Tel-Aviv University
The Raymond and Beverly Sackler Faculty of Exact Sciences
School of Computer Science

SPINE: A Framework for Signaling-Regulatory Pathway Inference from Cause-Effect Experiments

This thesis is submitted in partial fulfillment
of the requirements towards the M.Sc. degree
Tel-Aviv University
School of Computer Science

by
Oved Ourfali

The research work in this thesis has been carried out
under the supervision of Dr. Roded Sharan.

March, 2007

Acknowledgments:

I would like to thank my thesis advisor, Roded Sharan, for the initial idea, the close guidance and, and the fruitful ideas. I learned from him a lot while working on this thesis.
I would also like to thank Eytan Ruppin for his guidance, comments and ideas.
Finally, I would like to thank Tomer Shlomi for his explanations throughout my work, and his contribution to basic ideas behind it.

Abstract

The complex program of gene expression allows the cell to cope with changing genetic, developmental and environmental conditions. The accumulating large-scale measurements of gene knockout effects and molecular interactions allow us to begin to uncover regulatory and signaling pathways within the cell that connect causal to affected genes on a network of physical interactions.

We present a novel framework, SPINE, for Signaling-regulatory Pathway INferencE. The framework aims at explaining gene expression experiments in which a gene is knocked out and as a result multiple genes change their expression levels. To this end, an integrated network of protein-protein and protein-DNA interactions is constructed, and signaling pathways connecting the causal gene to the affected genes are searched for in this network. The reconstruction problem is translated into that of assigning an activation/repression attribute with each protein so as to explain (in expectation) a maximum number of the knockout effects observed. We provide an integer programming formulation for the latter problem and solve it using a commercial solver.

We validate the method by applying it to a yeast subnetwork that is involved in mating. In cross validation tests, SPINE obtains very high accuracy in predicting knockout effects (99%). Next, we apply SPINE to the entire yeast network to predict protein effects and reconstruct signaling and regulatory pathways. Overall, we are able to infer 861 paths with confidence and assign effects to 183 genes. The predicted effects are found to be in high agreement with current biological knowledge.

Our work was presented in the 15th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB) and the 6th European Conference on Computational Biology (ECCB), and was published in Bioinformatics [24].

The algorithm and data are available at `http://cs.tau.ac.il/~roded/SPINE.html`.

Contents

Chapter 1

Introduction

High throughput technologies are routinely used to map molecular interaction within the cell. These include chromatin immuno-precipitation experiments [18] for measuring protein-DNA interactions, and yeast two-hybrid assays [8] and co-immunoprecipitation screens [10] for measuring protein-protein interactions (PPIs). The resulting maps promise to shed light on the mechanisms that allow the propagation of certain signals within the cell. In particular, we aim at explaining gene expression experiments in which a gene is knocked out and as a result multiple genes change their expression levels.

A series of works concerned the inference of physical interactions in signaling-regulatory networks from perturbation experiments [16, 2, 32]. The problem of explaining knockout experiments using a physical network was introduced by Yeang *et al.* [33]. The authors looked at a specific setting of the problem where the objective is to infer for each edge of the network a direction, specifying the direction in which information flows through that interaction, and a sign, representing the regulatory effect of the interaction (activation or repression). The annotated network can then be used for pathway inference, although this inference problem was not explicitly treated by Yeang *et al.*

To tackle the network annotation problem, Yeang *et al.* assumed that explanatory pathways should satisfy several constraints. In particular, they should be directed from the knockout gene to the affected gene, and the aggregate sign along the pathway's edges should complement the sign of the knockout effect. They devised a probabilistic framework for the annotation task, in which the different constraints are translated into potential functions (for

edges, paths and knockouts), and the objective is to maximize the product of all potentials. A follow-up work [34] devised methods for validation and refinement of the network model. A similar model was used by Yeang *et al.* [35] to integrate metabolic data with regulatory interactions and gene-knockout data. Workman *et al.* [31] used the same model to infer the regulatory pathways that control the response to DNA damage.

While Yeang *et al.* provided an elegant formulation of the annotation problem, their solution method suffers from several shortcomings: (i) The objective function does not distinguish between situations in which few knockout pairs are explained by many pathways each, and those in which many pairs are explained by few pathways each. Clearly, the latter should be preferable. (ii) The components in the objective function are multiplied, although they are dependent on one another, as also mentioned in [33]. (iii) The maximization procedure does not guarantee reaching a global optimum.

Here we propose a novel combinatorial model for the inference problem. Our optimization goal is to maximize the expected number of explained cause-effect pairs. We reformulate the problem as an integer programming problem and apply a commercial solver to it. Moreover, we provide a measure of confidence in the predictions made, which quantifies the change in the function being optimized when the prediction is flipped.

We validate the method by applying it to a yeast subnetwork involved in mating. In cross validation tests, SPINE obtains very high accuracy in predicting knockout effects (99%). Next, we apply SPINE to the entire yeast network to predict protein effects and reconstruct signaling and regulatory pathways. Overall, we are able to infer 861 paths with confidence and assign effects to 183 genes. The predicted effects are found to be in high agreement with current biological knowledge. Our results compare favorably to those attained by Yeang *et al.* [33] in both the small-scale and the large-scale applications.

The thesis is organized as follows: Chapter 2 gives general biological and computational background relevant to the issues addressed in this work. Chapter 3 presents the main problem, and a detailed description of the algorithm of Yeang *et al.* [33]. Chapter 4 describes SPINE our new method for inferring signaling pathways. Chapter 5 presents the results of applying SPINE to both small and large-scale networks. It also presents a comparison of SPINE's performance to a previous approach. Chapter 6 gives a brief summary of the thesis, conclusions and several ideas for further research. Finally, implementation issues appear in Chapter 7.

Chapter 2

Biological and Computational Background

This chapter gives both biological and computational background. The biological background section describes physical interaction networks and gene expression data. The computational background section surveys mathematical-programming optimization problems and common techniques for solving them.

2.1 Biological Background

The physical interactions that are the focus of this work are protein-protein interactions (PPIs) and protein-DNA interactions (regulatory interactions). Protein-protein interactions refer to the physical association of protein molecules. These interactions are critical to all cellular processes. For example, signals from the exterior of a cell are mediated to the inside of that cell by the interactions among signalling proteins. Proteins also interact with one another to form protein complexes, the backbone of cellular machinery.

Regulatory interactions are physical interactions between transcription factor proteins and DNA sequences that control the transcription of genes. Such interactions can either induce the transcription of a gene or suppress it.

In a knockout experiment, the cell is engineered to lack the expression and activity of one

or more genes. Then, the expression levels of all the other genes are measured. Knockout experiments are often performed in order to determine the functional role of a specific gene in the organism by studying the defects caused by the resulting mutation. The expression data we use refer to *single-gene knockout experiments*, i.e., experiments in which a single gene is perturbed.

2.1.1 Protein-Protein Interaction Networks

Protein-protein interaction data have increased dramatically throughout the last few years. There are two main methods for detecting protein interactions: *Yeast Two-Hybrid* (*Y2H*) and *Co-Immunoprecipitation* (*coIP*).

Yeast Two-Hybrid

The yeast two-hybrid technique [14, 30] allows the detection of pairwise protein interactions. This is done by running an experiment that would cause a reporter gene to be expressed if the two examined proteins physically interact. Y2H uses two protein domains of the yeast GAL4 protein that have specific functions: a DNA-binding domain, capable of binding to a DNA sequence, and an activation domain, capable of activating transcription of the monitored gene. The gene transcription process can only occur when both domains are present. Thus, the two proteins of interest are attached to binding and activation domains of the reporter gene. The protein that is attached to the DNA-binding domain is called the *bait*, and the protein that is attached to the activation domain is call the *prey*. If they interact, an active transcription unit will be formed and the reporter gene will be expressed, forming a protein product that can be detected (see Figure 2.1).

