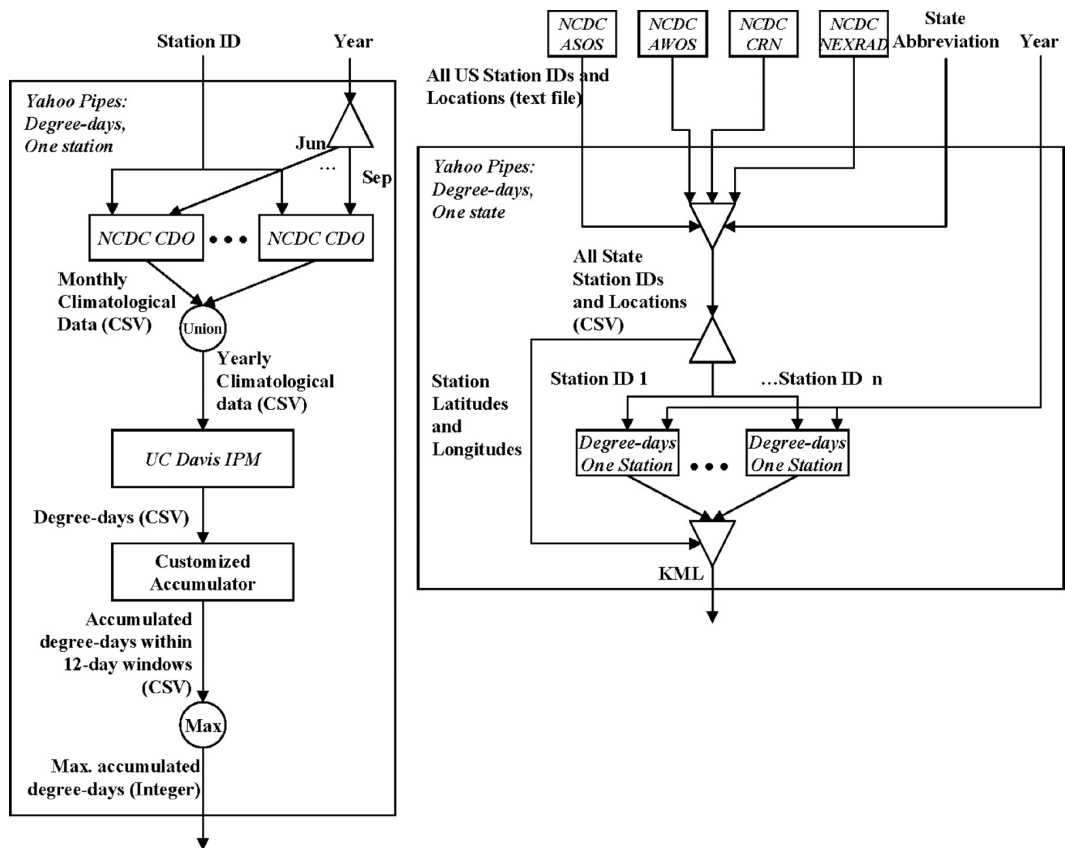


Figure 8.8. Schematic diagram illustrating how the developers combined different web-based data and analytical services to produce the application.

and maximum size, we also implemented a cache at our local server. In addition, we wrote three small programs to connect different components of the pipeline. The first program extracted the properties of each weather station (ID, name, latitude and longitude) from the text files provided by the NCDC site. These files use fixed column lengths to separate different data fields instead of the more common use of delimiters. Since this file format is not currently supported by Yahoo! Pipes, we had to write our own parsers. The second program was used to submit temperature data to the IPM site for calculating degree-days. The IPM site provides two methods for data input: a text form for entering data on screen and the ability to submit a data file through the HTTP Post method. We were unable to use the first method as there were not enough input boxes for entering a whole month of data. Meanwhile, the data retrieval modules of Yahoo! Pipes only support the HTTP Get method, not Post. We therefore wrote our own connector for posting the data file prepared by Yahoo! Pipes to the IPM site. The third program processed the degree-days output of the IPM site and sent it back to Yahoo! Pipes. It used a sliding window to calculate the accumulated degree-days within each window.

The application was developed in one and a half months. In addition to application development, this time included research to find available Web 2.0 resources as well as project design. The KML maps are displayed in Google Earth for 2006 and 2007 (Figures 8.9 and 8.10); they show the total number of human and bird cases and a label for each weather station. Next to each weather station is a '+' or '-', indicating whether the degree-day was above or below the assigned threshold. For 2006 and 2007, both states have every weather station above the threshold (all have a '+'), which suggests that the weather in each state supports viral transmission. However, in New Jersey, the number of human cases decreases from 2006 to 2007 while the number of bird cases remains relatively the same. In Delaware,

the number of bird cases decreases slightly.



Figure 8.9. WNV and temperature data for Delaware and New Jersey in 2006.

### 8.3 Strengths and weaknesses

In this section, we discuss the general strengths and weaknesses of Web 2.0 mashup technologies based on our current experience in using them to integrate HCLS data. We

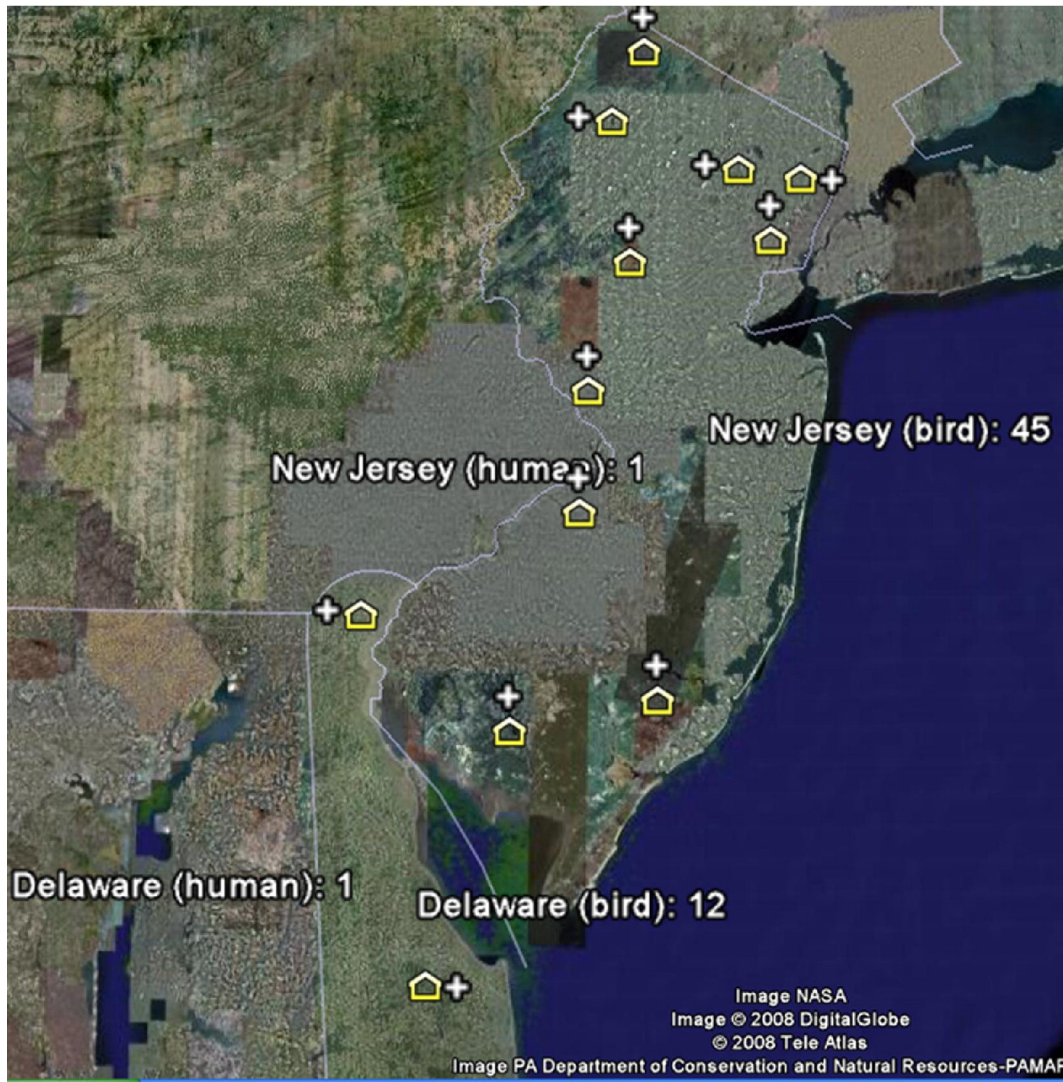


Figure 8.10. WNV and temperature data for Delaware and New Jersey in 2007.



have identified the following strengths:

- *Applicability.* The tools that we used in the mashup examples are useful for diverse areas of biomedical research. For example, Yahoo! Pipes supports a great variety of input and output data types that biomedical researchers need to deal with, from the most popular tab-delimited format to structured XML and semantically rich RDF. Common mashup tasks such as data integration by means of ID mapping can be performed without coding.
- *Ease of use.* As demonstrated by the mashup examples, tools like Dapper and Yahoo! Pipes provides an easy-to-use Web interface for extracting and integrating data from diverse sources. Extraction and integration with these intuitive tools is easier than writing code in a particular programming language (e.g., Perl) to parse and integrate data.

The tools in general have intuitive designs that require little learning time for beginners. New users are also greatly assisted by the active user community in solving their technical problems through reading or joining in related discussions at designated online message boards.

- *Reusability and extensibility.* Web 2.0 mashup tools like Yahoo! Pipes and Dapper are designed for sharing and reuse. For example, the Yahoo! Pipes site allows its users to describe and publish their pipes. Through its “Show off your Pipe” message board, users can comment and rank each other’s pipes. In addition, the shared pipes can be easily extended or modified by others to add new features. For instance, it is straightforward to take components from several publicly shared pipes to form a new, customized pipe.

- *Interoperability.* As shown in our examples, different Web 2.0 tools can be easily combined to enhance the mashup capability. For example, Yahoo! Pipes can be complemented by Dapper by allowing fetching of data in formats that are not supported by Yahoo! Pipes. In addition, Dapper provides an Application Programming Interface (API) that allows Web services for searching the dapps and software development toolkits (e.g., in Perl and Java) for accessing dapps programmatically.
- *Active roles of users.* Web 2.0 applications emphasize the active participation of users in reporting bugs, suggesting new functions, or even implementing new features through specific software development kits (SDK). These activities facilitate the improvement of applications much more rapidly than in traditional software engineering paradigms.

In spite of these strengths, we have experienced and would note several issues that arise in creating data mashups using the tools.

- *Missing features and instability.* Tools like Yahoo! Pipes and Dapper are relatively new, and are still under active development. Since many of these tools were initially designed for casual lightweight mashup tasks such as aggregating news feeds from a small number of Web sites, their designs did not incorporate a breadth of computational theory. For example, while Yahoo! Pipes provides operations commonly found in database query languages, such as selection and renaming, some other essential operations such as column selection (i.e., “projection” in database terms) and table-joining are currently either not supported or supported only in arcane ways. Many such features are needed in order for these tools to be widely adopted for daily research activities.

Additionally, these new tools still contain bugs. In particular, due to the heavy use of client-side scripting (e.g., JavaScript), these tools are especially prone to errors that arise from the many brands and versions of browsers that are in use today but not completely compatible. Moreover, as with any Web servers, a Web 2.0 site may become unreachable without prior warnings.

- *Performance and scalability.* Given the distributed nature of the Web and the limited speed of the network connections, mashing up large datasets from different sources can be very slow. We encountered this problem when attempting to integrate the whole microarray data table (consisting of tens of thousands of rows) with the corresponding annotation data. There was a timeout when we executed the pipe for the entire table. The largest number of rows that we were able to integrate successfully using our pipes was around 1500, and the task took about 1.5 min to run. In comparison, with all the datasets stored locally, integrating tens of thousands of rows should not take more than a few seconds using a customized script.
- *Security.* Most Web 2.0 sites do not have a strong security policy for their users. The users have to bear the security risks if they upload their data to these Web 2.0 sites. Although the user may choose not to publish their data to the public, he/she loses control of the data once the data are uploaded to a Web 2.0 server. The security is at the mercy of the person(s) in charge of the server security. Therefore, it is not recommended to use public Web 2.0 sites to share sensitive/confidential data.
- *Flexibility.* Although the Web 2.0 tools are found to be very useful in our two data mashup scenarios, by nature they are not as flexible as customized scripts. There are always some special cases that the standard widgets cannot handle properly. One solution, which is already adopted by Yahoo! Pipes, is to allow users to

supply customized Web services as widgets. This is a promising approach in general, although standard Simple Object Protocol (SOAP) based Web services (<http://xml.coverpages.org/soap.html>) are still not yet supported.

- *Quality of final output.* Professional users are unlikely to switch to Web 2.0 tools until the aesthetic quality of the final graphical or tabular output matches the quality that may be achieved with local software.

For the West Nile Virus use case, we hypothesized that the use of grid-like applications and Web 2.0 technologies would facilitate the integration of public health data from diverse sources. However, in our experience this process was far from straightforward. Yahoo! Pipes was not able to handle fetches of large amounts of data due to timeout, internal caching, and synchronization errors. Some websites seemed to have been designed to confound mashup approaches. For example, the NCDC site limited our daily download of web pages to 100 pages, but it is unclear if there is an official daily access limit. When 100 pages were exceeded, the site returned an error page. Likewise, for the USGS site, when too many requests were issued using Dapper within a short period of time, the web server returned blank fields where WNV case count were supposed to appear. Further output pages in this situation were indistinguishable from the situation where no WNV cases were reported. Other problems included a lack of support for basic data management functions (aggregation, table joins, etc.) and limited support for conversion of data between various output forms.

We conclude that while this approach is feasible, the development effort was not significantly reduced from more conventional software engineering approaches. This was in part due to the lack of maturity of present mashup tools and in part due to design aspects of

two of the web sites that inhibited their use as impromptu web services. The design does illustrate the usefulness of grid-like computing approaches and Web 2.0 in public health and the value of web-based integration of data and analytical services.

## 8.4 HCLS 3.0

According to Spivacks ([http://novaspivack.typepad.com/nova\\_spivacks\\_weblog/2006/11/web\\_30\\_versus\\_w.html](http://novaspivack.typepad.com/nova_spivacks_weblog/2006/11/web_30_versus_w.html)), Web 3.0 refers to “a supposed third generation of Internet-based services - such as those using Semantic Web, microformats, natural language search, data-mining, machine learning, recommendation agents, and artificial intelligence technologies - that emphasize machine-facilitated understanding of information in order to provide a more productive and intuitive user experience.” Semantic Web (SW) technologies play a core role in this definition.

The World Wide Web Consortium (W3C) has launched the Semantic Web for Health Care and Life Sciences Interest Group (HCLSIG; <http://www.w3.org/2001/sw/hcls/>), which has been chartered to develop and support the use of SW technologies to improve collaboration, research and development, and innovation adoption in the HCLS domains [158]. One of the ongoing efforts involves converting a variety of HCLS data sources into the standard Semantic Web data formats endorsed by W3C: Resource Description Framework (RDF) (<http://www.w3.org/RDF/>) and Web Ontology Language (OWL) (<http://www.w3.org/TR/owl-ref/>) formats. While OWL is semantically more expressive than RDF (<http://www.w3.org/TR/owl-ref/>), OWL and RDF bear the same syntax. Datasets expressed in either format can be queried by the standard RDF query language - SPARQL (<http://www.w3.org/TR/rdf-sparql-query/>). For OWL datasets (ontolo-

gies), tools such as Pallet (<http://www.mindswap.org/2003/pellet/>), RacerPro (<http://www.racer-systems.com>), and Fact++ (<http://owl.man.ac.uk/factplusplus/>) can be used to perform OWL-based reasoning. At WWW 2007, a demonstration organized by the HCLSIG showed how to use SPARQL to query across a number of OWL ontologies in the Alzheimer's disease research context. In addition, Semantic Web applications such as YeastHub [39], SWAN [66], and BioDash [136] have already emerged in the HCLS domains.