Co-Immunoprecipitation

In this method, the bait protein is marked by a tag. An antibody which recognizes the tag is used to trap the bait protein and precipitate it. In the precipitation process, any protein physically associated with the bait is precipitated as well. Following this, mass spectrometry is used to determine the identity of these prey proteins (see Figure 2.2). The advantage of this method is in its ability to discover interactions between a single bait and multiple prey

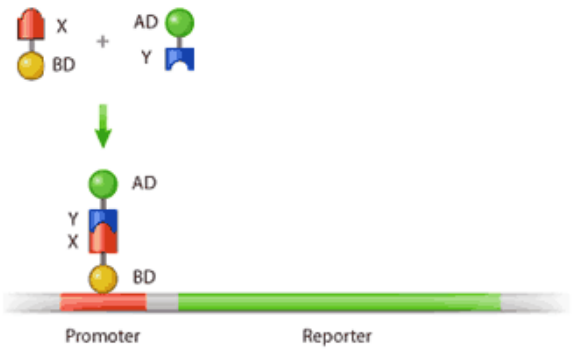

Figure 2.1: The yeast two hybrid process. The expression of the reporter gene is measured in order to detect protein-protein interactions. If proteins X and Y interact, their DNA-binding domain and activation domain will combine to form a functional transcriptional activator, which will induce the transcription of the reporter gene (figure taken from [28]).

proteins. However, direct interactions cannot be distinguished from interactions mediated by other proteins in a complex.

Interaction Reliability

Employing procedures such as Y2H and coIP, described above, has enabled the discovery of thousands of protein-protein interactions. However, those technologies still suffer from high rates of false positives. Hence, several approaches to assess the reliability of PPIs have been developed. One approach, by Deng *et al.* [7], estimates the reliabilities of different interaction data sources using the distribution of gene expression correlation coefficients. They separately considered experiments that report pairwise interactions like Y2H and those that report complex membership like mass spectrometry. Pairwise interactions from the literature and membership in protein complexes from Munich Information center for Protein Sequences (MIPS) [22] were used as the gold standard positive set in each case. Randomly chosen protein pairs formed the gold standard negative data set. Reliabilities for each data source were computed using a maximum likelihood scheme based on the expression pro-

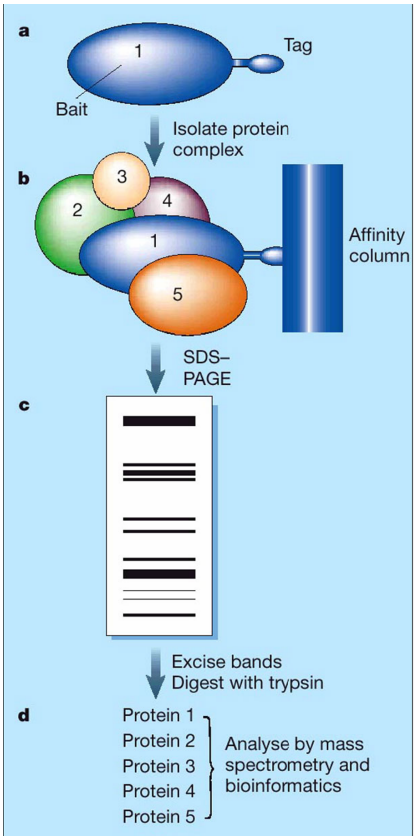

Figure 2.2: The co-IP assay. a) A specific protein bait is prepared and is attached to an affinity tag in order to allow the purification of the bait protein and the associated proteins. b) The bait protein then interacts with other proteins and is purified. c) The purified protein complex is resolved, and discrete protein bands are excited and digested into small peptide fragments. d) Peptides are identified using mass spectrometry methods. The identity of a protein associated with a given bait is determined by comparing its peptide fingerprint against known databases (figure taken from [21]).

files of each data source, based on the assumption that proteins that the distribution of gene expression correlations differs between pairs whose corresponding proteins interact and pairs whose corresponding proteins do not interact. In addition to assigning reliabilities to each such source, the authors also provided a conditional probability scheme to compute probabilities for groups of interactions observed in two or more data sources.

Other methods, suggested by Bader *et al.* [4] and Sharan *et al.* [29], assign confidence values to protein interactions using a logistic regression model. Briefly, in [29] true positive and true negative interactions were used to train a logistic regression model, which assigns each interaction a reliability score based on the experimental evidence for this interaction, which includes the type of experiments in which the interaction was observed, and the number of observations in each experimental type. The experiments were partitioned into four categories: co-immunoprecipitation screens, yeast two-hybrid assays, large scale experiments and small scale experiments. Throughout this thesis we used the logistic regression model that appears in [29].

2.1.2 Protein-DNA Interaction Networks

ChIP-chip, also known as *ChIP-on-chip* or *genome wide location analysis*, is a technology for isolating genomic sites occupied by specific DNA binding proteins in living cells [25]. These sites may indicate functions of various transcriptional regulators. In the term "ChIP-chip", "ChIP" refers to *chromatin immunoprecipitation* which is a method for isolating DNA fragments that are bound by specific DNA binding proteins. "Chip" refers to the DNA microarray technology [19] for measuring the concentrations of these DNA fragments. The DNA microarray probes can tile the whole genome, so the ChIP-chip data can be obtained over the whole genome in the form of a one-dimensional signal, where a peak in the signal is generally present at a protein binding site. Therefore, the protein binding sites can be located by detecting the peaks in the signal.

ChIP-chip consists of the following steps:

1. Binding: bound transcription factors and other DNA associated proteins are cross-linked to DNA with formaldehyde. (DNA and proteins are linked together by covalent bonds, which are more stable than hydrogen bonds, after this chemical reaction.)

2. Chopping: DNA sequences are chopped into small fragments using sonication while the transcription factors are still cross-linked to DNA. Thus, among all the chopped DNA fragments, some are bound by proteins, and the rest are not.

3. Isolation: DNA fragments bound by proteins are isolated by chromatin immunoprecipitation (ChIP). During ChIP, antibodies that can specifically recognize the DNA-binding protein of interest are added. These antibodies will then bind to their target protein and cause the whole complex to precipitate. Therefore, DNA segment bound by proteins can be isolated and enriched.

4. Cross-linking between DNA and protein is reversed and DNA is released, amplified by ligation-mediated polymerase chain reaction and labeled with a red fluorescent dye (Cy5). At the same time, a sample of DNA which is not enriched by the above immunoprecipitation process is also amplified by LM-PCR and labeled with a green fluorescent dye (Cy3).

5. Both IP-enriched and unenriched DNA pools of labeled DNA are hybridized to the same high-density oligonucleotide arrays (chip). On the array there are thousands to millions of spots containing probes to detect the existence of certain DNA fragments. A probe in the ChIP-chip technology is a single strand DNA segment of about 50 nucleotide long, complementary to their target sequence. The microarray is then scanned and two images, corresponding to Cy5 (IP enriched) and Cy3 (control) signals, respectively, are extracted.

Protein-DNA data has grown enormously during the past few years, using technologies such as ChIP-chip, described above. In contrast to technologies for discovering protein-protein interactions, ChIP-chip suffers less from false positives, and the reliability of each interaction is considered very high [18].

2.2 Computational Background

This section gives the computational background which forms the basis of this thesis. The algorithmic approach in this thesis is based on the field of mathematical-programming, especially optimization problems that are based on linear- and integer-Programming. We

start with describing linear programming and approaches for solving linear programming optimization problems. Next, we describe integer programming and approaches for solving integer programming optimization problems.

2.2.1 Linear Programming

Linear programming (LP) has been a fundamental topic in the development of the computational sciences. The subject has its origins in the early work of L.B.J. Fourier on solving systems of linear inequalities, dating back to the 1820's. More recently, a competition between two methods for solving LP problems, the simplex and interior point methods, has led to rapid improvements in the algorithmics of linear programming. This, combined with remarkable advances in computing hardware and software, have brought linear programming tools to the desktop, in a variety of application software for decision support. Linear programming has provided a fertile ground for the development of various algorithmic paradigms. Diverse topics such as symbolic computation, numerical analysis, computational complexity, computational geometry, combinatorial optimization, and randomized algorithms all have some linear programming connection. Moreover, linear-programming has a tremendous contribution to the field of resource allocation, which has many applications such as allocating national resources to local needs, choosing an investment with lower risk, agricultural planning, medical treatment planning, and etc.

A linear programming problem is stated as follows:

Maximize/Minimuze $Z = \sum_{j=1}^{n} c_j x_j$	
subject to	
$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$	$i = 1, 2, \ldots, m$

where Z is the objective function we wish to minimize/maximize, x_j $(j = 1, 2, \ldots, n)$ are the variables, c_j $(j = 1, 2, \ldots, n)$ are coefficients for the objective function, and a_{ij} and b_i $(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$ are coefficients for the different constraints.

The *Simplex* Method

The *simplex* method was developed by George Dantzig in 1974 and is considered as one of the most efficient methods for solving linear programming problems. A linear programming problem consists of a collection of linear inequalities on a number of real variables and a

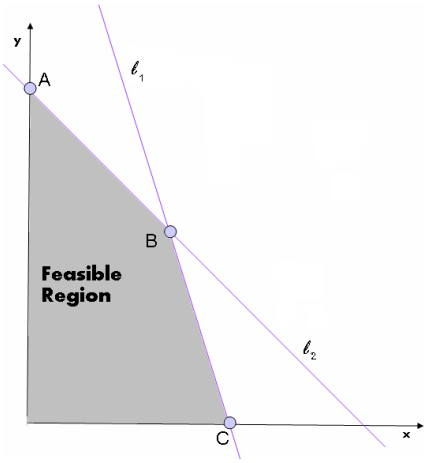

Figure 2.3: An example for a linear-programming problem in a 2-dimensional Euclidean space.

given linear functional (on these real variables) which is to be maximized or minimized. It is easier to understand the problem in geometric form, and for simplicity we will refer to the linear-programming problem under 2-dimensional Euclidean space. Figure 2.3 shows an example of such a problem, where x is the variable, l_1 and l_2 represent the constraints, and we wish to maximize the value of y.

The *feasible region*, i.e., the region in which every solution satisfies the constraints, is colored gray. A *corner-point solution* is a solution which is in an intersection of constraints. Two corner point solutions are *adjacent* if they are connected via a straight line that runs over the feasible solution. For example, in Figure 2.3 the points A,B and C are corner-point solutions, and the pairs A, B and B, C are adjacent corner solutions. The simplex method is based on the following observations:

1. If the problem has a single optimal feasible solution, then it must be a corner-point feasible solution.

2. If the problem has multiple optimal feasible solutions then at least two of them must be corner adjacent solutions.

3. The number of corner-point solutions is finite.

4. If a corner-solution has no adjacent corner solutions that have better objective value then it is the optimal solution.

According to observations 1 and 2, the optimal solution should be searched only among the corner-point feasible solutions. According to observation 3, the number of those solutions is finite. Finally, observation 4 gives a convenient optimality test.

Thus, the simplex method works as follows:

1. Start with a feasible corner-solution.

2. If there exists a better adjacent corner-solution, move to it.

3. Repeat step 2 until reaching optimality, i.e., until there is no better adjacent solution.

Generally, when looking at the n-dimensional Euclidean space, we consider a closed convex polytope, P, defined by intersecting a number of half-spaces; each half-space is the area which lies on one side of a hyperplane. If the objective is to maximize a linear functional $L(x)$, consider the hyper-planes $H(c)$ defined by $L(x) = c$. As c increases, these form a parallel family. If the problem is well-posed, we wish to find the largest value of c such that $H(c)$ intersects P. If there is no solution to the LP problem, then there is no such largest value of c. In this case we can show that the optimum value of c is attained on the boundary of P.

As in the 2-dimensional Euclidean space, the simplex method starts with some vertex of the polytope. Each iteration an adjacent vertex is chosen, such that the value of the objective function does not decrease. If no such vertex exists, we have found a solution to the problem. However, usually, such an adjacent vertex is not unique, and a *pivot rule* must be specified to determine which vertex to pick. Various pivot rules exist.

The original formulation of the simplex method, formulated by Dantzig, visits all 2^n vertices before arriving at the optimal vertex, thus the worst-case complexity of the algorithm is exponential. It is an open question whether there is a pivot rule with polynomial time worst-case complexity.

Nevertheless, the simplex method is remarkably efficient in practice. Attempts to explain this employ the notion of average complexity or smoothed complexity. The optimization process is described in Figure 2.4.

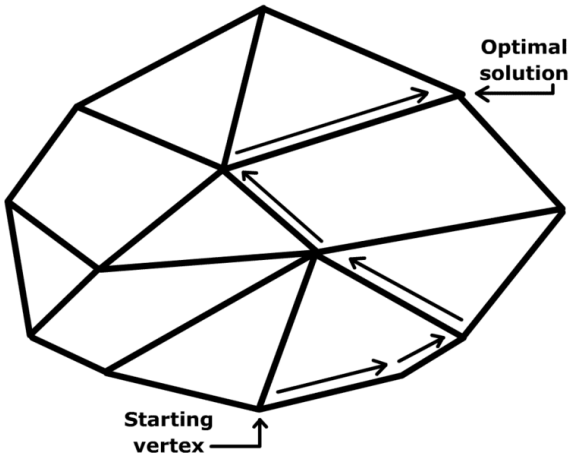

Figure 2.4: A series of linear inequalities defines a polytope as a feasible region. The simplex algorithm starts at some vertex and moves along the edges of the polytope until it reaches the vertex of the optimum solution (figure taken from wikipedia).

The Projective Method

In 1979, Leonid Khaciyan presented the *ellipsoid* method, guaranteed to solve any linear program in a number of steps which is a polynomial function of the size of the linear program. Consequently, the ellipsoid method is faster than the simplex method in contrived cases where the simplex method performs poorly. In practice, however, the simplex method is far superior to the ellipsoid method.

In 1984, Narendra Karmarkar introduced the projective method for linear programming also known as the *interior-point* method, or the *barrier* method. This method combines the desirable theoretical properties of the ellipsoid method and practical advantages of the simplex method. Its success initiated an explosion in the development of interior-point methods. These methods, in contrast to the simplex method, do not pass from vertex to vertex, but pass only through the interior of the feasible region.

2.2.2 Integer Programming

Integer programming (IP) is an expressive framework for modeling and solving discrete optimization problems that arise in a variety of contexts in many fields of engineering, and also in other fields such as economy. Integer programming representations work with implicit algebraic constraints (linear equations and inequalities on integer valued variables) to capture the feasible set of alternatives, and linear objective functions (to minimize or maximize over the feasible set) that specify the criterion for defining optimality. When an optimization problem consists of both linear and integer valued variables it is called *Mixed Integer Programming* (*MIP*) optimization problem. This algebraic approach permits certain natural extensions of the powerful methodologies of linear programming to be brought to bear on combinatorial optimization and on fundamental algorithmic questions.

Branch-and-Bound Technique

Integer programming problems are solved using the *branch-and-bound* technique. The branch step is done by fixing the value of a certain variable to one of the possible values. This results in a *solution tree*. The fixed variable at each node in the tree is called the *branching variable*. Note that for n binary variables, the tree size is 2^n, so, we have to narrow the group of possible solutions in order to solve the problem efficiently. At this point the bound step comes to rescue.

When branching the solution tree, all the variables from the root to the current node are fixed. The optimal solution for this integer-programming problem, given the fixed variables, is smaller or equal to the optimal solution without the requirement that the variables are integers (the relaxed linear-programming problem). Thus, at each node the appropriate linear-programming problem is solved, and the optimal value of the objective function serves as an upper bound for the current node. If we have already found another solution that exceeds the bound, then the current node can be marked as a leaf, and no branching is done on that node.

Acceleration Methods

Although the branch-and-bound technique can solve integer-programming problem effectively, the worst case is still exponential. Thus, a few techniques were developed to reduce the space of feasible solutions, in order to improve the performance of the solving process.

One such method works in a preprocessing manner, i.e., before the problem is optimized. In this method we try to fix as much variables as possible, according to the different constraints. Moreover, we can also remove constraints if they are already satisfied in all the possible solutions. Changing the coefficients of the different constraints is also possible, thus reducing the bound of the relaxed linear-programming problem.

Examples (for binary variables):

- $3x_1 + 2x_2 \leq 2 \Rightarrow x_1 = 0$.
- $3x_1 + 2x_2 \leq 6$ - This constraint can be removed, because every binary value for the variables satisfies this constraint.
- $4x_1 + 5x_2 + x_3 \leq 2 \Rightarrow 2x_1 + 2x_2 + x_3 \leq 2$.

Another method is using *cutting planes*. A cutting plane for an integer programming problem is a new constraint that eliminates some possible solutions for the relaxed linear programming, without eliminating any possible solution for the integer programming problem. There are several algorithms for achieving those cuts, such as cover cuts, clique cuts, disjunctive cuts, flow cover cuts, gomory fractional cuts, generalized upper bound (GUB) cover cuts, implied bound cuts, mixed integer rounding cuts and flow path cuts. For information on these algorithms see [1, 5].