While Web 2.0 offers human-friendly tools for mashing up data, the Semantic Web [20] better enables computers to help human users find and integrate information over the Internet, and to perform such activities in a more sophisticated way. As pointed out by Ankolekar et al. [8], Web 2.0 and Semantic Web are not two conflicting visions. They are, instead, complementary to each other. There is a potential benefit to mashing up Web 2.0 and Semantic Web in the context of HCLS. To implement the vision of Semantic Web, more datasets need to be converted into RDF/OWL formats. This conversion process may be facilitated by Web 2.0 tools that can be used to extract and aggregate non-SW content from numerous Web sites, producing data converted into RDF/OWL. Furthermore, Web 2.0 tools may be used to assist users to annotate a small amount of data. Such small annotated data sets may then be used as examples to train automatic annotation algorithms.

Currently, many Web 2.0 tools can process RSS feeds (which use a simple RDF structure). It would be desirable for these tools to be able to understand semantically richer formats like RDF Schema (RDFS) and OWL, thus supporting richer and possibly more intelligent integration. For example, SPARQL may be supported by future Web 2.0 tools for fetching, filtering, and aggregating RDF/OWL data sources. In addition, "RDF-attributes" or RDFa (<http://www.w3.org/TR/xhtml-rdfa-primer/>) has been proposed by W3C as an alternative to microformat for embedding ontological elements into existing HTML (more

precisely XHTML) documents, mashing up human readability and machine readability. It would be logical for future Web 2.0 tools (e.g., Dapper) to recognize RDFa, even though RDFa parsing tools like GRDDL (Gleaning Resource Descriptions from Dialects of Languages) (<http://www.w3.org/2004/01/rdxh/spec>) are available.

Figure 8.11 depicts an example demonstrating implementation of semantic mashup between existing Web pages using RDFa. On the left of Figure 8.11, a Web page of NeuronDB (<http://senselab.med.yale.edu/neurondb/>) shows the neuronal properties including receptors (e.g., GabaA and GabaB) and currents (e.g., I Potassium and I Calcium) located in different compartments (e.g., Dad, Dem and Dep) of the “cerebellar purkinje cell”. On the right of Figure 8.11, there are 2 linked Web pages of the Cell Centered Database (CCDB) (<http://ccdb.ucsd.edu>). The top Web page shows the different neuronal images for “purkinje neuron”, while the bottom page shows the detailed information about the “purkinje neuron”, including the region in the brain where the neuron is located. In this case, it is located in the “cerebellum”. Using RDFa, we can associate ontological fragments (in OWL format) with HTML elements. The OWL components (represented by dotted rectangles) corresponding to the circled HTML elements are shown in Figure 8.11. The semantic relationships are explicitly expressed using the OWL-DL syntax. For example, in CCDB, the class “PurkinjeNeuron” has a property named “region” whose value is “Cerebellum”. In addition, semantic mashup is achieved using the “equivalentClass” construct supported by OWL-DL. In this case, the NeuronDB class “CerebellarPukinjeCell” is equivalent to the CCDB class “PurkinjeNeuron” whose region property has the value “Cerebellum”.

To take the concept of RDFa further, we may entertain the possibility of extending it to work for any XML format rather than limiting its domain to XHTML. One main benefit

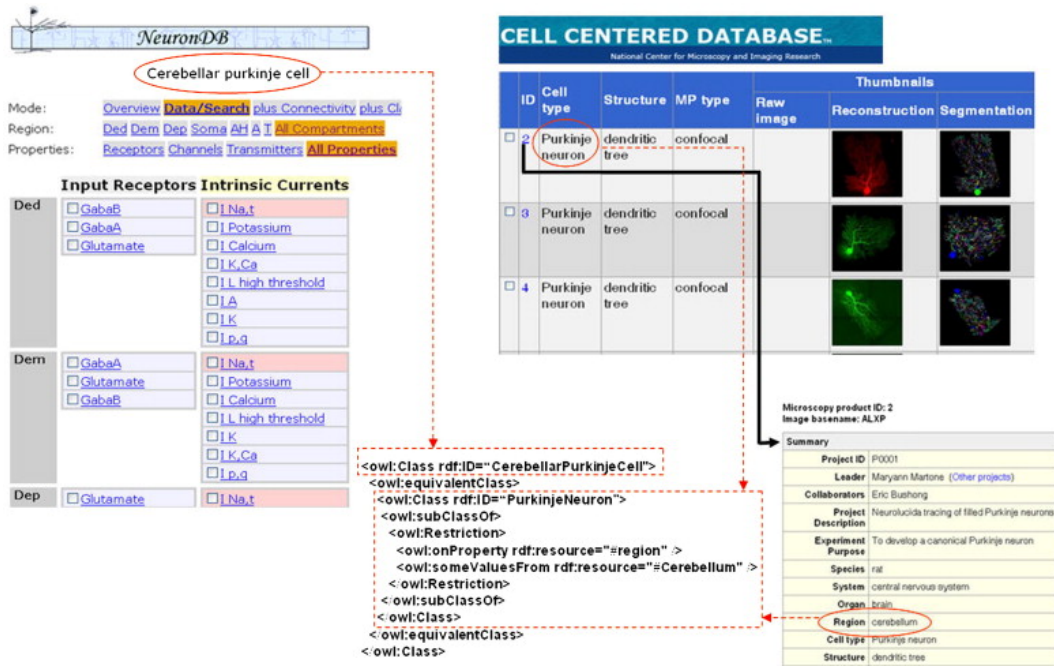


Figure 8.11. Semantic mashup between existing Web pages.

of such an extension is that existing visualization tools like Google Maps use XML as the input data format. Embedding ontologies in these XML formats would add a querying capability for ontology, while exploiting the visualization capability currently supported by existing tools. For example, if some geo-ontologies are integrated into Keyhole Markup Language (KML) (<http://code.google.com/apis/kml/>), geographic mashup by Google Maps/Earth may be performed in a fully semantic manner.

With regard to the cancer data mashup, we have encountered some cancer-related data that are tallied within geographic regions that exhibit different granularities. Some data may be collected at the city level, while other data may be collected at the county or state level. To support semantic mashup based on locations, one may define an ontology in which a city (e.g., North Haven) is located in a county (e.g., Greater New Haven), which is in turn



located in a state (e.g., Connecticut). Given such an ontology, location-based inference may be performed when mashing up data.

The Semantic Web community has been working with data providers to convert their data into RDF/OWL ontologies. While the ultimate goal is to come up with heavy-weight (semantically rich) ontologies for supporting sophisticated machine reasoning, it may be worthwhile to also provide coarser ontologies that can be easily incorporated into future Web 2.0 tools. Currently these tools use tags and folksonomies to annotate and categorize content. A mashup of folksonomy and ontology merits exploration. For example, popular tags may evolve into standard terms. In addition, taxonomic or hierarchical relationships may be identified among existing tags. This bottom-up approach may allow social tagging to evolve into the development of standard ontologies. This evolution is reflected by the transformation of social wiki into semantic wiki. Instead of tagging wiki pages based on user-defined terms, semantic wiki tools such as ontowiki (<http://ontowiki.net/Projects/OntoWiki>) allow users to semantically (ontologically) annotate Web pages. The semantic mashup scenario depicted in Figure 8.11 can potentially be achieved using semantic wiki as well. In this case, OWL-formatted metadata will be generated for facilitating semantic data mashup.

HCLS represents flagship domains in which SW applications may be developed and shown to be successful (<http://www.thestandard.com/internetnews/001301.php>). One possible direction for future work may be to develop SW applications that would provide the infrastructure to support semantic mashup of HCLS data in a user-friendly and social-friendly fashion. We therefore envisage a transformation from Web 2.0 mashup to Web 3.0 semantic mashup, producing a better synergy between human and computer.

## 8.5 HCLS 2.0 + HCLS 3.0 = e-HCLS

e-Science describes science that is increasingly done through distributed global collaborations enabled by the Internet, using very large data collections, large-scale computing resources, and high performance visualization (<http://e-science.ox.ac.uk/public/general/definitions.xml>). It involves two components: semantic components and social components. e-HCLS is e-Science within the HCLS context. While the Semantic Web has the potential to play an important role in the semantic representation of e-Science, Web 2.0 has the potential to transform from the so-called “me-Science” (<http://www.gridtoday.com/grid/963514.html>), that is driven by an individual researcher or laboratory, into what we call “we-Science”, which is driven by community-based collaboration. The mashup scenarios described in our paper shed some light on the potential impact of social networking on HCLS.

Our public health data mashup scenario has demonstrated the benefit of sharing data (maps) in the community. Once the data are shared in a standard format (e.g., KML), visualization and integration may be readily achieved. While different groups have independently created different maps (e.g., cancer profiles and environmental pollution) to meet their own needs, new insights or knowledge can be derived when these maps are mashed up. This mashup is made possible by providing a global information commons like Geo-Commons.

The microarray mashup scenario has illuminated the importance of data integration in data mining/analysis. Web 2.0 can potentially be used to create a social network that facilitates collaboration between microarray data providers and microarray data miners. In this case, via a microarray data commons (Web 2.0 site), data providers can publish their

datasets, while data miners can publish their data analysis algorithms/programs. This way, not only can the data providers search for the appropriate tools for analyzing their datasets, but the data miners may also search for appropriate datasets for testing their analysis methods. They may furthermore make comments about their experience of using certain datasets/tools. Lastly, they can use the site to publish analysis results and to allow others to make comments about them. Currently, public microarray Web sites such as Gene Expression Omnibus (GEO) [54] and ArrayExpress [28] do not support this type of social networking.

A number of social networking sites/projects have emerged, which are tailored to the needs of different HCLS communities. For example, Alzforum (<http://www.alzforum.org>) is a site that facilitates communication and collaboration within the Alzheimer's Disease (AD) research community. It also allows its members to comment on AD research articles and publish such comments. Connotea (<http://www.connotea.org>) is a free online reference management for all researchers, clinicians and scientists. myExperiment (<http://myexperiment.org>) is a beta tool that allows scientists to contribute to a pool of scientific workflows, build communities and form relationships. In contrast to traditional peer-reviewed publication, Nature Precedings (<http://precedings.nature.com>) is a site for researchers to share documents, including presentations, posters, white papers, technical papers, supplementary findings, and manuscripts. It provides a rapid way to disseminate emerging results and new theories, solicit opinions, and record the provenance of ideas. It would be interesting to see: (i) how these sites would enable discovery, creativity and innovation, and (ii) whether a larger social network can be formed if these social network sites are interoperable.

The Web 2.0/3.0 data mashup scenarios we have described are based on the assump-

tion that the data are publicly accessible without the concern about security. However, this concern becomes real when mashing up sensitive healthcare data such as medical administrative data including hospital discharge data, claims data, medical records, and so on. The ability to integrate medical administrative data from different sources is crucial to outcome research [122]. The access to these medical administrative databases is restricted to approved researchers. In addition, it is often a requirement that manipulation, analysis, and transmission of such data need to be done in a secure manner. Developers have begun to explore how to provide a secure mechanism for mashing up sensitive data. For example, IBM has recently announced *MASH!!L*, which is a new technology for supporting secure data mashup (<http://www.physorg.com/news124641823.html>).

## Part III

# Network Tools Developed for the Community

## Chapter 9

# tYNA: the Yale Network Analyzer

### 9.1 Introduction

As we have been emphasizing throughout this thesis, in the era of systems biology, the focus on understanding complex organisms is shifting from studying individual genes and proteins towards the relationships between them [168], and these relationships are usually expressed in terms of various kinds of biological networks. Recent developments of large-scale experiments such as mass spectrometry and array-based techniques [69, 108, 115] have generated rough descriptions of the complete networks in the cells. Many studies have reported interesting biological findings from these networks, including the relationships between various statistical properties of a gene and its function and essentiality, and the elucidation of controls at the molecular level based on network motifs [86, 115, 132, 169, 210].

These studies require heavy computations on multiple networks. We have developed a Web system, tYNA (the Yale Network Analyzer, <http://tyna.geresteinlab.org>), to

provide researchers with a set of tools to carry out such computations with great ease. The system provides five main types of functionality: (1) Network management: storing, retrieving and categorizing networks. A comprehensive set of widely used network datasets is preloaded, put into standard form, and categorized with a set of tags. (2) Network visualization: displaying networks in an interactive graphical interface (Figure 9.1). (3) Network comparison and manipulation: various kinds of filtering and multiple network operations. (4) Network analysis: computing various statistics for the whole network and subsets, and finding motifs and defective cliques. (5) Network Mining: predicting one network based on the information in another.

Our system shares some elements with some other network analysis and visualization systems, such as Cytoscape [167], JUNG<sup>1</sup>, N-Browse<sup>2</sup>, and Osprey<sup>3</sup>, but also offers some additional features such as defective clique finding. Besides, being a Web-based system, tYNA also has some unique advantages:

- Users can share networks through a centralized database.
- Computationally intensive tasks such as motif finding and statistics calculations can be performed on powerful servers.
- The system can be linked from/to other online resources.
- Users can incorporate some functions of tYNA into their own programs using the SOAP-based web service interface.

Table 9.1 summarizes some major differences between the systems. We do not attempt

---

<sup>1</sup><http://jung.sourceforge.net/>

<sup>2</sup><http://nematoda.bio.nyu.edu:8080/NBrowse/N-Browse.jsp?last=false>

<sup>3</sup><http://biodata.mshri.on.ca/osprey/>

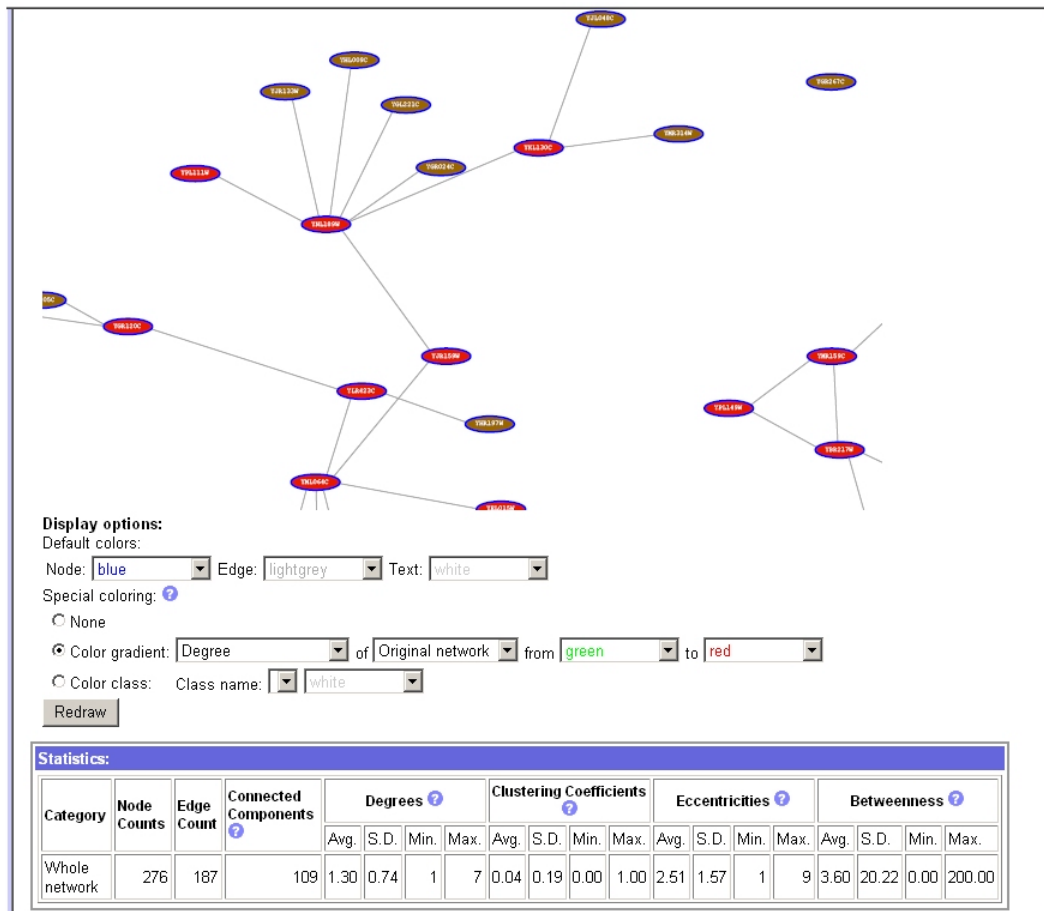


Figure 9.1. The intersection of two yeast-two-hybrid datasets [97, 185] with all nodes having no edges in the intersection filtered by a statistics filter. The nodes are colored according to their degrees. Also shown in the figure are the various statistics of the resulting network.