Chapter 3

Problem Definition and Previous Work

The problem we tackle in this work is explaining cause-effect pairs of a knocked-out gene (cause) and the gene affected by it (effect) using a network of physical interactions. In this chapter we will formally define the problem and describe a previous approach for solving it by Yeang *et al.* [33, 34].

3.1 Physical Network Model

Yeang *et al.* first introduced their physical network model in [33]. Their model consists of protein-DNA and protein-protein interaction networks. These networks are merged into a *skeleton graph* – a directed graph $G = (V, E)$, where V contains the genes or their protein product, while $E = E_P \cup E_R$ contains the possible protein-protein (E_P) or protein-DNA (E_R) interactions. While the direction of a protein-DNA edge is determined a priori, the direction of a protein-protein edge is initially undetermined.

3.1.1 Basic Definitions

The skeleton graph provides only a template of possible pairwise molecular bindings. In order to make predictions about regulatory effects, they define the following core variables:

- $X_E = \{x_e | e \in E\}$ a collection of binary variables denoting the presence or absence of interactions (both protein-protein and protein-DNA interactions).
- $S_E = \{s_e | e \in E\}$ a collection of binary variables denoting with activation (0) or repression (1). They are referred to as the *signs* of the edges.
- $D_E = \{d_e | e \in E\}$ a collection of binary variables denoting the direction of the edges. The protein-DNA are directed a-priori, thus those variables are relevant to PPIs only.

3.1.2 Model Assumptions

Yeang *et al.* make the following assumptions:

1. The primary causal mechanisms underlying gene regulation considered in the model are cascades of molecular interactions. Other mechanisms such as chromatin modification, alternative splicing, or signal transduction via small molecules are overlooked in the model.
2. The effect of gene deletions propagates along the molecular cascades represented in the model. Combinatorial interactions such as complex formation, parallel pathways, or competitive binding, are not considered.
3. Each protein-protein interaction possesses a unique direction in the pathways that it participates in.
4. All the data are assumed to be collected under normal growth conditions. Moreover, they assume that the regulatory circuitry is not radically changed following a knock-out, at least not in terms of how the effects of the deletion are propagated.
5. The model only takes into account interactions that were detected in some experiment, i.e., it is limited to the skeleton graph, disregarding possible undetected interactions (false negatives).

3.1.3 Explanatory Pathways

The model defines certain basic rules for paths that *explain* knockout pairs. A *k-explanatory path* of a knockout pair (s, t) is an ordered set of vertices $\pi = (s = p_1, p_2 \dots p_{k+1} = t)$ that satisfies the following conditions:

1. π is a path: $\forall_{1 \leq i \leq k}(p_i, p_{i+1}) \in E$, and all the edges along the path exist: $\forall_{1 \leq i \leq k} x_{(p_i, p_{i+1})} = 1$.
2. π is a simple path: $\forall_{i \neq j}(p_i \neq p_j)$.
3. All the edges in π are in the forward direction (from s to t).
4. The last edge in the path is a protein-DNA edge: $(p_k, p_{k+1}) \in E_R$.
5. If an intermediate gene along the path has been knocked out then it should also exhibit an effect on the target protein: $\forall_{1 \leq i \leq k}(\exists_{p \in V}(p_i, p) \in X) \rightarrow ((p_i, p_{k+1}) \in X)$.

3.1.4 Data Association and Potentials

The modeling in the work of Yeang *et al.* was done using *factor graphs*. A factor graph is a bipartite graph (X, F) where X is a set of variables and F is a set of *factors*. A factor f_j is a function mapping a subset of variables $X_j \subseteq X$ to some range. This graph represents the factorization

$$g(A) = \prod_{j=1}^{m} f_j(A_j) \tag{3.1}$$

where A is an assignment to all the variables in X and A_j is the assignment of A to all the variables in X_j. Factor graphs are visualized as unidirected bipartite graphs, where the variables appear as circles, and the factors appear as squares. The edges reflect the dependencies between the variables and the factors. An example of such a factor graph appears in Figure 3.1.

Yeang *et al.* used this model, where the factors were assigned with potential functions for each of the variables. Those potential functions reflect the probability of the observed values for each variable. They then used the max-product algorithm in order to find the configuration that maximized the joint probability distribution (JPD) induced by the model.

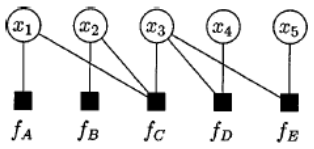

Figure 3.1: An example of a factor graph representing the joint distribution $f_A(x_1)f_B(x_2)f_C(x_1, x_2, x_3)f_D(x_3, x_4)f_D(x_3, x_5)$ (figure taken from [17]).

The different potentials and the algorithmic approach are described in detail later in this chapter.

Physical Interaction Data

The physical interaction data used in Yeang's model contained protein-protein and protein-DNA interactions. Each such interaction is assigned with a potential value. For example, for a binary variable x indicating the presence/absence of an interaction, the measurements pertaining to this variable are incorporated into the model according to

$$\phi(x) = \left[\frac{P(data|x=1)}{P(data|x=0)}\right]^x \tag{3.2}$$

Potential functions are combined by a product across independent observations.

The available protein-protein interactions data come from multiple sources, without a coherent overall error model. Thus, Yeang *et al.* defined an error model based on a subdivision made in DIP [26] to interactions verified in small-scale or multiple experiments, and used the empirical results of Deane *et al.* [6] to estimate the reliability of each putative interaction. Deane *et al.* performed two independent tests, ERP (Expression Profile Reliability) and PVM (Paralogous Verification Method), on DIP and its subsets to gauge the level and type of errors introduced. EPR examines the distribution of the Euclidean distances between expression profiles of protein pairs and uses this information to estimate the false positive rate of a set of protein pairs. PVM, on the other hand, tests whether paralogs of a protein pair also interact and gives corresponding confidence measures. Yeang *et al.* incorporated

both EPR and PVM to the potential function of the protein-protein interactions, to reflect the confidence they have in these interactions.

As for the protein-DNA interactions, the ChIP-chip experiments provide a p-value for each pair of transcription factor and a DNA promoter region. This confidence value reflects the relative abundance of the promoter DNA segments enriched by chIP, compared to unenriched segments. These p-values were transformed into potential functions.

Functional Data

The functional data, containing the differentially expressed genes from the gene-knockout experiments, and also associated with p-values. Yeang *et al.* defined by K_p the index set of significant knockout effects. The significant effects according to the error model are not assumed to be necessarily correct. Each pairwise effect is first tied to an unobserved variable that represents actual (as opposed to measured) knockout effect. The actual knockout effects are subsequently associated with variables along candidate pathways.

Specifically, the following variables were defined:

- $K = \{k_{ij}|(i,j) \in K_p\}$ is a collection of discrete variables of actual pairwise single knockout effects. The value is 1 for up-regulation, -1 for down-regulation and 0 for un-effected.
- $O = \{o_{ij}|(i,j) \in K_p\}$ denotes the relative measurements of gene-expression levels in the knockout experiments with respect to the wild type.

The actual knockout effect (k_{ij}) is tied to the measurement (o_{ij}) via a potential function ϕ_{ij} analogously to the protein-DNA and protein-protein interaction data:

$$\phi_{ij}(k_{ij}; o_{ij}) \propto \left[\frac{P(o_{ij}|k_{ij}=1)}{P(o_{ij}|k_{ij}=0)}\right] \tag{3.3}$$

where the likelihood ratios are derived from the available error model.

Consistent Pathways

The aim is to explain the knockouts using explanatory pathways. The identification of explanatory pathways is done in the pre-processing phase, because there is no need to annotate

the sign and direction of edges. In order for an explanatory pathway to be a *consistent explanatory pathway* the aggregate sign of the path must be consistent with the knockout effect. Each edge along a path has a sign variable, and the sign of the path is the aggregate effect, which is represented by the multiplication of the signs of all its edges, i.e., $\Pi_{e\in\pi} s_e = -k_{ij}$.

Yeang *et al.* defined $\Sigma = \{\sigma_{ija} | (i,j) \in K_p, \pi_a \in \Pi\}$ as a collection of binary variables, denoting whether an explanatory pathway is a consistent explanatory pathway, where Π is the collection of explanatory pathways. Given a path π_a, let E_a denote the set of edges in that path. In order to identify the explanatory pathways that are consistent with the knockout effect, the following potential function is defined:

$$\psi_{ija}(X_a, S_a, D_a, k_{ij}) = \prod_{e\in E_a} I(x_e = 1) I(\prod_{e\in E_a} s_e = -k_{ij}) \prod_{e\in E_a \cap E_P} I(d_e = \hat{d}_e) \tag{3.4}$$

where $I(\cdot)$ is a 0/1 indicator function. Note that the direction is relevant only for PPI, because the protein-DNA interactions are directed a-priori.

In case there are multiple explanatory pathways that connect the knockout pair, Yeang *et al.* require that at least one such path is consistent. Denote this set of paths by Π_{ij}. Recall that σ_{ija} denotes a selection variable for path π_a, i.e. $\sigma_{ija} = 1$ if π_a is used to consistently explain k_{ij}, and zero otherwise. Thus, the potential function in equation 3.4 augmented with variable σ_{ija} is:

$$\psi_{ija}(X_a, S_a, D_a, k_{ij}, \sigma_{ija}) = \epsilon_1 + (1-\epsilon_1) I(\sigma_{ija} = 1) \psi_{ija}(X_a, S_a, D_a, k_{ij}) \tag{3.5}$$

The potential function does not "vanish" even when the constraints are violated, in order to allow other causes that explain the knockout effect.

Next, Yeang *et al.* construct a potential function term ψ^{OR}_{ija} to specify the condition that at least one explanatory path is consistent. By using ϵ, this potential function is a "soft" logical OR:

$$\psi^{OR}_{ija}(\sigma_{ij1}, \ldots, \sigma_{ij|\Pi_{ij}|}) = \epsilon + (1-\epsilon)(1 - \prod_{a\in\Pi_{ij}} I(\sigma_{ija=0})) \tag{3.6}$$

Combining equations 3.5 and 3.6, the potential function associated with each knockout effect is:

$$\psi_{ij}(X_{ij}, S_{ij}, D_{ij}, \Sigma_{ij}, k_{ij}) = \psi^{OR}_{ija}(\sigma_{ij1}, \ldots, \sigma_{ij|\Pi_{ij}|}) \prod_{a\in\Pi_{ij}} \psi_{ija}(X_a, S_a, D_a, k_{ij}, \sigma_{ija}) \tag{3.7}$$

where X, S, D and Σ correspond to indicators for the existence of edges, their sign, their direction and whether they are consistent explanatory paths, respectively.

3.1.5 Algorithmic Approach

The potential functions mentioned above are combined into a joint distribution over the variables in the physical model so that the probability value reflects the degree of support that each possible physical model (annotated graph) has with respect to the available data. Specifically:

$$P(X_E, S_E, D_E, K, \Sigma | Y_E, O_K) \propto \prod_{e \in E} \phi(x_e | y_e) \prod_{(i,j) \in K_p} \phi_{ij}(k_{ij}; o_{ij}) \psi_{ij}(X_{ij}, S_{ij}, \Sigma_{ij}, k_{ij}) \quad (3.8)$$

Note that the potential functions are combined in a product form because of the assumption that individual measurements of interactions or knock-out effects are statistically independent. This may not be a realistic assumption; for example, readings of adjacent spots on microarrays may be correlated.

The goal is to find the most likely realization of the physical model, i.e., find the most likely settings of the model variables in the joint distribution (a maximum a posteriori (MAP) configuration). Since the joint distribution involves a large number of variables Yeang *et al.* settle for finding an approximate setting using distributed message passing algorithm such as *max-product* [17].

Denote by U the set of all the variables. In order to find a MAP configuration it is sufficient to evaluate the *max-marginals* for each variable $x \in U$, i.e., the maximum joint probability over the subset of possible configurations for all the other variables, given a fixed value of x. If the MAP configuration is unique then the max-marginals have unique maximizing arguments. Otherwise, there are multiple MAP configurations.

The max-product algorithm is a local propagation algorithm that evaluates the approximate value of the max-marginals for each variable, as follows: The algorithm determines, for each potential function what information needs to be shared between the variables in order to evaluate those max-marginals, under the assumption that the variables are independent in the absence of the potential function in question. All the operations of the algorithm are local in this sense. The algorithm is guaranteed to find the correct max-marginals only in case the factor graph has a tree-like structure.

As mentioned earlier, there may be more than one MAP configuration. In that case Yeang *et al.* apply a recursive procedure:

- At each iteration the max-product algorithm is applied given the fixed variables from

previous iterations.

- Variables are fixed if they yield a unique MAP configuration.
- If there are still undetermined variables, one is chosen randomly and fixed to a random value.

This procedure is applied until all the variables are set. In a relatively well constrained case, the recursive search is capable of identifying all the approximate MAP configurations, by building a decision tree, where internal nodes represent fixed values and the leaves represent the full configuration. However, enumerating all MAP configuration becomes intractable as the number of variables greatly exceeds the number of available constraints. In that case, Yeang *et al.* revise the recursive algorithm to decompose the networks into sub-networks, such that the variables in different subnetworks do not interfere given the constraints from the available data. Thus, the MAP configuration is expressed as a product of the sub-configurations in the different sub-networks.

Chapter 4

The *SPINE* Framework

This chapter gives a detailed description of SPINE, our framework for signaling-regulatory pathway inference from cause-effect experiments. First, we describe the input data. Then we provide a description of the characteristics of explanatory pathways and the optimization criterion employed. Finally, we describe the algorithmic approach. A summary of the notation in this chapter appears in Table 4.1.

4.1 Data Description

The input data consist of a directed network $G = (V, E)$ of physical interactions and a collection X of cause-effect pairs of genes. The latter is commonly obtained using knockout experiments in which a gene is perturbed (*cause*) and the expression levels of all other genes are monitored. Those genes that significantly change their expression levels in response to the knockout are said to be *affected* by it. All other genes are assumed to be non-affected. In the context of knockout experiments we refer to a cause-effect pair also as a *knockout pair*.

The nodes in the network represent genes and their protein products (no distinction is made between those two). The network has two types of edges: protein-DNA interactions and protein-protein interactions (PPIs). Protein-DNA interactions are naturally represented as edges directed from a transcription factor to a regulated gene. Each PPI is represented using two oppositely directed edges. In addition to a direction, every interaction in the network is

Notation	Definition
$G = (V, E)$	the graph on a set V of vertices and a set E of edges
E_P	the protein-protein interactions
E_R	the protein-DNA interactions
e	an edge in the graph
π	a path in the graph
Π	a collection of paths in the graph
$s(\pi)$	the source of path π
$t(\pi)$	the target of path π
P^i_Π	the collection of all sets $S \subseteq \Pi$ of cardinality i.
$p(object)$	the probability of an object (path, group of paths, etc.)
$sgn(e)$	the sign of an edge (0 - activation, 1 - repression)
$r(e)$	the reliability of an edge
X	collection of knockout pairs
$K_{s,t}$	boolean indicator for knockout $(s, t) \in X$ (1 - explained, 0 - not explained)
$e(s, t)$	complement of the knockout effect $(s, t) \in X$ (0 - activation, 1 - repression)
N_π	indicator for path π specifying whether π is an explanatory path (1 - explanatory, 0 - non-explanatory)
M_π	indicator for path π specifying whether π is a consistent explanatory path (1 - consistent, 0 - inconsistent)
M_Π	indicator for a set of explanatory paths Π (1 - all the paths are consistent, 0 - otherwise)

Table 4.1: Model notation.

assigned a positive *reliability* value, representing the probability that the interaction is true; and a binary *sign*, representing its effect (0-activation or 1-repression). In the following we denote the sign and reliability of each interaction $e \in E$ by $sgn(e)$ and $r(e)$, respectively. We denote the set of protein-DNA interactions by $E_R \subseteq E$. Finally, we denote by $e(s,t)$ the complement of the observed effect on gene t when knocking out gene s. $e(s,t)$ is assumed to represent the effect of s on t in wild-type.

4.2 Explanatory Pathways

For a path π, let $s(\pi)$ and $t(\pi)$ denote its source and target, respectively. Let N_π be an indicator variable denoting whether π explains the pair $(s(\pi), t(\pi))$. As mentioned earlier, the work of Yeang *et al.* [33] defined certain basic rules for paths that *explain* knockout pairs (chapter 2). An explanatory path π is said to be *consistent* if in addition:

1. The aggregate sign of π is equal to $e(s(\pi), t(\pi))$. Formally: $\oplus_{e\in\pi}(sgn(e)) = e(s(\pi), t(\pi))$, where $\oplus$ is the xor operator (addition modulo 2).