**Table 9.1. A comparison of several network analysis and visualization systems. Note that we have included only some network analysis and visualization systems in this comparison.**

	Cytoscape (2.3.1)			JUNG (1.0)	N-Browse (10 Aug 2006)	Osprey (1.2)	tYNA (10 Aug 2006)
	Basic	NetworkAnalyzer plugin (1.0)	Metabolica plugin (1.0)				
Main purpose	Visualization	Network analysis	Motif finding	Graph library	Visualization	Visualization	Network analysis
System	Standalone	Plugin	Plugin	Standalone	Web	Standalone	Web
· Link from external resources	Indirect: Java Web Start	N/A	N/A	No	Direct	No	Direct
· Web service interface	No	No	No	No	No	No	Yes: SOAP
Statistics calculation	No	Yes	No	Yes	No	No	Yes
· Degree	No	Yes	No	Yes	No	No	Yes
· Clustering coefficient	No	Yes	No	Yes	No	No	Yes
· Shortest path length (eccentricity)	No	Yes	No	Yes	No	No	Yes
· Betweenness	No	No	No	No	No	No	Yes
Motif finding	No	No	Yes	No	No	No	Yes
· Chain	No	No	No	No	No	No	Yes
· Cycle	No	No	Yes	No	No	No	Yes
· Feed-forward loop	No	No	Yes	No	No	No	Yes
· Complete two-layer	No	No	No	No	No	No	Yes
· Defective clique	No	No	No	No	No	No	Yes
Multiple network operations	Yes	No	No	No	Yes	Yes	Yes
User network management	Session	N/A	N/A	Individual files	Database	Individual files	Database
· Network classification	By session	N/A	N/A	No	No	No	By tagging attributes

to make the list exhaustive. Furthermore, since each system has its unique goals, it is not completely fair to compare them in this way. This table simply serves as a quick reference for readers who are interested in knowing some of the differences between the systems.

## 9.2 Using tYNA

tYNA provides a simple view with some basic features, and an advanced view for more complex analyses.

### 9.2.1 Uploading networks and categories

The first step of analysis is to upload networks. tYNA accepts various file formats, including the SIF format of Cytoscape. One may also enter additional attributes to organize the networks into groups, such as network type (e.g. protein-protein interaction), organism (e.g. yeast) and experimental method (e.g. yeast-two-hybrid) (Figure 9.2). Furthermore,

tYNA allows users to analyze subsets of the networks (e.g., active parts in a dynamic network [86, 123]) by using category files.

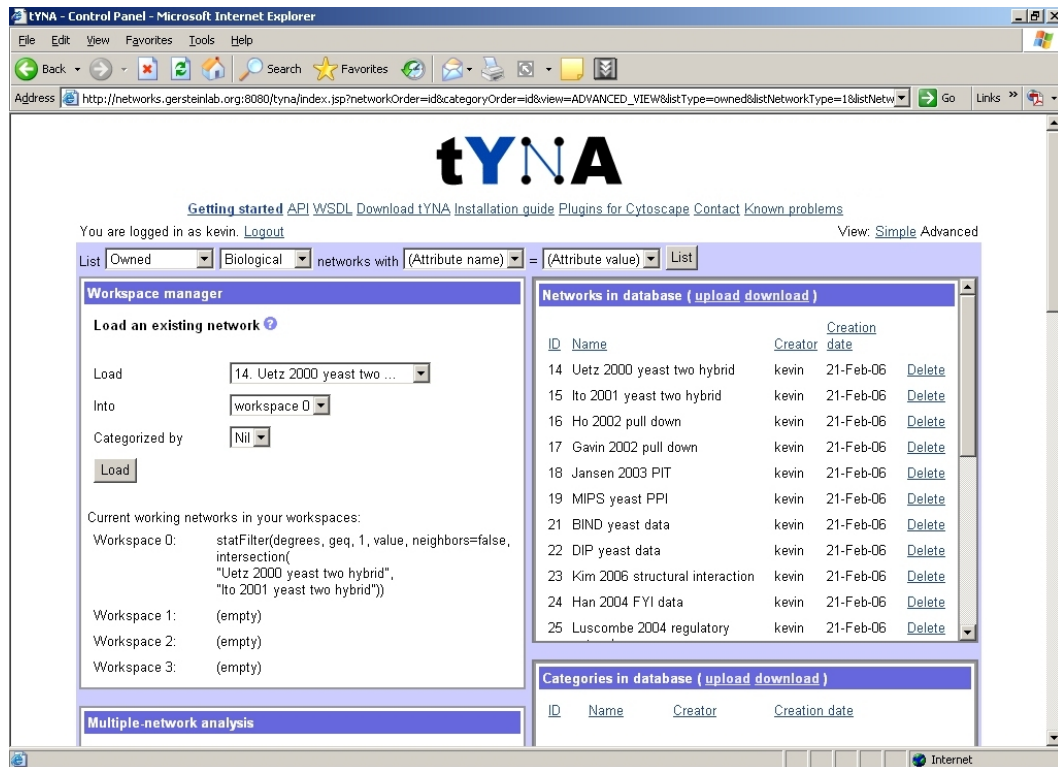


Figure 9.2. Networks uploaded and categorized in the tYNA database.

### 9.2.2 Loading networks into workspaces

After uploading a network, one may view its statistics and visualize it graphically by loading it into a workspace. A workspace is a working area for a single network (Figure 9.1). Various statistics are computed, such as the clustering coefficient, eccentricity and betweenness [210]. Networks are visualized in Scalable Vector Graphics (SVG) using the

aiSee package<sup>4</sup>, which facilitates an interactive interface: one may change the appearance of the network in real time (Figure 9.3 and Figure 9.4).

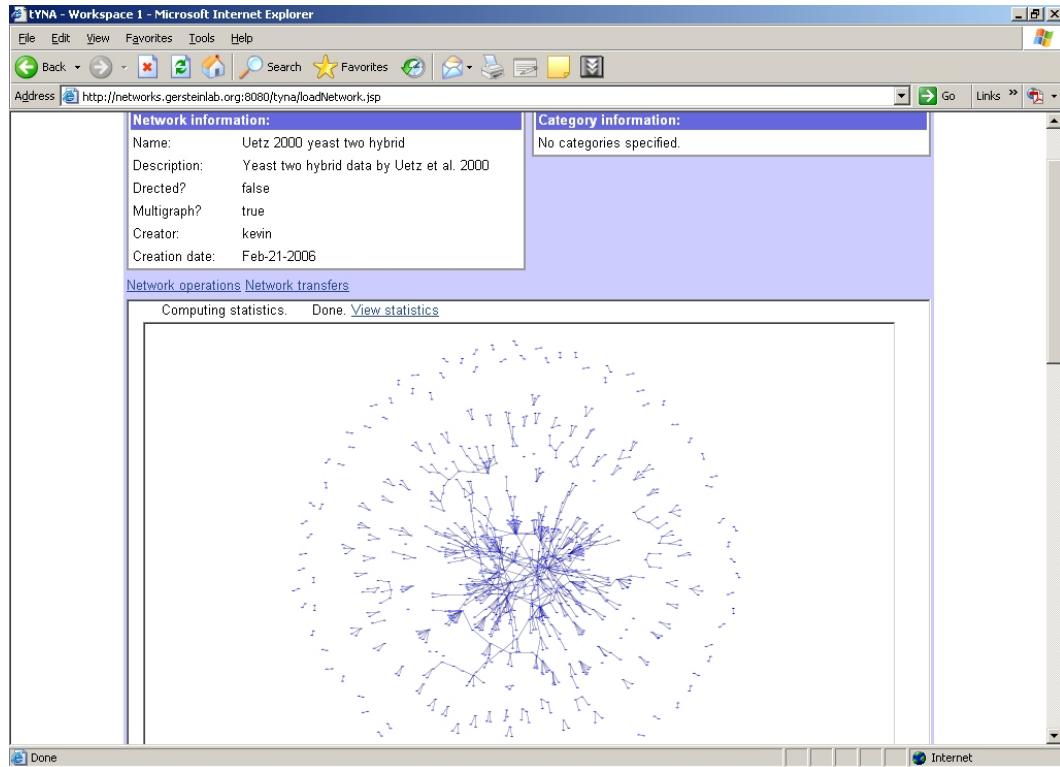


Figure 9.3. Visualizing a network in a workspace.

### 9.2.3 Single-network operations (advanced view)

Filtering allows one to retain a portion of the network, based on a statistics cutoff (e.g. the 5% of nodes with the highest out-degrees) or node names. It will easily allow one to identify the hubs and bottlenecks in a graph. Motif finding identifies various regular patterns in the network, including chains, cycles, feed-forward loops and complete two-

<sup>4</sup><http://www.aisee.com>

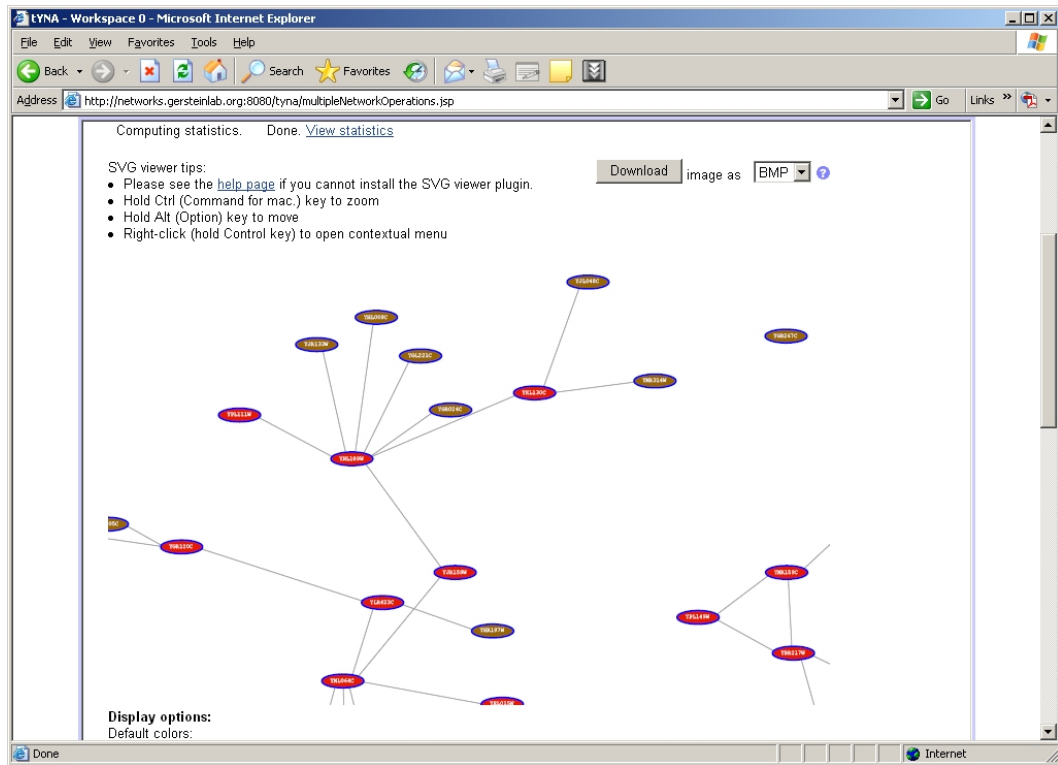


Figure 9.4. Controls for image export and SVG viewer tips.

layers. They generalize the motifs discussed in previous studies [115, 132, 169]. Currently all occurrences of a specified motif pattern are returned. We will study the feasibility of returning only statistically over-represented motifs in future work. tYNA also identifies defective cliques [209] that suggest potential missing edges in a network (Figure 9.5 and Figure 9.6).

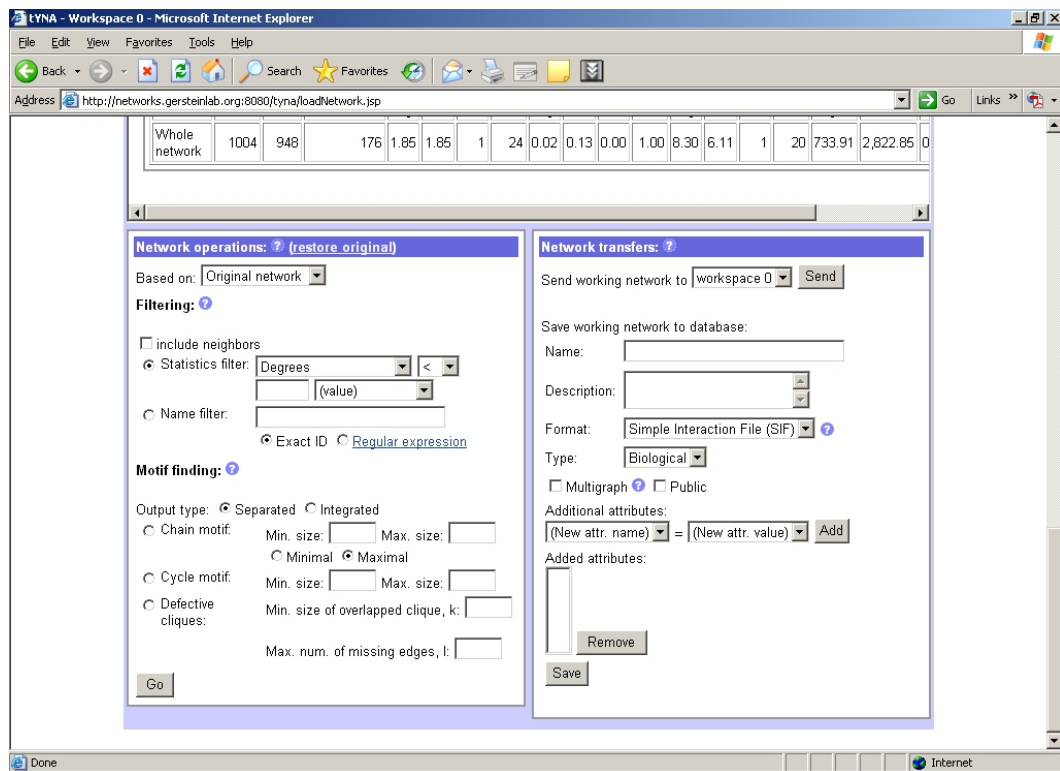


Figure 9.5. Single network operations (left) and network transfer options (right).