2. Every proper suffix γ of π that connects another knockout pair is a consistent explanatory pathway.

The first requirement appeared in the original definition in [33]. Here we also added the second requirement in order to prevent a scenario in which a long explanatory pathway is consistent, although it relies on an inconsistent explanatory pathway suffix. Note that we focus here on simple paths, as a protein is likely to play a single role in a given pathway. Further note that the consistency requirement, imposed here on suffixes, is adequate only for sub-paths that end with a protein-DNA edge, as the effect is observed at the transcription level.

These definitions can be easily generalized to accommodate for already known edge signs. For ease of presentation, we concentrate in the following on the case where all edge signs are unknown. In particular, if we denote by M_π an indicator variable specifying whether an explanatory path π is consistent, then

$$M_\pi \equiv (\oplus_{e\in\pi} sgn(e) = e(s(\pi), t(\pi)) \wedge (\forall_{\gamma\in \text{suffix}(\pi)}(N_\gamma \rightarrow M_\gamma)) \tag{4.1}$$

Yeang *et al.* [33] considered in their work the inference of edge signs only. Another possibility is to assign signs to nodes rather than edges. In this case, all the edges emanating from a node carry its sign. This may be less accurate but would yield many fewer variables and hence avoid overfitting problems and computational bottlenecks. Indeed, most proteins in current databases are classified as either activators or repressors solely, so this relaxation seems to be in line with current biological knowledge. In the rest of the paper we focus on this latter variant, and use the edge variant mainly for comparison purposes.

4.3 Optimization Criterion

Intuitively, the goal of the algorithm is to infer regulatory pathways in the network that provide a consistent explanation for the input set of knockout pairs. In practice, the pathways are dependent and there could be more than one pathway explaining a given pair. Hence, rather than trying to pinpoint a single pathway per pair, we aim at assigning signs to the interactions in the network so that the data can be best explained. There are several definitions to what a "best" explanation is. A parsimonious definition would seek a sign assignment so that a maximum number of pairs have at least one consistent path. Note that an implicit assumption here is that the effect propagates through a single pathway, and the existence of other, possibly inconsistent pathways does not contradict the existence of the effect. However, the parsimonious criterion ignores the information on edge reliabilities and could, e.g., prefer solutions in which many pairs are explained by very low probability paths. Thus, we define our optimization problem as that of finding an assignment that maximizes the expected number of pairs that have at least one consistent path. We give a formal definition of this criterion below.

Expectation calculation

Define the probability of a path as the product of the reliabilities of its edges:

$$p(\pi) = \prod_{e \in \pi} r(e) \tag{4.2}$$

We say that a knockout pair (s, t) is *explained* or *satisfied* if there exists at least one explanatory path that is consistent with it. We associate an indicator variable $K_{s,t}$ with this

event.

We wish to find a setting of the node or edge signs that maximizes the expected number of satisfied knockout pairs, which is given by:

$$\begin{aligned} E(\textstyle\sum_{(s,t)\in X} K_{s,t}) &= \textstyle\sum_{(s,t)\in X} E(K_{s,t}) \qquad = \\ &= \textstyle\sum_{(s,t)\in X} p(K_{s,t}=1) \end{aligned} \tag{4.3}$$

Given a collection Π of explanatory paths for a knockout pair (s,t), we denote by M_Π an indicator variable specifying whether all paths in Π are consistent with this pair. Clearly, $M_\Pi \equiv \wedge_{\pi\in\Pi} M_\pi$. The probability that all the paths exist, $p(\Pi)$, is equal to the probability that all the edges in Π exist. Denoting the latter set of edges by E_Π we have:

$$p(\Pi) = \prod_{e\in E_\Pi} r(e) \tag{4.4}$$

For a knockout pair $(s,t) \in X$, denote by Π the collection of explanatory paths for it, with $|\Pi| = n$. To compute the probability that at least one of the paths exist, we must take into account the dependencies among them due to edge intersections. Let P^i_Π denote the collection of all sets $S \subseteq \Pi$ of cardinality i. Using the inclusion-exclusion principle, the probability that at least one of the paths in Π is consistent is:

$$p(K_{s,t}=1) = \sum_{1\le i\le n} (-1)^{i-1} \sum_{\Gamma\in P^i_\Pi} p(\Gamma) M_\Gamma \tag{4.5}$$

As mentioned earlier, in the objective function proposed by Yeang *et al.* the components are multiplied, although they are dependent on one another. Here, path dependencies are explicitly treated by the model (Equation 4.5). Notably, there are two additional differences, not mentioned above, between our model and that of Yeang *et al.*: (i) No requirement is placed in [33] on the consistency of suffixes of an explanatory path. (ii) SPINE models PPI edges as potentially bi-directional, while in [33] they are forced to have a single direction. As we shall see below, in practice, most of the edges attain a single direction also under our model.

4.4 Algorithmic Approach

We tackle the optimization problem using an integer program reformulation of it, to which we apply the commercial $CPLEX^{TM}$ solver. Below we give the integer programming for-

mulation and the full SPINE algorithm. A diagram that describes the SPINE's input, main phases and output, appears in Figure 4.1.

4.4.1 Integer-Programming Formulation

We reformulate the network annotation problem as an integer linear program. The objective function is simply $\sum_{(s,t)\in X} p(K_{s,t} = 1)$. The constraints are of two types:

1. Path constraints: determining the consistency of explanatory paths.

2. Knockout pair constraints: determining the expectation value of knockout pairs.

To obtain a linear program we convert the Boolean equations defining the variables of the model, as well as the objective function, into linear forms.

Denote by q_π the longest suffix of π which is an explanatory path. We express M_π linearly using two constraints as follows:

1. Consistent path sign:

$$M_\pi \leq \sum_{e\in\pi} sgn(e) - 2d_\pi + 1 - e(s(\pi), t(\pi)) \leq 1 \tag{4.6}$$

   where $0 \leq d_\pi \leq \lceil \frac{|\pi|}{2} \rceil$ is a dummy variable, used to express the parity of the repressor signs.

2. Consistent suffix:

$$M_\pi \leq M_{q_\pi} \tag{4.7}$$

   Note that this constraint is used only in case there is indeed a suffix which is an explanatory path.

As for Equation 4.5, the challenge is to express M_Γ, an indicator for a set of paths, in a linear form. This can be done using the following constraint:

$$0 \leq \sum_{\gamma\in\Gamma} M_\gamma - |\Gamma| M_\Gamma < |\Gamma| \tag{4.8}$$

4.4.2 Confidence Assignment

SPINE tries to optimize the expected number of explained knockout pairs. For a given instance of the problem, let O_{max} denote the optimum value of the objective function. Clearly, there may be more than one configuration of protein signs that attains this value. Hence, for each protein v we provide a *confidence value* C_v measuring how confident we are in its sign assignment. This value is calculated by running SPINE on a modified instance, in which the sign of the protein is flipped, and recording the difference between the new value of the objective function and O_{max}. We declare v to be *confident* if $C_v > \epsilon$, for a pre-specified $\epsilon \geq 0$. We consider a knockout pair to be *confidently explained* if there is a consistent explanatory path for it whose proteins are all confident.

Yeang *et al.* [33] did not define a confidence measure. However, they distinguished between variable assignments that uniquely maximize the objective function and those that do not. In their model, once all the unique assignments (e.g., of signs) have been determined, one variable is fixed arbitrarily, and the inference procedure is iteratively applied until all the variables are determined. Hence, in contrast to SPINE, some of the inferred assignments may be arbitrary.

4.5 A Speedup Heuristic

To deal with large integer programs, we infer the model variables using multi-level runs. At each level $1 \leq l \leq k$ we start from the confident variables inferred on previous levels, and search for a setting of the remaining variables of the model that maximizes the expected number of explained knockout pairs using explanatory paths of length $\leq l$. This heuristic is based on the assumption that short, confident pathways are more likely than long ones.

We use $\epsilon = 0$ and consider only confident predictions, as non-confident predictions can be arbitrarily flipped without changing the overall value of the objective function. The full SPINE algorithm is summarized in Figure 4.2. Note that this algorithm solves both the node- and the edge-variant of the problem.