#### 9.2.4 Multiple-network operations (advanced view)

Multiple-network operations allow one to select multiple networks, perform some operations on each of them, and merge them into a single network. For example, the intersection

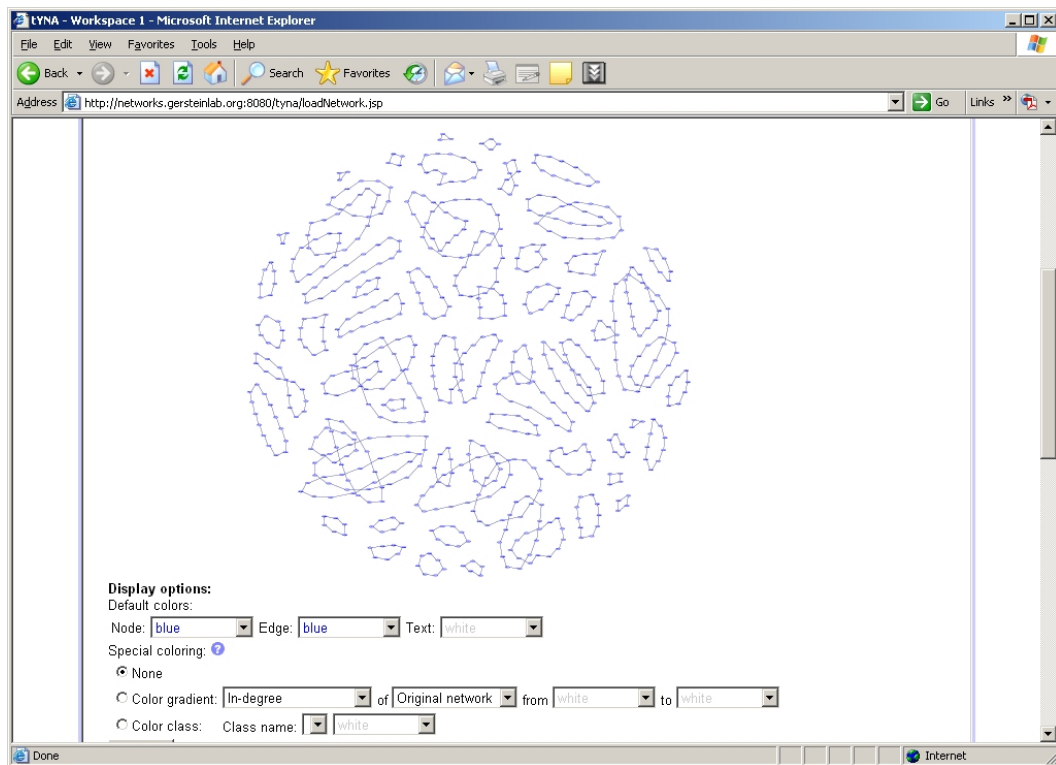


Figure 9.6. All cycles that involve three or more proteins identified from a regulatory network.

of multiple high-throughput protein-protein interaction networks offers a high-confidence set of potential interactions. The relationships between different kinds of networks, such as gene regulation and co-expression, can also be studied (Figure 9.7).

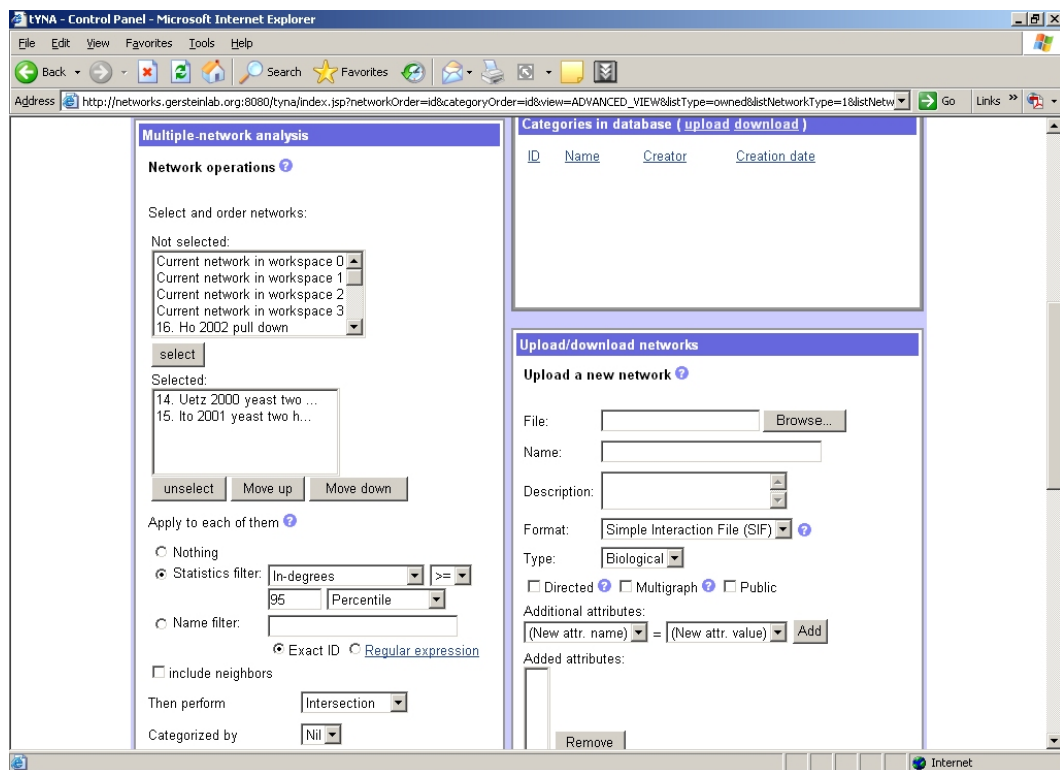


Figure 9.7. Multiple network operations.

### 9.2.5 Mining and edge overlap (advanced view)

The edge overlap feature allows the comparison of the edges in two networks. It can test, using some prediction functions, how well one network predicts another. Some prediction functions are predefined, such as identity, sibling and couple (Table 9.2).

**Table 9.2. Definitions for edge prediction functions.**

Function	Definition
Couple	If the first network contains the edges $A \rightarrow C$ and $B \rightarrow C$ , predict that $A-B$ is an edge in the second network.
Sibling	If the first network contains the edges $A \rightarrow B$ and $A \rightarrow C$ , predict that $B-C$ is an edge in the second network.
Identity	If the first network contains the edges $A-B$ , predict that $A-B$ is an edge in the second network.

### 9.2.6 Saving and downloading analyzed networks

Finally, one may save a working network into the database, or send it to another workspace (as a temporary backup). Likewise, one may download a working network in various network and graphics formats, including SIF, bitmap, postscript and PDF (Figure 9.4 and Figure 9.5).

## 9.3 Implementation

All source codes were written in Java using standard J2EE architecture. A detailed JavaDoc API is available for users who want to use the classes in their own codes. The tYNA database can also be accessed through standard SOAP-based web services, and we have developed a plugin (available on the tYNA website) to interface tYNA with the network visualization system Cytoscape (Figure 9.8).

## 9.4 Discussion

As biological network analysis is a vigorous research area, new statistics, motifs, and mining algorithms are expected to emerge continuously. tYNA was thus designed in a



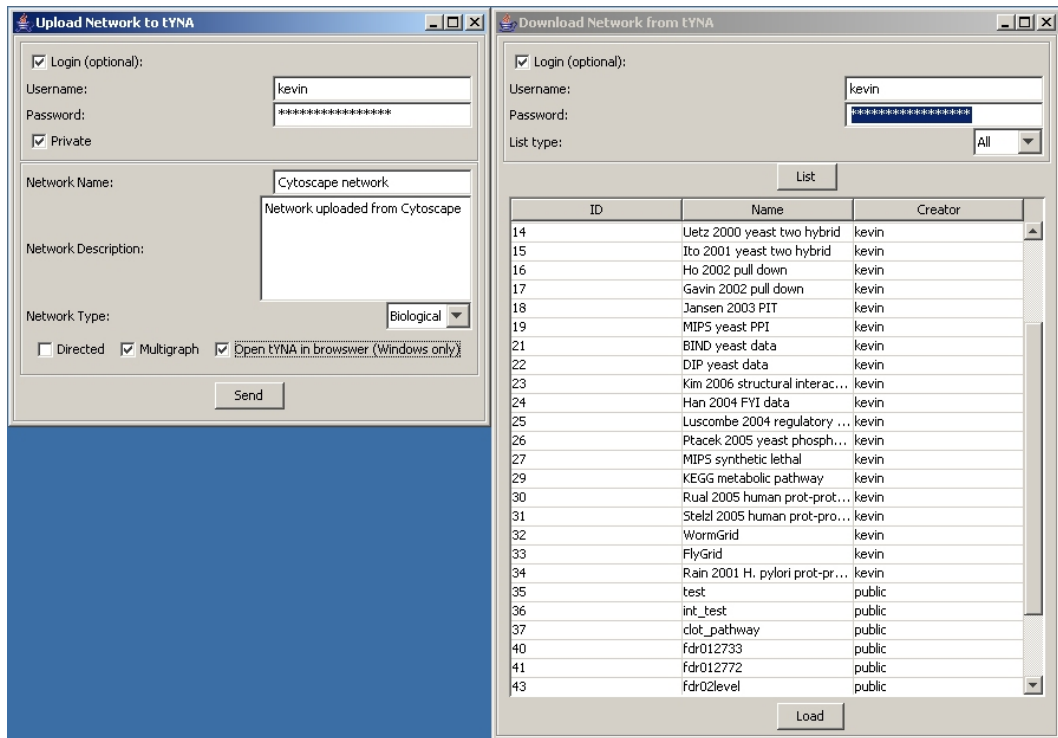


Figure 9.8. Plugin for Cytoscape.

modular fashion so that new features can be readily added. Being a Web system, the new features are immediately made accessible to users.

We plan on connecting tYNA with dataset management systems such as YeastHub [39], and Bind [4], DIP [198], MINT [211], Reactome [99] and annotation databases to provide a unified platform for performing complex analyses. We think that the combination of the analysis features provided by tYNA and the advanced visualization facilities of Cytoscape can prove particularly powerful. We also plan on interfacing tYNA with analysis and visualization tools such as bioPIXIE [134] and Pajek [15], which would allow researchers to combine the distinct features of each tool.

## Chapter 10

# The Coevolution Server

### 10.1 Introduction

Coevolution (covariation/correlated mutation) is the change of a biological object triggered by the change of a related object. For example, the coding genes of some interacting proteins are preserved or eliminated together in new species [146], or have similar phylogenetic trees [76]. At the amino acid level, some residues under physical or functional constraints exhibit correlated mutations [74, 173, 180]. In the context of this thesis, coevolution can be viewed as a kind of relationship between biological objects, which define networks at the gene or residue levels.

Coevolving residues in a protein are detected in a two-step process: 1) the multiple sequence alignment (MSA) of the protein and its homologs is constructed or obtained; 2) a coevolution score is calculated for each pair of sites in the MSA. There are two main difficulties in this process. First, a large number of scoring functions have been proposed in

the literature (see Halperin [85] for a recent survey). It can be difficult to choose from them, as they exhibit subtle yet significant differences, and it is likely that different applications would require different functions. Second, coevolution analyses could be confounded by uneven sequence representations, insufficient evolutionary divergence, and the presence of gaps in the MSA. A successful coevolution study has to take all these details into account.

To address this need, we have developed an integrated system (<http://coevolution.gersteinlab.org>) that provides a simple interface for preprocessing data, computing coevolution scores, and analyzing the results. It offers a great variety of scoring variations (over 100) for studying different types of proteins and testing different hypotheses. The workflow of the system is shown in Figure 10.1. More details on the scoring functions, preprocessing options, and result analysis are provided below.

## 10.2 Scoring Functions

### 10.2.1 Correlation-based functions

For a pair of sites  $i$  and  $j$  in an MSA, the correlation score [75, 85] is computed as follows:

$$Cor(i, j) = \frac{2}{N(N-1)} \frac{\sum_{k < l} w_{kl} (s_{ikl} - \bar{s}_i)(s_{jkl} - \bar{s}_j)}{\sigma_i \sigma_j} \quad (10.1)$$

where  $s_{ikl}$  is the score for substituting the  $i$ -th residue of sequence  $k$  by that of sequence  $l$ ,  $\bar{s}_i$  and  $\sigma_i$  are the mean and standard deviation of substitution scores at site  $i$ ,  $N$  is the

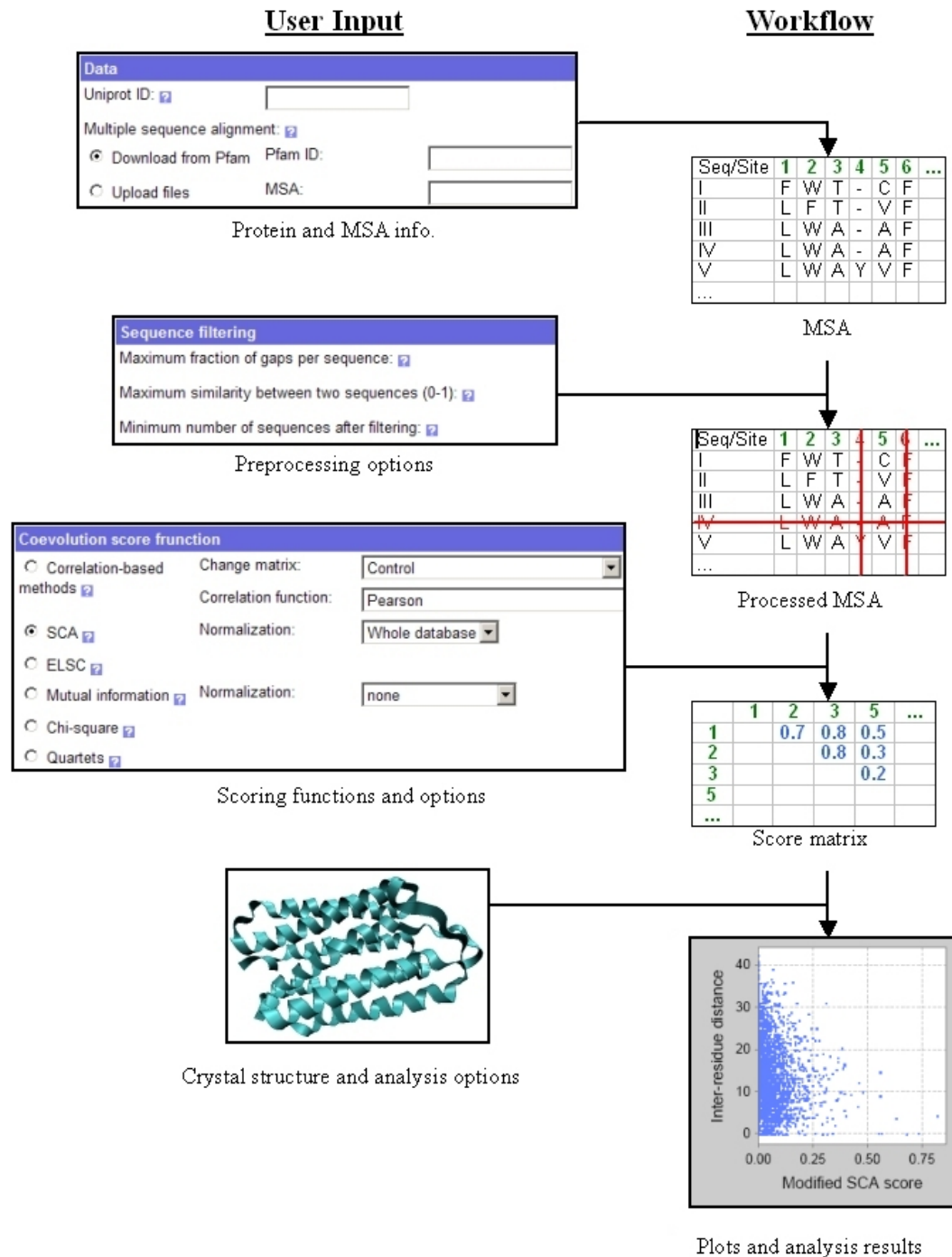


Figure 10.1. The workflow of the system.

number of sequences in the MSA, and  $w_{kl}$  is the weight for the sequence pair  $k, l$ . If the two sites are coevolving in that radical substitutions at the first site are accompanied by radical substitutions at the second site, the correlation will be high. Our system provides the classical McLachlan matrix [128] that scores substitutions based on the physiochemical properties of the residues, as well as matrices based on residue volume, pI, and hydrophathy index, for studying the properties individually. Two variations are provided for each of them: the “absolute value version” considers only the magnitude, while the “raw version” also considers the direction of change, for detecting compensatory mutations. The correlation can be computed from raw values (Pearson correlation) or from value ranks (Spearman correlation [143]). Several schemes are provided for the weights  $w_{kl}$ , preventing false coevolution signals due to uneven sequence representation or site conservation.

### 10.2.2 Perturbation-based functions

The idea of perturbation-based functions is to perform a “perturbation” at a first site, and observe its effect on a second site. The Statistical Coupling Analysis (SCA) method [121] defines a statistical energy term for a site, and computes the energy change at a second site when the first site is perturbed by retaining only the sequences with a certain residue.<sup>1</sup> The Explicit Likelihood of Subset Variation (ELSC) method [47] is based on the same idea, but has the energy computations replaced by probabilities according to hypergeometric distributions. The mutual information (MI) method [74] can be viewed as a generalized perturbation method that considers the subsetting of all twenty kinds of residues, and combines them by a weighted average according to their frequencies. To deal with finite sample size effects and phylogenetic influence, the normalization options in [126]

---

<sup>1</sup>Our implementation provides an asymmetric SCA score matrix, as well as extra summarizing statistics. Details can be found at the appendix of this chapter.

are also provided.

### 10.2.3 Independence tests

The chi-square test (c.f. the OMES method [112]) and the quartets method [65] both identify site pairs that are unlikely to be independent. The former computes the p-value under the null hypothesis of independent sites. The latter counts the number of quartets in the two-dimensional histogram of residue frequencies that deviate considerably from the expectation.

## 10.3 Preprocessing options

To improve the sensitivity and specificity of the functions, options are provided for preprocessing sequences, sites and site pairs.

### 10.3.1 Sequence filtering and weighting

Sequences that contain too many gapped positions or are too similar to others in the MSA (which might cause sites to appear coevolving) can be removed by specifying the gap and similarity thresholds respectively. A minimum number of sequences can also be specified to avoid small sample size effects.

A sequence weighting scheme based on the topology of the phylogenetic tree [71] and one based on Markov random walk are provided. Both schemes down-weight sequences that are very similar to others in the MSA.

### 10.3.2 Site filtering

After sequence filtering, sites that contain too many gaps or are too conserved can be discarded. The former is likely non-informative, while the latter may artificially inflate some coevolution scores.

### 10.3.3 Site pair filtering

Sites that are close in the primary sequence may produce trivial coevolution signals that hide other more unexpected coevolution events. Such site pairs can be filtered by specifying the minimum sequence separation. It has also been observed that insertions/deletions of multiple residues may create artificial coevolution signals [142]. An option is provided for filtering site pairs that participate in the same gaps in too many sequences.

### 10.3.4 Other options

Grouping similar residues into a smaller alphabet may increase the sensitivity [147]. Our system provides two residue groupings proposed in the literature [56, 81]. It has also been observed that gaps might give important coevolution signals [142]. An option is provided for treating gaps as noise or as the twenty first residue when computing coevolution scores.



## 10.4 Scores analysis

In some proteins coevolving residues tend to be close to each other in the 3D structure [47, 74]. This suggests that the instability created by the mutation of a residue may be (partially) compensated for by a corresponding mutation of a close residue. Coevolution signals may thus convey some information about the protein structure. For instance it is interesting to study how well the coevolution scores predict the residue contact map [85]. Our system provides functions for plotting and analyzing the coevolution scores against inter-residue distances, and standard machine-learning techniques (e.g. ROC curve) for evaluating the effectiveness of the various coevolution functions in predicting interacting residues. A shuffling scheme for evaluating the significance of the scores is also provided in the program package for running locally.

## 10.5 Example

We provide a worked example of our system in operation on the web site, which illustrates coevolution in the transmembrane protein bacteriorhodopsin due to physically constrained residues not adjacent in the primary sequence. The example can be easily loaded by clicking the corresponding link on the main page. Running the example will compute the coevolution scores between site pairs separated by at least 3 residues. The scatterplot for coevolution scores against inter-residue distances generated using a known PDB structure (Figure 10.1) shows that residue pairs receiving high scores do tend to be closer in the crystal structure.

Due to the intensive computation involved in the score calculations, currently only one

scoring function is allowed to be used each time. Anyone interested in performing large-scale comparisons can download the Java programs from the web site and run locally on most platforms (Windows, Macintosh, Linux, UNIX, etc.). Detailed installation instructions are provided on the web site.

## 10.6 Discussion

Although the scatterplot in Figure 10.1, and other studies in the literature, have suggested some relationships between coevolution and physical constraints, to what extent coevolution scores could help us understand physical structures remains unclear. We hope the current application can serve as a neutral tool for further exploration in this area.

The current system focuses on functions that do not assume any mutation models. Other functions, such as the likelihood method by Pollock et al. [147] and the Bayesian mutational mapping method [49] may be added in a later version.

Coevolution signals have been used in recent studies to predict sequence regions involved in protein-protein interactions with different levels of success [144, 85]. We plan on extending the system to include inter-protein residue coevolution in the next phase of development.

## 10.7 Appendix: our implementation of the SCA method

### 10.7.1 Introduction

The Statistical Coupling Analysis (SCA) method is one of the earliest and most popular methods for measuring the coevolution of pairs of sites. It was first described in Lockless and Ranganathan [121]. We based our implementation of the SCA method on the description in this article, as well as the description in Suel et al. [180], and its web supplement at <http://www.hhmi.swmed.edu/Labs/rr/SCA.html>. In the following we will call them “the reference sources”. We have also referenced the Matlab software of the original authors (SCA version 1.5) for some implementation details.

Since not all the algorithmic details are given in the reference sources, and we need to fit our implementation into the overall software framework, we have made a number of design choices. We have made our best effort in having the choices reasonable and close to the original definitions in the reference sources. Yet we have to stress that our implementation is not completely the same as the one of the original authors, and users of our system should be aware of the details of our implementation, which we describe below.

We have also referenced Dekker et al [47] since the authors also implemented the SCA method and made some design choices. However, our choices are not exactly the same as theirs.

We have discussed our design with some of the SCA inventors (Rama Ranganathan and William Russ). We follow their suggestion to produce a non-square, non-symmetric SCA matrix as was done in the original SCA papers (based on the details described below, which are close to, but not completely the same as their SCA software), with each row

being a site and each column a perturbation. Then, to produce one single coevolution score for each pair of sites, we use a statistic to summarize the validated scores of the different perturbations. To distinguish this final statistic from the original perturbation scores, we call the former “Modified SCA” to emphasize its difference from the latter.

### 10.7.2 Design choices

- The constant  $kT^*$ : the exact value of the temperature parameter  $T^*$  is not described in the three reference sources. Since  $kT^*$  is a constant in all calculations, and it is described as “an arbitrary energy unit” in both Lockless and Ranganathan [121] and Suel et al. [180], we fix its value to one. We remind users who want to compare the SCA scores of different MSAs to perform proper normalizations according to the number of sequences in the MSAs. (Note: in the following we assume some data preprocessing may have been performed. So for example when we talk about an MSA, we actually refer to the preprocessed MSA that may have some sequences and/or sites filtered.)
- Normalization by  $P_{MSA}^x$ : it is mentioned in the reference sources, and implemented in the original code used to produce the results of the reference sources. However, as mentioned in Dekker et al. [47], and confirmed by the SCA inventors (personal communication), it is not implemented in the software package. Since this normalization is not important according to the SCA inventors (personal communication), it is not included in our implementation.
- Acceptance criteria: the three reference sources emphasize that the multiple sequence alignments (MSAs) used in the analysis should be large enough and diverse enough to be a statistical representative of the evolutionary constraints on the protein family.

These concerns are partially handled by the various data preprocessing methods of our system. But in order to have our implementation of the SCA method as close to the original definitions as possible, we also implement the acceptance criteria described in the three reference sources.

The most detailed description of the acceptance criteria is in the web supplement of Suel et al. [180]. It lists three acceptance criteria:

- “The MSA should be so diverse that several sites display amino-acid distributions near to the mean in all natural proteins.”
- “The MSA should be so large that random elimination of sequences from the alignment does not change the amino-acid frequencies at sites much.”
- “Perturbations at sites in the MSA should produce sub-alignments that are also large and diverse, such that they remain a representative subset of the parent MSA and do not substantially alter the state of statistical equilibrium.”

We were unable to precisely locate the corresponding implementation details in the Matlab code in a form that we could re-implement. Therefore we designed our acceptance procedure according to the above criteria as follows:

1. We first calculate the statistical energy vectors  $\overrightarrow{\Delta G_j^{stat}}$ <sup>2</sup> for all sites  $j$  at which at least 85% of the sequences are not a gap. The five sites with the smallest magnitudes (2-norms) of  $\overrightarrow{\Delta G_j^{stat}}$  (i.e., the five most unconserved sites) are chosen to have the magnitudes averaged. If less than five sites pass the 85% requirement,

---

<sup>2</sup>As in SCA 1.5, both  $\overrightarrow{\Delta G_j^{stat}}$  and  $\overrightarrow{\Delta \Delta G_{ij}^{stat}}$  are divided by 100 after the calculations described in the reference sources.

the MSA is rejected as having too many gaps. If the average magnitude is more than a threshold  $\alpha$  (default to 1.0), the MSA is rejected as being too conserved.

2. Using the five sites in step 1, we perform random sequence elimination. It is divided into iterations. At the  $i$ -th iteration, the number of sequences to be eliminated is  $0.03 \times i \times n$ , where  $n$  is the total number of sequences in the MSA. 100 random eliminations are performed per iteration, and the average of the averaged  $\overrightarrow{\Delta G_j^{stat}}$  of the five sites are recorded. If the averaged  $\overrightarrow{\Delta G_j^{stat}}$  reaches  $\alpha$  before  $\beta\%$  (default to 50) of the sequences in the MSA are eliminated, the MSA is rejected as being too small.
3. Using the five sites in step 1, we perform another random sequence elimination to determine the size threshold for a perturbation. The same number of sequences is eliminated in each iteration, but instead of recording  $\overrightarrow{\Delta G_j^{stat}}$ , we treat each random elimination as a perturbation, and record the average coupling energies  $\overrightarrow{\Delta \Delta G_{ij}^{stat}}$  over the 100 random eliminations of the iteration. When the average  $\overrightarrow{\Delta \Delta G_{ij}^{stat}}$  reaches  $\gamma$  (default to 0.07), the number of sequences not eliminated in the iteration is set as the size threshold. When calculating the perturbation scores, any perturbation that results in less remaining sequences than the threshold is rejected.

- Perturbation and symmetry: in the original SCA implementation, a perturbation is performed by retaining one of the 20 amino acids at a site. Therefore for a pair of sites, there are in total  $2 \times 20 = 40$  possible perturbations. All perturbations passing the acceptance criteria are recorded. The final SCA matrix is a rectangular matrix with each site as a row and each perturbation (a site number-residue pair) as a column.

Our system outputs this original SCA matrix whenever the SCA method is chosen.

In order to fit in the overall framework with one single coevolution score for each site pair, we also use a statistic to summarize the (at most 40) non-rejected perturbation scores into one final coevolution score, which we call the “modified SCA score”.

Let  $SCA(i, j, a)$  be the original SCA score between sites  $i$  and  $j$  by retaining only residue  $a$  at site  $j$ , our modified SCA score is calculated as

$$\frac{1}{2n} \left[ \sum_{a:n_{j,a} \geq t} n_{j,a} SCA(i, j, a) + \sum_{a:n_{i,a} \geq t} n_{i,a} SCA(j, i, a) \right]$$

,

where  $n$  is the number of sequences in the MSA,  $n_{i,a}$  is the number of sequences having residue  $a$  at site  $i$ , and  $t$  is the size threshold determined by the acceptance criterion. Basically, the score is a weighted sum of the SCA scores of the various perturbations, but taking into account only the non-rejected ones.

## Chapter 11

# Conclusion

In this thesis, we have demonstrated that the accuracy of biological network reconstruction can be greatly improved by exploiting some data properties and problem structures.

In the study of training set expansion, we observed that the sparse and unevenly distributed training data limited the effectiveness of the local modeling approach, especially when the feature space was of a high dimension. By propagating highly confident predictions to other local models, and generating auxiliary training examples from the most similar and most dissimilar object pairs in the kernel matrix, we observed consistent performance improvement over local modeling across different datasets with different features and in different evaluation settings.

Motivated by the natural concept hierarchy of protein, domain and residue interactions, we extended the idea of training set expansion to vertically expand training sets at different levels in the study of multi-level learning. We considered different architectures, methods for coupling the different levels, and ways to carry out learning at each level. Our empirical



study demonstrated the advantage of bidirectional flow of predictions between levels using training set expansion as opposed to independent levels and unidirectional flow of training data.

Training set expansion is only one possible way to couple the different levels. We also studied how the prediction problems at the protein and domain levels could be coupled by using combined optimization. Noticing that the prediction accuracy of previous learning methods are quite sensitive to errors in the input protein interaction network, we developed a new algorithm that uses protein-level features as a cross check of the training data. The prediction accuracy was observed to be improved, more prominently when predicting protein interactions.

The combined optimization approach demonstrates the advantage of combining data from both the protein and domain levels. We also studied how heterogeneous datasets could be combined to improve the prediction of gene regulatory networks. The two types of data we considered, namely expression profiles of deletion strains and time series data after an initial perturbation, are complementary in making predictions. Deletion profiles provide information for predicting direct and simple regulation, while perturbation data allow the detection of more complex regulation. The key of combining the two types of data is to rank the predictions according to their confidence values. This approach was shown fruitful in the DREAM challenge, in which our prediction method outperformed the predictors from other teams.

The idea of exploiting special data properties and problem structures is not only useful to network reconstruction, but also other kinds of machine learning problems in bioinformatics, for a number of reasons:

- There is a lot of domain knowledge that can be used to guide the learning process, yet standard learning algorithms are not designed to utilize such knowledge.
- Rapid technological advancements have triggered the generation of more and more data, both in terms of volume and variety. Some algorithms that were impractical due to lack of data are now becoming feasible.
- Many computational problems in biology have been extensively studied and it has become very difficult to achieve further improvements merely by algorithmic enhancements such as better optimization procedures. New data and un-utilized domain knowledge are rich sources of novel ideas.

Following this idea, we have identified a number of directions for future research on biological network reconstruction and analysis:

**Spatial and temporal aspects of interactions:** In this thesis we have focused on the problem of finding out the edges of the interaction networks, without considering where and when they exist. The exact location and time that the interactions occur actually have a lot of implications to the underlying biology. For instance, the concept of activity motifs has recently been proposed [35] as a means to understand the order of activity in different parts of a biological pathway. The additional spatial and temporal information allow for the study of dynamics within cells, which cannot be obtained from static networks.

**New sequencing data:** The recent technological breakthrough in high-throughput sequencing has drastically reduced the time and cost of decoding long sequences. Some microarray-based experiments are starting to be replaced or at least complemented by their sequencing counterparts. For example, chromatin immunoprecipitation with microarray (ChIP-chip) is being compared to its sequencing alternative with sequencing (ChIP-

seq) [125], while tiling array is compared to RNA sequencing (RNA-seq) [192]. In general the data obtained from sequencing data are less noisy as compared to microarray measurements. An important conceptual difference between these two types of technology is that microarray involves a pre-defined set of target sequences to be measured, while sequencing does not require any prior knowledge about the targets. A lot of previously unknown sequences are expected to be obtained. From a network perspective this means that new nodes with missing features are added to the networks. Algorithms should therefore be more ready to handle missing data, and to take in raw sequences as features. For instance, kernels have been proposed for sequence data in recent years [109, 116, 118, 117, 153, 159]. Newer ones are expected to be developed as more sequences become available and the need for more specialized sequence kernels arises.

**Network of networks:** Currently different kinds of networks are reconstructed separately, yet due to their close relationships combining the reconstruction problems could potentially offer new insights. For example, in our work on reconstructing gene regulatory networks, we implicitly assume that regulators are independent of each other. Yet in reality regulators can be complexes formed by multiple proteins. Therefore if two proteins are both observed to have some probability of regulating a gene, knowing that they physically interact or form a complex according to the protein interaction network would increase our confidence that they are both regulators. Conversely, if two proteins are observed to regulate very similar sets of genes, they might actually form a complex and work together in regulating the genes.