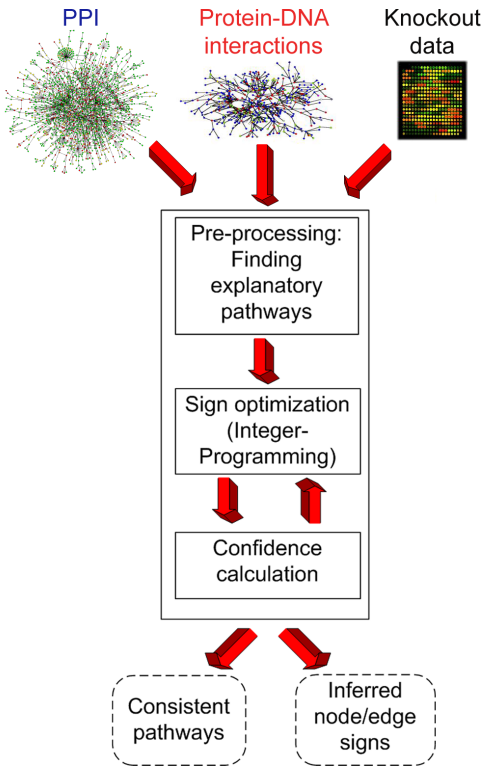

Figure 4.1: The SPINE computational pipeline. Top: the input, consisting of a PPI network, a protein-DNA network and gene-expression data from knockout experiments. Middle: the main analysis stages. Bottom: the output, consisting of inferred signs for proteins/interactions and consistent explanatory pathways induced by them.

$\underline{SPINE(G, X, k)}$
For i in $1 \ldots k$ **loop**
    $(solution, O_{max}) = Optimize(G, X, i)$
    $C =$ MeasureConfidence$(solution, O_{max})$
    Foreach variable $v \in solution$ **loop**
        If $C_v > \epsilon$ **then**
            Update the sign of v in G
        End if
    End loop
End loop

$\underline{Optimize(G, X, k)}$
Find explanatory paths of length $\leq k$
Foreach unexplained knockout pair $x \in X$ **loop**
    Foreach explanatory path π for x **loop**
        Construct linear equations for π
    End loop
    Construct linear equations for the constraints of x
    Construct linear equations that calculate $p(K_x = 1)$
End loop
Maximize the expected number of explained knockout pairs subject to the constraints
Return the optimal solution and its value

$\underline{\text{MeasureConfidence}(Solution, O_{max})}$
Foreach variable $v \in Solution$ **loop**
    $O_v =$ maximum objective given $sgn(v) = (1 - sgn(v))$
    $C_v = O_{max}$ - O_v
End loop

Figure 4.2: The SPINE algorithm. Top: the main procedure; middle: pre-processing and optimization; bottom: confidence computation.

4.6 A Toy Example

Figure 4.3(a) gives an example of a simple input network. The nodes $\{A, B, C, D, E, F, G\}$ represent proteins, the solid edges represent protein-DNA interactions (for convenience, no PPIs are used), and the dashed edges represent knockout pairs along with the required knockout effect. The reliability of each edge appears to its side.

The explanatory paths in this example (with their probabilities in parentheses) are:

- $\pi_1 = A \rightarrow B \rightarrow C \rightarrow D$ (0.9).
- $\pi_2 = B \rightarrow C \rightarrow D$ (1).
- $\pi_3 = A \rightarrow G \rightarrow D$ (0.45).
- $\pi_4 = A \rightarrow E \rightarrow F$ (0.8).

The knockout pair indicators are:

$$\begin{aligned} K_{A,D} &\equiv (M_{\pi_1} \vee M_{\pi_3}) \\ K_{B,D} &\equiv M_{\pi_2} \\ K_{A,F} &\equiv M_{\pi_4} \end{aligned}$$

Using Equation 4.3 we get the total expectation:

$$\begin{aligned} E(\sum_{(s,t)\in X} K_{s,t}) &= \sum_{(s,t)\in X} p(K_{s,t} = 1) = \\ &= p(K_{A,D} = 1) + p(K_{B,D} = 1) + p(K_{A,F} = 1) = \\ &= p(\pi_1)M_{\pi_1} + p(\pi_3)M_{\pi_3} - p(\{\pi_1, \pi_3\})M_{\{\pi_1,\pi_3\}} + p(\pi_2)M_{\pi_2} + p(\pi_4)M_{\pi_4} \end{aligned}$$

When looking for an assignment of signs to edges, there are many possible configurations that satisfy all those paths. In all those configurations $A \rightarrow B$ must have a positive sign, since both knockout pairs (A, D) and (B, D) have the same effect. All the other edges remain unsigned, since there is more than one possibility to set their signs. Hence, there is a single confident edge in this variant, and no consistent path is confidently inferred (Figure 4.3(b)).

In contrast, in the node-based variant, many more variables are determined (Figure 4.3(c)). Indeed, A will be signed positive for the same argument as above. Thus, E has

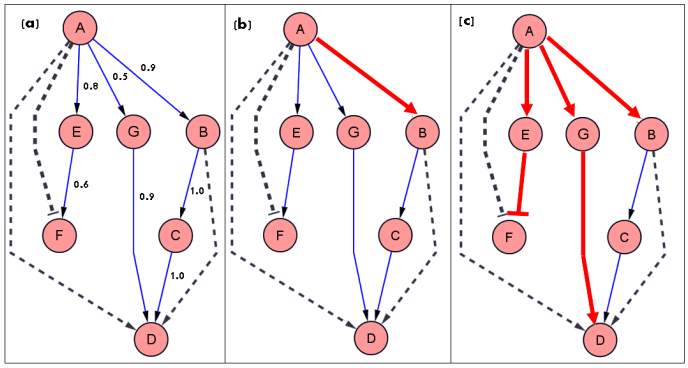

Figure 4.3: A toy example. Nodes represent proteins and edges represent protein-DNA interactions. Knockout pairs are denoted by dashed lines, with the desired effect denoted by the arrow type: regular (positive) or cut (negative). Edges whose sign has been inferred by the algorithm are bolded and colored red and those that were not inferred are non-bolded and colored blue. (a) The input network, including edge reliabilities. (b) The inferred confident network in the edge variant. (c) The inferred confident network in the node variant.

to be signed negative, and G has to be signed positive. The only unsigned nodes in this example are B and C: signing both of them as either positive or negative will make π_2 a consistent path.

Chapter 5

Experimental Results

This chapter describes the experimental results of SPINE. We tested SPINE both in a small-scale and a large-scale setting on a yeast (*Saccharomyces cerevisiae*) data set. First, we applied SPINE to part of a yeast subnetwork involved in mating in order to test the different variants of the algorithm and for purposes of comparison to a previous analysis made in [33]. Next, we applied SPINE to genome-wide yeast data and evaluated its prediction performance.

5.1 Data Acquisition

We constructed an integrated yeast network of 15,147 protein-protein and 5,568 protein-DNA interactions, spanning a total of 5,313 genes/proteins. Protein-protein interactions were taken from DIP ([26], April 2005 download) and were assigned confidence scores using a previously described logistic regression method [27]. Protein-DNA interactions were taken from [20] and were assigned a confidence of 0.96 based on the false positive rate estimations in [18].

Gene expression data were taken from [13] and included genome-wide gene expression measurements in yeast under 210 different single-gene knockouts. A previous analysis by Yeang *et al.* [33] yielded a collection X of 24,457 pairs of a knocked out gene and an affected gene, all with p-value ≤ 0.02.

Method	Left out	#Trials	Correct	Incorrect	Undecided	Accuracy
Yeang *et al.*	1	103	97.1%	2.9%	0%	**97%**
SPINE - edge variant	1	103	98%	1%	1%	**99%**
Yeang *et al.*	5	200	96.5%	3.5%	0%	**96.5%**
SPINE - edge variant	5	200	96.9%	0.4%	2.7%	**99%**
SPINE - node variant	1	103	37.86%	62.14%	0%	**37.86%**

Table 5.1: Cross validation results for the yeast mating subnetwork. A hidden knockout pair is considered to be successfully predicted if all explanatory paths for it are consistent with its true effect. The accuracy represents the percentage of correct predictions among predictions made. In the leave-five-out cross-validation, the rates were computed as in [33] by dividing the number of correct, incorrect or undecided held-out pairs by the total number of held-out pairs in all trials.