# Bibliography

- [1] Rafal Adamczak, Aleksey Porollo, and Jaroslaw Meller. Combining prediction of secondary structure and solvent accessibility in proteins. *Proteins: Structure, Function, and Bioinformatics*, 59:467–475, 2005.
- [2] Mark Aizerman, Emmanuil Braverman, , and Lev Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821V–837, 1964.
- [3] Mario Albrecht, Carola Huthmacher, Silvio C. E. Tosatto, and Thomas Lengauer. Decomposing protein networks into domain-domain interactions. *Bioinformatics*, 21(Suppl. 2):ii220–ii221, 2005.
- [4] C. Alfarano, C. E. Andrade, K. Anthony, N. Bahroos, M. Bajec, K. Bantoft, D. Betel, B. Bobechko, K. Boutilier, E. Burgess, K. Buzadzija, R. Cavero, C. D’Abreo, I. Donaldson, D. Dorairajoo, M. J. Dumontier, M. R. Dumontier, V. Earles, R. Farrall, H. Feldman, E. Garderman, Y. Gong, R. Gonzaga, V. Grytsan, E. Gryz, V. Gu, E. Haldorsen, A. Halupa, R. Haw, A. Hrvojjic, L. Hurrell, R. Isserlin, F. Jack, F. Juma, A. Khan, T. Kon, S. Konopinsky, V. Le, E. Lee, S. Ling, M. Magidin, J. Moniakis, J. Montojo, S. Moore, B. Muskat, I. Ng, J. P. Paraiso, B. Parker, G. Pintilie, R. Pirone,

- J. J. Salama, S. Sgro, T. Shan, Y. Shu, J. Siew, D. Skinner, K. Snyder, R. Stasiuk, D. Strumpf, B. Tuekam, S. Tao, Z. Wang, M. White, R. Willis, C. Wolting, S. Wong, A. Wrong, C. Xin, R. Yao, B. Yates, S. Zhang, K. Zheng, T. Pawson, B. F. F. Ouellette, and C. W. V. Hogue. The biomolecular interaction network database and related tools 2005 update. *Nuclear Acids Research*, 33(Database Issue):D418–D424, 2005.
- [5] Jon C. Allen. A modified sine wave method for calculating degree days. *Environmental Entomology*, 5(3):388–396, 1976.
- [6] Patrick Aloy and Robert B Russell. Structure-based systems biology: A zoom lens for the cell. *FEBS Letters*, 579:1854–1858, 2005.
- [7] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
- [8] Anupriya Ankolekar, Markus Kroetzsch, Thanh Tran, and Denny Vrandečić. The two cultures: Mashing up web 2.0 and the semantic web. In *The 16th International World Wide Web Conference*, pages 825–834, 2007.
- [9] Rolf Apweiler, Amos Bairoch, Cathy H. Wu, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J. Martin, Darren A. Natale, Claire ODonovan, Nicole Redaschi, and Lai-Su L. Yeh. UniProt: the universal protein knowledgebase. *Nuclear Acids Research*, 32:D115–D119, 2004.
- [10] Ramon Aragues, Daniel Jaeggi, and Baldo Oliva. PIANA: Protein interactions and network analysis. *Bioinformatics*, 22(8):1015–1017, 2006.

- [11] Ramon Aragues, Andrej Sali, Jaume Bonet, Marc A. Marti-Renom, and Baldo Oliva. Characterization of protein hubs by inferring interacting motifs from protein interactions. *PLoS Computational Biology*, 3(e178), 2007.
- [12] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.
- [13] Joel S Bader, Amitabha Chaudhuri, Jonathan M Rothberg, and John Chant. Gaining confidence in high-throughput protein interaction networks. *Nature Biotechnology*, 22(1):78–85, 2003.
- [14] C. Baral, K. Chancellor, N. Tran, N. L. Tran, A. Joy, and M. Berens. A knowledge based approach for representing and reasoning about signaling networks. *Bioinformatics*, 20(Suppl. 1):i15–i22, 2004.
- [15] V. Batagelj and A. Mrvar. Pajek - analysis and visualization of large networks. In M. Junger and P. Mutzel, editors, *Graph Drawing Software*, pages 77–103. Springer, 2003.
- [16] Pedro Beltrao, Christina Kiel, and Luis Serrano. Structures in systems biology. *Current Opinion in Structural Biology*, 17:378–384, 2007.
- [17] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21(Suppl. 1):i38–i46, 2005.

- [18] Allister Bernard, David S. Vaughn, and Alexander J. Hartemink. Reconstructing the topology of protein complexes. In *Eleventh Annual International Conference on Research in Computational Molecular Biology RECOMB*, 2007.
- [19] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. World-wide web. *Communications of the ACM*, 37:76–82, 1994.
- [20] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific America*, 284(5):34–43, 2001.
- [21] Kevin Bleakley, Gérard Biau, and Jean-Philippe Vert. Supervised reconstruction of biological networks with local models. *Bioinformatics*, 23(ISMB/ECCB 2007):i57–i65, 2007.
- [22] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *The Eleventh Annual Workshop on Computational Learning Theory*, pages 92–100, 1998.
- [23] Joel R. Bock and David A. Gough. Predicting protein-protein interactions from primary structure. *Bioinformatics*, 17(5):455–460, 2001.
- [24] Béla Bollobás. *Modern Graph Theory*. Springer, 1998.
- [25] Richard Bonneau, David J Reiss, Paul Shannon, Marc Facciotti, Leroy Hood, Nitin S Baliga, and Vesteynn Thorsson. The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biology*, 7(R36), 2006.

- [26] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *The Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.
- [27] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [28] Alvis Brazma, Helen Parkinson, Ugis Sarkans, Mohammadreza Shojatalab, Jaak Vilo, Niran Abeygunawardena, Ele Holloway, Misha Kapushesky, Patrick Kemmeren, Gonzalo Garcia Lara, Ahmet Oezcimen, Philippe Rocca-Serra, and Susanna-Assunta Sansone. ArrayExpress x a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research*, 31(1):68–71, 2003.
- [29] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H. Lackner, Jurg Bahler, Valerie Wood, Kara Dolinski, and Mike Tyers. The BioGRID interaction database: 2008 update. *Nucleic Acids Research*, 36:D637–D640, 2008.
- [30] Declan Butler. Mashups mix data in global service. *Nature*, 439(7072):6–7, 2006.
- [31] Daniel R. Caffrey, Shyamal Somaroo, Jason D. Hughes, Julian Mintseris, and Enoch S. Huang. Are protein-protein interfaces more conserved in sequence than the rest of the protein surface? *Protein Science*, 13(1):190–192, 2004.
- [32] CR Cantor. Orchestrating the human genome project. *Science*, 248(4951):49–51, 1990.
- [33] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machine, 2008. <http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf>.



- [34] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.
- [35] Gal Chechik, Eugene Oh, Oliver Rando, Jonathan Weissman, Aviv Regev, and Daphne Koller. Activity motifs reveal principles of timing in transcription control of the yeast metabolic network. *Nature Biotechnology*, 26(11):1251–1259, 2008.
- [36] Hong-Chu Chen, Hsiao-Ching Lee, Tsai-Yun Lin, Wen-Hsiung Li, and Bor-Sen Chen. Quantitative characterization of the transcriptional regulatory network in the yeast cell cycle. *Bioinformatics*, 20(12):1914–1927, 2004.
- [37] Xue-Wen Chen and Mei Liu. Prediction of protein-protein interactions using random decision forest framework. *Bioinformatics*, 21(24):4394–4400, 2005.
- [38] Kei-Hoi Cheung, Deyun Pan, Andrew Smith, Michael Seringhaus, Shawn M. Douglas, and Mark Gerstein. An XML-based approach to integrating heterogeneous yeast genome data. In *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS)*, pages 236–242, 2004.
- [39] Kei-Hoi Cheung, Kevin Y. Yip, Andrew Smith, Remko de Knikker, Andy Masiar, and Mark Gerstein. YeastHub: A semantic web use case ofr integrating data in the life sciences domain. *Bioinformatics*, 21(Supp. 1):i85–i96, 2005.
- [40] Kei-Hoi Cheung, Kevin Y. Yip, Jeffrey P. Townsend, and Mathew Scotch. HCLS 2.0/3.0: Health care and life sciences data mashup using web 2.0/3.0. *Journal of Biomedical Informatics*, 41:694–705, 2008.

- [41] Jo-Lan Chung, Wei Wang, and Philip E. Bourne. Exploiting sequence and structure homologs to identify protein-protein binding sites. *Proteins: Structure, Function, and Bioinformatics*, 62:630–640, 2006.
- [42] Jo-Lan Chung, Wei Wang, and Philip E. Bourne. High-throughput identification of interacting protein-protein binding sites. *BMC Bioinformatics*, 8(223), 2007.
- [43] A. A. Cohen, N. Geva-Zatorsky, E. Eden, M. Frenkel-Morgenstern, I. Issaeva, A. Sigal, R. Milo, C. Cohen-Saidon, Y. Liron, Z. Kam, L. Cohen, T. Danon, N. Perzov, and U. Alon. Dynamic proteomics of individual cancer cells in response to a drug. *Science*, 322(5907):1511–1516, 2008.
- [44] Germund Dahlquist and Åke Björck. *Numerical Methods*. Dover, 2003.
- [45] Antoine Danchin. *The Delphic Boat – What Genomes Tell Us*. Harvard University Press, 2003.
- [46] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Michael Erdmann, and Ian Horrocks. The semantic web: the roles of XML and RDF. *IEEE Internet Computing*, 4:63–73, 2000.
- [47] John P. Dekker, Anthony Fodor, Richard W. Aldrich, and Gary Yellen. A perturbation-based method for calculating explicit likelihood of evolutionary covariance in multiple sequence alignments. *Bioinformatics*, 20(10):1565–1572, 2004.
- [48] Minghua Deng, Shipra Mehta, Fengzhu Sun, and Ting Chen. Inferring domain-domain interactions from protein-protein interactions. *Genome Research*, 12(10), 2002.

- [49] Matthew W. Dimmic, Melissa J. Hubisz, Carlos D. Bustamante, and Rasmus Nielsen. Detecting coevolving amino acid sites using bayesian mutational mapping. *Bioinformatics*, 21(Suppl. 1):i126–i135, 2005.
- [50] The DREAM (dialogue for reverse engineering assessments and methods) project. [http://wiki.c2b2.columbia.edu/dream/index.php/Main\\_Page](http://wiki.c2b2.columbia.edu/dream/index.php/Main_Page).
- [51] The DREAM3 challenges. [http://wiki.c2b2.columbia.edu/dream/index.php/The\\_DREAM3\\_Challenges](http://wiki.c2b2.columbia.edu/dream/index.php/The_DREAM3_Challenges).
- [52] Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155–161, Cambridge, MA, 1997. MIT Press.
- [53] Jay C. Dunlap, Katherine A. Borkovich, Matthew R. Henn, Gloria E. Turner, Matthew S. Sachs, N. Louise Glass, Kevin McCluskey, Michael Plamann, James E. Galagan, Bruce W. Birren, Richard L. Weiss, Jeffrey P. Townsend, Jennifer J. Loros, Mary Anne Nelson, Randy Lambregts, Hildur V. Colot, Gyungsoon Park, Patrick Collopy, Carol Ringelberg, Christopher Crew, Liubov Litvinkova, Dave DeCaprio, Heather M. Hood, Susan Curilla, Mi Shi, Matthew Crawford, Michael Koerhsen, Phil Montgomery, Lisa Larson, Matthew Pearson, Takao Kasuga, Chaoguang Tian, Meray BaŞtürkmen, Lorena Altamirano, and Junhuan Xu. Enabling a community to dissect an organism: Overview of the neurospora functional genomics project. *Advances in Genetics*, 57:49–96, 2007.

- [54] Ron Edgar, Michael Domrachev, and Alex E. Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210, 2002.
- [55] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, 1998.
- [56] Adrian H. Elcock and J. Andrew McCammon. Identification of protein oligomerization states by analysis of interface conservation. *Proceedings of the National Academy of Sciences of the United States of America*, 98(6):2990–2994, 2001.
- [57] EPA terms of environment: Glossary, abbreviations and acronyms. <http://www.epa.gov/OCEPAterms/dterms.html>.
- [58] Jordi Espadaler, Oriol Romero-Isart, Richard M. Jackson, and Baldo Oliva. Prediction of protein-protein interactions using distant conservation of sequence patterns and structure relationships. *Bioinformatics*, 21(16):3360–3368, 2005.
- [59] Piero Fariselli, Florencio Pazos, Alfonso Valencia, and Rita Casadio. Prediction of protein-protein interaction sites in heterocomplexes with neural networks. *European Journal of Biochemistry*, 269(5):1356–1361, 2002.
- [60] Usama Fayyad and Ramasamy Uthurusamy. Data mining and knowledge discovery in databases. *Communications of the ACM*, 39:24–26, 1996.
- [61] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge and Data Mining*. AAAI Press, 1996.
- [62] D. Fenyő. The biopolymer markup language. *Bioinformatics*, 15(4):339–340, 1999.

- [63] Robert D. Finn, Mhairi Marshall, and Alex Bateman. iPfam: Visualization of protein-protein interactions in PDB at domain and amino acid resolutions. *Bioinformatics*, 21(3):410–412, 2005.
- [64] Robert D. Finn, John Tate, Jaina Mistry, Penny C. Coggill, Stephen John Sammut, Hans-Rudolf Hotz, Goran Ceric, Kristoffer Forslund, Sean R. Eddy, Erik L. L. Sonnhammer, and Alex Bateman. The Pfam protein families database. *Nucleic Acids Research*, 36:D281–D288, 2008.
- [65] Boris Galitsky. Revealing the set of mutually correlated positions for the protein families of immunoglobulin fold. *In Silico Biology*, 3(0022), 2003.
- [66] Yong Gao, June Kinoshita, Elizabeth Wu, Eric Miller, Ryan Lee, Andy Seaborne, Steve Cayzer, and Tim Clark. SWAN: a distributed knowledge infrastructure for alzheimer disease research. *Journal of Web Semantics*, 4(3):222–228, 2006.
- [67] Timothy S. Gardner, Diego di Bernardo, David Lorenz, and James J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–205, 2003.
- [68] Audrey P. Gasch, Paul T. Spellman, Camilla M. Kao, Orna Carmel-Harel, Michael B. Eisen, Gisela Storz, David Botstein, and Patrick O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11(12):4241–4257, 2000.
- [69] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dumpelfeld, Angela Edelmann, Marie-Anne Heurtier, Verena Hoffman, Christian

- Hoefert, Karin Klein, Manuela Hudak, Anne-Marie Michon, Malgorzata Schelder, Markus Schirle, Marita Remor, Tatjana Rudi, Sean Hooper, Andreas Bauer, Tewis Bouwmeester, Georg Casari, Gerard Drewes, Gitte Neubauer, Jens M. Rick, Bernhard Kuster, Peer Bork, Robert B. Russell, and Giulio Superti-Furga. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440:631–636, 2006.
- [70] Justin Gerke, Kim Lorenz, and Barak Cohen. Genetic interactions between transcription factors cause natural variation in yeast. *Science*, 323(5913):498–501, 2009.
- [71] Mark Gerstein, Erik L. L. Sonnhammer, and Cyrus Chothia. Volume changes in protein evolution. *Journal of Molecular Biology*, 236:1067–1078, 1994.
- [72] Guri Giaever, Angela M. Chu, Li Ni, Carla Connelly, Linda Riles, Steeve Veronneau, Sally Dow, Ankuta Lucau-Danila, Keith Anderson, Bruno Andre, Adam P. Arkin, Anna Astromoff, Mohamed El Bakkoury, Rhonda Bangham, Rocio Benito, Sophie Brachat, Stefano Campanaro, Matt Curtiss, Karen Davis, Adam Deutschbauer, Karl-Dieter Entian, Patrick Flaherty, Francoise Foury, David J. Garfinkel, Mark Gerstein, Deanna Gotte, Ulrich Guldener, Johannes H. Hegemann, Svenja Hempel, Zelek Herman, Daniel F. Jaramillo, Diane E. Kelly, Steven L. Kelly, Peter Kotter, Darlene LaBonte, David C. Lamb, Ning Lan, Hong Liang, Hong Liao, Lucy Liu, Chuanyun Luo, Marc Lussier, Rong Mao, Patrice Menard, Siew Loon Ooi, Jose L. Revuelta, Christopher J. Roberts, Matthias Rose, Petra Ross-Macdonald, Bart Scherens, Greg Schimmack, Brenda Shafer, Daniel D. Shoemaker, Sharon Sookhai-Mahadeo, Reginald K. Storms, Jeffrey N. Strathern, Giorgio Valle, Marleen Voet, Guido Volckaert, Ching yun Wang, Teresa R. Ward, Julie Wilhelmy, Elizabeth A. Winzeler, Yonghong Yang, Grace Yen, Elaine Youngman, Kexin Yu, Howard Bussey, Jef D. Boeke, Michael

- Snyder, Peter Philippsen, Ronald W. Davis, and Mark Johnston. Functional profiling of the *saccharomyces cerevisiae* genome. *Nature*, 418(6896):387–391, 2002.
- [73] Michael A. Gilchrist, Laura A. Salter, and Andreas Wagner. A statistical framework for combining and interpreting proteomic datasets. *Bioinformatics*, 20(5):689–700, 2004.
- [74] Gregory B. Gloor, Louise C. Martin, Lindi M. Wahl, and Stanley D. Dunn. Mutual information in protein multiple sequence alignments reveals two classes of coevolving positions. *Biochemistry*, 44(19):7156–7165, 2005.
- [75] Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics*, 18:309–317, 1994.
- [76] Chern-Sing Goh, Adrew A. Bogan, Marcin Joachimiak, Dirk Walther, and Fred E. Cohen. Co-evolution of proteins with their interaction partners. *Journal of Molecular Biology*, 299:283–293, 2000.
- [77] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institutes thesaurus and ontology. *Journal of Web Semantics*, 1:75–80, 2003.
- [78] Shawn M. Gomez, Shaw-Hwa Lo, and Andrey Rzhetsky. Probabilistic prediction of unknown metabolic and signal-transduction networks. *Genetics*, 159(3):1291–1298, 2001.

- [79] Shawn M. Gomez, William Stafford Noble, and Andrey Rzhetsky. Learning to predict protein-protein interactions from protein sequences. *Bioinformatics*, 19(15):1875–1881, 2003.
- [80] Nabil Guelzim, Samuele Bottani, Paul Bourguine, and Francois Képès. Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics*, 31(1):60–63, 2002.
- [81] Mainak Guharoy and Pinak Chakrabarti. Conservation and relative importance of residues across protein-protein interfaces. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15447–15452, 2005.
- [82] Katia S Guimarães, Raja Jothi, Elena Zotenko, and Teresa M Przytycka. Predicting domain-domain interactions using a parsimony approach. *Genome Biology*, 7(R104), 2006.
- [83] Jie Guo, Xiaomei Wu, Da-Yong Zhang, and Kui Lin. Genome-wide inference of protein interaction sites: Lessons from the yeast high-quality negative protein-protein interaction dataset. *Nucleic Acids Research*, 36(6):2002–2011, 2008.
- [84] L.M. Haas, P.M. Schwarz, P. Kodali, E. Kotlar, J.E. Rice, and W.C. Swope. DiscoveryLink: a system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(2):489–511, 2001.
- [85] Inbal Halperin, Haim Wolfson, and Ruth Nussinov. Correlated mutations: Advances and limitations. a study on fusion proteins and on the cohesin-dockerin families. *Proteins: Structure, Function, and Bioinformatics*, 63:832–845, 2006.



- [86] Jing-Dong J. Han, Nicolas Bertin, Tong Hao, Debra S. Goldberg, Gabriel F. Berriz, Lan V. Zhang, Denis Dupuy, Albertha J. M. Walhout, Michael E. Cusick, Frederick P. Roth, and Marc Vidal. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995):88–93, 2004.
- [87] Daniel Hanisch, Ralf Zimmer, and Thomas Lengauer. ProML - the protein markup language for specification of protein sequences, structures and families. *In Silico Biology*, 2(0029), 2002.
- [88] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.
- [89] Henning Hermjakob, Luisa Montecchi-Palazzi, Gary Bader, Jerome Wojcik, Lukasz Salwinski, Arnaud Ceol, Susan Moore, Sandra Orchard, Ugis Sarkans, Christian von Mering, Bernd Roechert, Sylvain Poux, Eva Jung, Henning Mersch, Paul Kersey, Michael Lappe, Yixue Li, Rong Zeng, Debashis Rana, Macha Nikolski, Holger Husi, Christine Brun, K Shanker, Seth G N Grant, Chris Sander, Peer Bork, Weimin Zhu, Akhilesh Pandey, Alvis Brazma, Bernard Jacq, Marc Vidal, David Sherman, Pierre Legrain, Gianni Cesareni, Ioannis Xenarios, David Eisenberg, Boris Steipe, Chris Hogue, and Rolf Apweiler. The HUPO PSI's molecular interaction format x a community standard for the representation of protein interaction data. *Nature Biotechnology*, 22(2):177–183, 2004.
- [90] Hailiang Huang, Bruno Jedynak, and Joel S. Bader. Where have all the interactions gone? estimating the coverage of two-hybrid protein interaction maps. *PLoS Computational Biology*, 3(e214), 2007.

- [91] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, the rest of the SBML Forum: A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremeling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.
- [92] Won-Ki Huh, James V. Falvo, Luke C. Gerke, Adam S. Carroll, Russell W. Howson, Jonathan S. Weissman, and Erin K. O’Shea. Global analysis of protein localization in budding yeast. *Nature*, 425:686–691, 2003.
- [93] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Beatrice A. Cuche, Edouard de Castro, Corinne Lachaize, Petra S. Langendijk-Genevaux, and Christian J. A. Sigrist. The 20 years of PROSITE. *Nucleic Acids Research*, 36:D245–D249, 2008.
- [94] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.
- [95] Sarah Hunter, Rolf Apweiler, Teresa K. Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, Robert D. Finn, Julian Gough, Daniel Haft, Nicolas Hulo, Daniel Kahn, Elizabeth Kelly, Aurelie Laugraud, Ivica Letunic, David Lonsdale, Rodrigo Lopez,

- Martin Madera, John Maslen, Craig McAnulla, Jennifer McDowall, Jaina Mistry, Alex Mitchell, Nicola Mulder, Darren Natale, Christine Orengo, Antony F. Quinn, Jeremy D. Selengut, Christian J. A. Sigrist, Manjula Thimma, Paul D. Thomas, Franck Valentin, Derek Wilson, Cathy H. Wu, and Corin Yeats. InterPro: the integrative protein signature database. *Nucleic Acids Research*, 37:D211–D215, 2009.
- [96] Mudassar Iqbal, Alex A. Freitas, Colin G. Johnson, and Massimo Vergassola. Message-passing algorithms for the prediction of protein domain interactions from protein-protein interaction data. *Bioinformatics*, 24(18):2064–2070, 2008.
- [97] Takashi Ito, Kosuke Tashiro, Shigeru Muta, Ritsuko Ozawa, Tomoko Chiba, Mayumi Nishizawa, Kiyoshi Yamamoto, Satoru Kuhara, and Yoshiyuki Sakaki. Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 97:1143–1147, 2000.
- [98] Ronald Jansen, Haiyuan Yu, Dov Greenbaum, Yuval Kluger, Nevan J. Krogan, Sambath Chung, Andrew Emili, Michael Snyder, Jack F. Greenblatt, and Mark Gerstein. A bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–453, 2003.
- [99] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D’Eustachio, E. Schmidt, B. de Bono, B. Jassal, G.R. Gopinath, G.R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: A knowledgebase of biological pathways. *Nucleic Acids Research*, 33(Data-base Issue):D428–D432, 2005.

- [100] Raja Jothi, Praveen F. Cherukuri, Asba Tasneem, and Teresa M. Przytycka. Co-evolutionary analysis of domains in interacting proteins reveals insights into domain-domain interactions mediating protein-protein interactions. *Journal of Molecular Biology*, 362:861–875, 2006.
- [101] Ravi S. Kamath, Andrew G. Fraser, Yan Dong, Gino Poulin, Richard Durbin, Monica Gotta, Alexander Kanapin, Nathalie Le Bot, Sergio Moreno, Marc Sohrmann, David P. Welchman, Peder Zipperlen, and Julie Ahringer. Systematic functional analysis of the *Caenorhabditis elegans* genome using RNAi. *Nature*, 421(6920):231–237, 2003.
- [102] Ryan Kelley and Trey Ideker. Systematic interpretation of genetic interactions using protein networks. *Nature Biotechnology*, 23(5):561–566, 2005.
- [103] S. Kerrien, Y. Alam-Faruque, B. Aranda, I. Bancarz, A. Bridge, C. Derow, E. Dimer, M. Feuermann, A. Friedrichsen, R. Huntley, C. Kohler, J. Khadake, C. Leroy, A. Liban, C. Lieftink, L. Montecchi-Palazzi, S. Orchard, J. Risse, K. Robbe, B. Roechert, D. Thorneycroft, Y. Zhang, R. Apweiler, and H. Hermjakob. IntAct – open source resource for molecular interaction data. *Nucleic Acids Research*, 35:D561–D565, 2007.
- [104] Philip M Kim, Long J. Lu, Yu Xia, and Mark B Gerstein. Relating three-dimensional structures to protein networks provides evolutionary insights. *Science*, 314(5807):1938–1941, 2006.
- [105] Wan Kyu Kim, Andreas Henschel, Christof Winter, and Michael Schroeder. The many faces of protein-protein interactions: A compendium of interface geometry. *PLoS Computational Biology*, 2(e124), 2006.

- [106] Hiroaki Kitano. Systems biology: a brief overview. *Science*, 295(5560):1662–1664, 2002.
- [107] Greg Kraushaar, Rajesh Patel, and Grant W. Stoneham. West nile virus: A case report with flaccid paralysis and cervical spinal cord. *American Journal of Neuro-radiology*, 26(1):26–29, 2005.
- [108] Nevan J. Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P. Tikuisis, Thanuja Punna, Jose M. Peregrin-Alvarez, Michael Shales, Xin Zhang, Michael Davey, Mark D. Robinson, Alberto Paccanaro, James E. Bray, Anthony Sheung, Bryan Beattie, Dawn P. Richards, Veronica Canadien, Atanas Lalev, Frank Mena, Peter Wong, Andrei Starostine, Myra M. Canete, James Vlasblom, Samuel Wu, Chris Orsi, Sean R. Collins, Shamanta Chandran, Robin Haw, Jennifer J. Rilstone, Kiran Gandi, Natalie J. Thompson, Gabe Musso, Peter St Onge, Shaun Ghanny, Mandy H. Y. Lam, Gareth Butland, Amin M. Altaf-UL, Shigehiko Kanaya, Ali Shilatifard, Erin O’Shea, Jonathan S. Weissman, C. James Ingles, Timothy R. Hughes, John Parkinson, Mark Gerstein, Shoshana J. Wodak, Andrew Emili, , and Jack F. Greenblatt. Global landscape of protein complexes in the yeast *saccharomyces cerevisiae*. *Nature*, 440:637–643, 2006.
- [109] Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of Bioinformatics and Computational Biology*, 3(3):527–550, 2005.
- [110] Anuj Kumar, Kei-Hoi Cheung, Nick Tosches, Peter Masiar, Yang Liu, Perry Miller, and Michael Snyder. The TRIPLES database: a community resource for yeast molec-

- ular biology. *Nuclear Acids Research*, 30(1):73–75, 2002.
- [111] 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.
- [112] Stefan M. Larson, Ariel A. Di Nardo, and Alan R. Davidson. Analysis of covariation in an SH3 domain sequence alignment: Applications in tertiary contact prediction and the design of compensating hydrophobic core substitutions. *Journal of Molecular Biology*, 303:433–4446, 2000.
- [113] Hyunju Lee, Minghua Deng, Fengzhu Sun, and Ting Chen. An integrated approach to the prediction of domain-domain interactions. *BMC Bioinformatics*, 7(269), 2006.
- [114] Thomas J Lee, Yannick Pouliot, Valerie Wagner, Priyanka Gupta, David WJ Stringer-Calvert, Jessica D Tenenbaum, and Peter D Karp. BioWarehouse: a bioinformatics database warehouse toolkit. *BMC Bioinformatics*, 7(170), 2006.
- [115] Tong Ihn Lee, Nicola J. Rinaldi, Francois Robert, Duncan T. Odom, Ziv Bar-Joseph, Georg K. Gerber, Nancy M. Hannett, Christopher T. Harbison, Craig M. Thompson, Itamar Simon, Julia Zeitlinger, Ezra G. Jennings, Heather L. Murray, D. Benjamin Gordon, Bing Ren, John J. Wyrick, Jean-Bosco Tagne, Thomas L. Volkert, Ernest Fraenkel, David K. Gifford, and Richard A. Young. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, 298(5594):799–804, 2002.
- [116] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for SVM protein classification. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Kevin Lauderdale, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing 2002*, pages 564–575, 2002.

- [117] Christina Leslie and Rui Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.
- [118] Christina S. Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.
- [119] Ivica Letunic, Tobias Doerks, and Peer Bork. SMART 6: Recent updates and new developments. *Nucleic Acids Research*, 37:D229–D232, 2009.
- [120] Yin Liu, Nianjun Liu, and Hongyu Zhao. Inferring protein-protein interactions through high-throughput interaction data from diverse organisms. *Bioinformatics*, 21(15):3279–3285, 2005.
- [121] Steve W. Lockless and Rama Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286(5438):295–299, 1999.
- [122] N. Lurie. Administrative data and outcomes research. *Medical Care*, 28(10):867–869, 1990.
- [123] Nicholas M. Luscombe, M. Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A. Teichmann, and Mark Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431(7006):308–312, 2004.
- [124] Daniel Marbach, Thomas Schaffter, Claudio Mattiussi, and Dario Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 2009. (in press).
- [125] Elaine R Mardis. ChIP-seq: Welcome to the new frontier. *Nature Methods*, 4(8):613–614, 2007.

- [126] L. C. Martin, G. B. Gloor, S. D. Dunn, and L. M. Wahl. Using information theory to search for co-evolving residues in proteins. *Bioinformatics*, 21(22):4116–4124, 2005.
- [127] Shawn Martin, Diana Roe, and Jean-Loup Faulon. Predicting protein-protein interactions using signature products. *Bioinformatics*, 21(2):218–226, 2005.
- [128] A. D. McLachlan. Tests for comparing related amino-acid sequences cytochrome c and cytochrome c551. *Journal of Molecular Biology*, 61:409–424, 1971.
- [129] Pedro Mendes, Wei Sha, and Keying Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19(Suppl. 2):ii122–ii129, 2003.
- [130] John Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London*, 209:415–446, 1909.
- [131] H. W. Mewes, D. Frishman, U. Güldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and B. Weil. MIPS: A database for genomes and protein sequences. *Nucleic Acids Research*, 20(1):31–34, 2002.
- [132] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.
- [133] Tom M. Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression, 2005. <http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf>.
- [134] Chad L Myers, Drew Robson, Adam Wible, Matthew A Hibbs, Camelia Chiriac, Chandra L Theesfeld, Kara Dolinski, and Olga G Troyanskaya. Discovery of biological networks from diverse functional genomic data. *Genome Biology*, 6, 2005.



- [135] Chad L. Myers and Olga G. Troyanskaya. Context-sensitive data integration and prediction of biological networks. *Bioinformatics*, 23(17):2322–2330, 2007.
- [136] Eric K. Neumann and Dennis Quan. BioDash: a semantic web dashboard for drug development. In Russ B. Altman, Tiffany Murray, Teri E. Klein, A. Keith Dunker, and Lawrence Hunter, editors, *Pacific Symposium on Biocomputing 2006*, pages 176–187, 2006.
- [137] See-Kiong Ng, Zhuo Zhang, and Soon-Heng Tan. Integrative approach for computationally inferring protein domain interactions. *Bioinformatics*, 19(8):923–929, 2003.
- [138] Tom M. W. Nye, Carlo Berzuini, Walter R. Gilks, M. Madan Babu, and Sarah A. Teichmann. Statistical analysis of domains in interacting protein pairs. *Bioinformatics*, 21(7):993–1001, 2005.
- [139] Daniel R. O’Leary, Anthony A. Marfin, Susan P. Montgomery, Aaron M. Kipp, Jennifer A. Lehman, Brad J. Biggerstaff, Veronica L. Elko, Peggy D. Collins, John E. Jones, and Grant L. Campbell. The epidemic of west nile virus in the united states, 2002. *Vector Borne Zoonotic Discovery*, 4(1):61–70, 2004.
- [140] Tim O’Reilly. What is web 2.0 – design patterns and business models for the next generation of software, 2005. <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.
- [141] Ainslie B Parsons, Renee L Brost, Huiming Ding, Zhijian Li, Chaoying Zhang, Bilal Sheikh, Grant W Brown, Patricia M Kane, Timothy R Hughes, and Charles Boone. Integration of chemical-genetic and genetic interaction data links bioactive compounds to cellular target pathways. *Nature Biotechnology*, 22(1):62–69, 2003.

- [142] Prinaka Patel, Kevin Y. Yip, Donald M. Engelman, and Mark Gerstein. (Unpublished data).
- [143] Florencio Pazos, Manuela Helmer-Citterich, Gabriele Ausiello, and Alfonso Valencia. Correlated mutations contain information about protein-protein interaction. *Journal of Molecular Biology*, 271:511–523, 1997.
- [144] Florencio Pazos and Alfonso Valencia. In silico two-hybrid system for the selection of physically interacting protein pairs. *Proteins: Structure, Function, and Bioinformatics*, 47:219–227, 2002.
- [145] Patrick G A Pedrioli, Jimmy K Eng, Robert Hubley, Mathijs Vogelzang, Eric W Deutsch, Brian Raught, Brian Pratt, Erik Nilsson, Ruth H Angeletti, Rolf Apweiler, Kei Cheung, Catherine E Costello, Henning Hermjakob, Sequin Huang, Randall K Julian Jr, Eugene Kapp, Mark E McComb, Stephen G Oliver, Gilbert Omenn, Norman W Paton, Richard Simpson, Richard Smith, Chris F Taylor, Weimin Zhu, and Ruedi Aebersold. A common open representation of mass spectrometry data and its application to proteomics research. *Nature Biotechnology*, 22(11):1459–1466, 2004.
- [146] Matteo Pellegrini, Edward M. Marcotte, Michael J. Thompson, David Eisenberg, and Todd O. Yeates. Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 96:4285–4288, 1999.
- [147] David D. Pollock, William R. Taylor, and Nick Goldman. Coevolving protein residues: Maximum likelihood identification and relationship to structure. *Journal of Molecular Biology*, 287:187–198, 1999.

- [148] 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:D767–D772, 2009.
- [149] Province of british columbia terminal weevils guidebook table of contents. <http://www.for.gov.bc.ca/tasb/legsregs/fpc/fpcguide/weevil/glossary.htm>.
- [150] Pål Puntervoll, Rune Linding, Christine Gemund, Sophie Chabanis-Davidson, Morten Mattingsda, Scott Cameron, David M. A. Martin, Gabriele Ausiello, Barbara Brannetti, Anna Costantini, Fabrizio Ferre, Vincenza Maselli, Allegra Via, Gianni Cesareni, Francesca Diella, Giulio Superti-Furga, Lucjan Wyrwicz, Chenna Ramu, Caroline McGuigan, Rambabu Gudavalli, Ivica Letunic, Peer Bork, Leszek Rychlewski, Bernhard Kuster, Manuela Helmer-Citterich, William N. Hunter, Rein Aasland, and Toby J. Gibson. ELM server: a new resource for investigating short functional sites in modular eukaryotic proteins. *Nucleic Acids Research*, 31(13):3625–3630, 2003.
- [151] Jian Qiu and Stafford Noble. Predicting co-complexed protein pairs from heterogeneous data. *PLoS Computational Biology*, 4(e1000054), 2008.
- [152] Sanguthevar Rajasekaran, Sudha Balla, Patrick Gradie, Michael R. Gryk, Krishna Kadaveru, Vamsi Kundeti, Mark W. Maciejewski, Tian Mi, Nicholas Rubino, Jay

- Vyas, and Martin R. Schiller. Minimoto miner 2nd release: a database and web system for motif search. *Nucleic Acids Research*, 37:D185–D190, 2009.
- [153] Huzefa Rangwala and George Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005.
- [154] William Reisen and Aaron C. Brault. West nile virus in north america: Perspectives on epidemiology and intervention. *Pest Management Science*, 63(7):641–646, 2007.
- [155] John Jeremy Rice, Yuhai Tu, and Gustavo Stolovitzky. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics*, 21(6):765–773, 2004.
- [156] Robert Riley, Christopher Lee, Chiara Sabatti, and David Eisenberg. Inferring protein domain interactions from databases of interacting proteins. *Genome Biology*, 6(R89), 2005.
- [157] Volker Roth, Julian Laub, Joachim M Buhmann, and Klaus R Müller. Going metric: Denoising pairwise data. In Suzanna Becker, Sebastian Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.
- [158] Alan Ruttenberg, Tim Clark, William Bug, Matthias Samwald, Olivier Bodenreider, Helen Chen, Donald Doherty, Kerstin Forsberg, Yong Gao, Vipul Kashyap, June Kinoshita, Joanne Luciano, M Scott Marshall, Chimezie Ogbuji, Jonathan Rees, Susie Stephens, Gwendolyn T Wong, Elizabeth Wu, Davide Zaccagnini, Tonya Hongsermeier, Eric Neumann, Ivan Herman, and Kei-Hoi Cheung. Advancing translational research with the semantic web. *BMC Bioinformatics*, 8(S2), 2007.

- [159] Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.
- [160] 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:D449–D451, 2004.
- [161] Sven-Eric Schelhorn, Thomas Lengauer, and Mario Albrecht. An integrative approach for predicting interactions of protein regions. *Bioinformatics*, 24(ECCB):i35–i41, 2008.
- [162] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, 1995.
- [163] Brian K. Schoichet and Irwin D. Kuntz. Protein docking and complementarity. *Journal of Molecular Biology*, 221:327–346, 1991.
- [164] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors. *Kernel Methods in Computational Biology*. MIT Press, 2004.
- [165] Matthew Scotch, Kevin Y. Yip, and Kei-Hoi Cheung. Development of grid-like applications for public health using web 2.0 mashup techniques. *Journal of the American Medical Informatics Association*, 15(6):783–786, 2008.
- [166] Sohrab P Shah, Yong Huang, Tao Xu, Macaire MS Yuen, John Ling, and BF Francis Ouellette. Atlas v a data warehouse for integrative bioinformatics. *BMC Bioinformatics*, 6(34), 2005.

- [167] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.
- [168] Roded Sharan and Trey Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24(4):427–433, 2006.
- [169] Shai S. Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics*, 31(1):64–68, 2002.
- [170] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, 2008.
- [171] Benjamin A. Shoemaker and Anna R. Panchenko. Deciphering protein-protein interactions. part II. computational methods to predict protein and domain interaction partners. *PLoS Computational Biology*, 3(e43), 2007.
- [172] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistical and Computing*, 14(3):199–222, 2004.
- [173] Michael Socolich, Steve W. Lockless, William P. Russ, Heather Lee, Kevin H. Gardner, and Rama Ranganathan. Evolutionary information for specifying a protein fold. *Nature*, 437(7058):512–518, 2005.
- [174] Ta Tsen Soong, Kazimierz O. Wrzeszczynski, and Burkhard Rost. Physical protein-protein interactions predicted from microarrays. *Bioinformatics*, 24(22):2608–2614, 2008.

- [175] Paul T Spellman, Michael Miller, Jason Stewart, Charles Troup, Ugis Sarkans, Steve Chervitz, Derek Bernhart, Gavin Sherlock, Catherine Ball, Marc Lepage, Marcin Swiatek, WL Marks, Jason Goncalves, Scott Markel, Daniel Iordan, Mohammadreza Shojatalab, Angel Pizarro, Joe White, Robert Hubley, Eric Deutsch, Martin Senger, Bruce J Aronow, Alan Robinson, Doug Bassett, Christian J Stoeckert Jr, and Alvis Brazma. Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biology*, 3:research0046.1–0046.9, 2002.
- [176] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.
- [177] Einat Sprinzak, Shmuel Sattath, and Hanah Margalit. How reliable are experimental protein-protein interaction data? *Journal of Molecular Biology*, 327:919–923, 2003.
- [178] Einat Sprinzak and Hanah Margalit. Correlated sequence-signatures as markers of protein-protein interaction. *Journal of Molecular Biology*, 311(4):681–692, 2001.
- [179] Robert D. Stevens, Alan J. Robinson, and Carole A. Goble. myGrid: Personalised bioinformatics on the information grid. *Bioinformatics*, 19(Suppl. 1):i302–i304, 2003.
- [180] Gurol M. Süel, Steve W. Lockless, Mark A. Wall, and Rama Ranganathan. Evolutionarily conserved networks of residues mediate allosteric communication in proteins. *Nature Structural Biology*, 10(1):59–69, 2002.
- [181] Roman L. Tatusov, Eugene V. Koonin, and David J. Lipman. A genomic perspective on protein families. *Science*, 278(5338):631–637, 1997.

- [182] Amy Hin Yan Tong, Marie Evangelista, Ainslie B. Parsons, Hong Xu, Gary D. Bader, Nicholas Page, Mark Robinson, Sasan Raghbizadeh, Christopher W. V. Hogue, Howard Bussey, Brenda Andrews, Mike Tyers, and Charles Boone. Systematic genetic analysis with ordered arrays of yeast deletion mutants. *Science*, 294(5550):2364–2368, 2001.
- [183] Koji Tsuda. Support vector classification with asymmetric kernel function. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, pages 183–188, 1999.
- [184] Koji Tsuda, Shotaro Akaho, and Kiyoshi Asai. The *em* algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, 2003.
- [185] Peter Uetz, Loic Giot, Gerard Cagney, Traci A. Mansfield, Richard S. Judson, James R. Knight, Daniel Lockshon, Vaibhav Narayan, Maithreyan Srinivasan, Pascale Pochart, Alia Qureshi-Emili, Ying Li, Brian Godwin, Diana Conover, Theodore Kalbfleisch, Govindan Vijayadamodar, Meijia Yang, Mark Johnston, Stanley Fields, and Jonathan M. Rothberg. A comprehensive analysis of protein-protein interactions in *saccharomyces cerevisiae*. *Nature*, 403:623–627, 2000.
- [186] Alexei Vazquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Global protein function prediction from protein-protein interaction networks. *Nature Biotechnology*, 21(6):697–700, 2003.
- [187] Jean-Philippe Vert and Yoshihiro Yamanishi. Supervised graph inference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1433–1440, Cambridge, MA, 2005. MIT Press.



- [188] Tra Thi Vu and Jiri Vohradsky. Nonlinear differential equation model for quantification of transcriptional regulation applied to microarray data of *Saccharomyces cerevisiae*. *Nucleic Acids Research*, 35(1):279–287, 2007.
- [189] Andreas Wagner. Reconstructing pathways in large genetic networks from genetic perturbations. *Journal of Computational Biology*, 11(1):53–60, 2004.
- [190] Haidong Wang, Eran Segal, Asa Ben-Hur, Daphne Koller, and Douglas L. Brutlag. Identifying protein-protein interaction sites on a genome-wide scale. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1465–1472, Cambridge, MA, 2005. MIT Press.
- [191] Haidong Wang, Eran Segal, Asa Ben-Hur, Qianru Li, Marc Vidal, and Daphne Koller. InSite: a computational method for identifying protein-protein interaction binding sites on a proteome-wide scale. *Genome Biology*, 8(R192), 2007.
- [192] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009.
- [193] Mark D. Wilkinson and Matthew Links. BioMoby: an open source biological web services proposal. *Briefings in Bioinformatics*, 3(4):331–341, 2002.
- [194] Derek Wilson, Ralph Pethica, Yiduo Zhou, Charles Talbot, Christine Vogel, Martin Madera, Cyrus Chothia, and Julian Gough. SUPERFAMILY – sophisticated comparative genomics, data mining, visualization and phylogeny. *Nucleic Acids Research*, 37:D380–D386, 2009.
- [195] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

- [196] Sharyl L. Wong, Lan V. Zhang, Amy H. Y. Tong, Zhijian Li, Debra S. Goldberg, Oliver D. King, Guillaume Lesage, Marc Vidal, Brenda Andrews, Howard Bussey, Charles Boone, and Frederick P. Roth. Combining biological networks to predict genetic interactions. *Proceedings of the National Academy of Sciences of the United States of America*, 101(44):15682–15687, 2004.
- [197] Stefan Wuchty. Evolution and topology in the yeast protein interaction network. *Genome Research*, 14(7):1310–1314, 2004.
- [198] Ioannis Xenarios, Lukasz Salwinski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. DIP, the database of interacting proteins: A research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30(1):303–305, 2002.
- [199] 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.
- [200] Yoshihiro Yamanishi, Jean-Philippe Vert, and Minoru Kanehisa. Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 21(Suppl. 1):i468–i477, 2005.
- [201] Chen-Hsiang Yeang and David Haussler. Detecting coevolution in and among protein domains. *PLoS Computational Biology*, 3(e211), 2007.
- [202] Kevin Y. Yip, Roger P. Alexander, Koon-Kiu Yan, and Mark Gerstein. Combining multiple models in reconstructing in silico regulatory networks. In *The 5th Annual RECOMB Satellite on Regulatory Genomics, the 4th Annual RECOMB Satellite on Systems Biology, and the 3rd Annual DREAM Reverse Engineering Challenges*, 2008.

- [203] Kevin Y. Yip and Mark Gerstein. Training set expansion: an approach to improving the reconstruction of biological networks from limited and uneven reliable interactions. *Bioinformatics*, 25(2):243–250, 2009.
- [204] Kevin Y. Yip, Philip M. Kim, Drew McDermott, and Mark Gerstein. Multi-level learning: Improving the prediction of protein, domain and residue interactions by allowing information flow between levels. (submitted).
- [205] Kevin Y. Yip, Prianka Patel, Philip M. Kim, Donald M. Engelman, Drew McDermott, and Mark Gerstein. An integrated system for studying residue coevolution in proteins. *Bioinformatics*, 24(2):290–292, 2008.
- [206] Kevin Y. Yip, Haiyuan Yu, Philip M. Kim, Martin Schultz, and Mark Gerstein. The tYNA platform for comparative interactomics: A web tool for managing, comparing and mining multiple networks. *Bioinformatics*, 22(23):2968–2970, 2006.
- [207] Haiyuan Yu, Pascal Braun, Muhammed A Yildirim, Irma Lemmens, Kavitha Venkatesan, Julie Sahalie, Tomoko Hirozane-Kishikawa, Fana Gebreab, Na Li, Nicolas Simonis, Tong Hao, Jean-Francois Rual, Amelie Dricot, Alexei Vazquez, Ryan R Murray, Christophe Simon, Leah Tardivo, Stanley Tam, Nenad Svrzikapa, Changyu Fan, Anne-Sophie de Smet, Adriana Motyl, Michael E Hudson, Juyong Park, Xiaofeng Xin, Michael E Cusick, Troy Moore, Charlie Boone, Michael Snyder, Frederick P Roth, Albert-Laszlo Barabasi, Jan Tavernier, David E Hill, and Marc Vidal. High-quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110, 2008.
- [208] Haiyuan Yu, Dov Greenbaum, Hao Xin Lu, Xiaowei Zhu, and Mark Gerstein. Genomic analysis of essentiality within protein networks. *Trends in Genetics*, 20:227–231, 2004.

- [209] Haiyuan Yu, Alberto Paccanaro, Valery Trifonov, and Mark Gerstein. Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, 22(7):823–829, 2004.
- [210] Haiyuan Yu, Xiaowei Zhu, Dov Greenbaum, John Karro, and Mark Gerstein. TopNet: A tool for comparing biological subnetworks, correlating protein properties with topological statistics. *Nucleic Acids Research*, 32(1):328–337, 2004.
- [211] Andreas Zanzoni, Luisa Montecchi-Palazzi, Michele Quondam, Gabriele Ausiello, Manuela Helmer-Citterich, and Gianni Cesareni. MINT: A Molecular INTERaction Database. *FEBS Letter*, 513(1):135–140, 2002.
- [212] Li Zou, Scott N. Miller, and Edward T. Schmidtman. A GIS tool to estimate west nile virus risk based on a degree-day model. *Environmental Monitoring and Assessment*, 129(1-3):413–420, 2007.