5.2 Application to a Mating subnetwork

Yeang *et al.* [33] constructed a yeast subnetwork involved in mating, which consists of 33 protein-DNA interactions, and 25 protein-protein interactions, covering a set of 149 knockout pairs. Applying our algorithm to this network with path length bound of 5 (as in [33]) resulted in inferring consistent explanatory paths for all 103 reachable knockout pairs.

To evaluate the performance of our algorithm we tested it in a cross-validation setting on this network, and compared our results to those attained by Yeang *et al.* [33]. For purposes of comparison, we first used the edge variant of our algorithm with the same evaluation criterion used in [33]. In detail, at each iteration one to five knockout pairs were hidden, and each was considered to be successfully predicted if all explanatory paths for it were consistent with its true effect. The results are shown in Table 5.1.

Evidently, the results are very similar to those of [33] with SPINE providing a slightly higher accuracy, both in the leave-one-out and leave-5-out tests. Notably, some of the knockout pairs remain undecided in the SPINE application since no explanatory path for them is predicted confidently. For SPINE, the only incorrect prediction was for knockout pair

Method variant	Maximization criterion	Left out	Correct	Incorrect	Undecided	Accuracy
Edge variant	Expectation	1	98%	1%	1%	**99%**
Edge variant	Number	1	47.6%	0%	52.4%	**100%**
Node variant	Expectation	1	89.3%	10.7%	0%	**89.3%**
Node variant	Number	1	60.2%	9.7%	30.1%	**86.1%**
Edge variant	Expectation	1 (noisy)	93.5%	3.9%	2.6%	**96%**
Node variant	Expectation	1 (noisy)	88.3%	11.7%	0%	**88.3%**

Table 5.2: Performance in cross-validation on the yeast mating subnetwork. A hidden knockout pair is considered to be successfully predicted if the expected number of explanatory paths with consistent signs is higher than the expected number of explanatory pathways with non-consistent signs. The accuracy represents the percentage of correct predictions among all predictions made.

(FUS3, PRY2), which is the only knockout pair involving FUS3. The only explanatory path from FUS3 to PRY2 is FUS3 $\rightarrow$ STE12 $\rightarrow$ PRY2; both interactions were signed as positive, yielding a positive path, in contrast to the needed knockout effect. Interestingly, the interaction FUS3 $\rightarrow$ STE12 is known to be positive, as FUS3 is known to activate STE12.

Next, we wished to test the effect of maximizing the expected number of satisfied knockout pairs, rather than simply their number. In these tests we used a more relaxed criterion for computing prediction accuracy: a knockout pair was considered to be predicted successfully if the expected number of explanatory paths that are consistent with its true effect exceeded the expected number of the inconsistent ones (if the numbers are equal we consider the pair as undecided). Using this criterion seems more natural under our model and indeed improved the accuracy for the node variant, although the results for the edge variant remained the same, as summarized in Table 5.1. Moreover, we tested the two alternative maximization criteria in both the edge- and node-variant. As shown in Table 5.2, maximizing the mere number of satisfied pairs resulted in markedly smaller numbers of confident sign variables in both variants, although both criteria attained similar accuracy levels.

By comparing the cross validation results for SPINE's edge and node variants it is evident

that the latter attains similar accuracy levels although it has significantly less variables to be optimized and, hence, is less prone to overfitting.

Finally, we conducted a noisy cross-validation test. In each trial, the sign of one knockout pair was flipped, and the sign of a second knockout pair was hidden and inferred. The results show that the model is robust to these perturbations and the accuracy is maintained at very high levels (Table 5.2).

5.3 Application to a Genome-Wide Network

Next, we applied our method to the genome-wide data. Due to the size of the network, and following Yeang *et al.* [33], we restricted the computation to paths of length at most 3. Out of 231 proteins participating in such paths, 183 protein signs were confidently inferred. Among 974 knockout pairs that had explanatory paths, 861 were confidently explained using the inferred signs. The contribution of each level in SPINE's execution is shown in Table 5.3. The amount of explained knockout pairs and inferred proteins increases in each level, and this demonstrates the importance of looking at paths rather than considering direct edges only. In comparison, when using only protein-DNA interactions the explanatory power of the model is markedly decreased: only 188 knockout pairs are confidently explained and only 30 protein signs are inferred.

In order to validate our predictions we gathered information on activator and repressor proteins from four different sources: GO ([3], December 2006 download) – molecular functions "transcriptional activator activity" and "transcriptional repressor activity"; KEGG [15]; Guelzim *et al.* [11]; and Myers *et al.* [23]. We excluded 19 genes that appeared as both activators and repressors in the different sources. The results are summarized in Table 5.4, showing a significant overlap between the model's prediction and the known signs. Examining the predictions made in each level of the algorithm we observe that most predictions, and also most errors, are made in level 3: 24/27 activators and 8/15 repressors are correctly predicted. The statistics of these predictions are shown in Table 5.4.

The confidence measure can vary from one inferred sign to another. Thus, we checked the accuracy of our predictions when using varying thresholds of the confidence measure for determining the set of predicted signs. The inferred signs were grouped into bins, where each bin contains the inferred signs above a certain confidence threshold. The results clearly

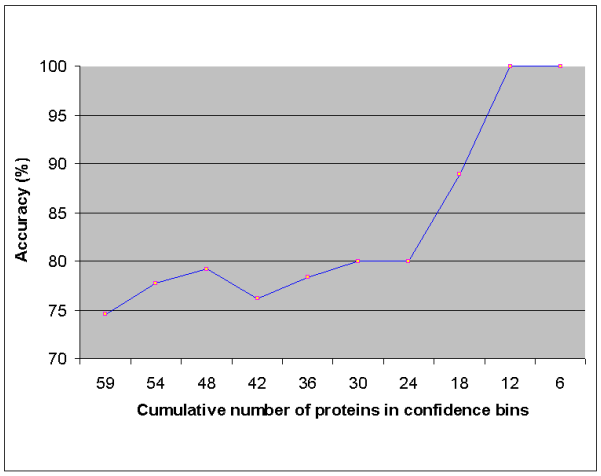

Figure 5.1: The prediction accuracy as a function of the confidence threshold. The cumulative number of proteins in each bin appears on the x-axis, and the accuracy percentage appears on the y-axis. The confidence bins cover the threshold range between 0.01 and 34.56.

show that increasing the confidence threshold also increases the accuracy of the predictions (Figure 5.1).

Next, we wished to compare our prediction performance to that of [34]. As we could not readily apply the latter method to our network, nor to the intersection of our network and that in [34], we made two alternative comparisons: (i) comparing SPINE's performance on our network to the results reported in [34]; and (ii) applying SPINE to the network of [34]. Both comparisons were problematic since in (i) the networks differ in about 20% of their constituent interactions, although their sizes are about the same, and in (ii) we only had the reliabilities of about 80% of the interactions in the network of [34], and the other 20% were arbitrarily set to the average reliability value. Thus, the results should be interpreted with caution

Recall that the method of Yeang *et al.* infers edge signs. To infer node signs, we applied a majority rule to the set of edge signs incident to each node (nodes for which no majority existed remained undecided). The statistics of these predictions are shown in Table 5.4. It can be seen that the overall performance of the algorithms is similar in the activator

Level	#Explained knockouts	#Inferred signs
1	107	14
2	183	30
3	861	183

Table 5.3: Cumulative contributions of different path lengths (levels) to the inference process.

prediction, but SPINE attains higher accuracy in repressor prediction in the first comparison, while the results of Yeang *et al.* are not significant. To further evaluate the results of both algorithms we computed their precision and recall. *Precision* is the percent of correct predictions out of all predictions (for both activators and repressors); *recall* is the percent of correct predictions out of all known signs. For the first comparison, the precision and recall of SPINE were 75% and 24%, respectively. The method of Yeang *et al.* yielded lower rates in both measures with 68% precision and 20% recall. In the second comparison, SPINE attained 70% precision and 22% recall.

In contrast to the method of Yeang *et al.*, SPINE does not constrain the directions of the protein-protein interactions. Interestingly, most (312 out of 340) protein-protein interactions appeared in one direction only in the inferred consistent pathways.

Figure 5.2 shows an example subnetwork inferred by the model, including both molecular and functional data. The explanatory pathways that appear in the subnetwork go through GAL4. GAL4 is a known activator. It is a DNA-binding transcription factor required for the activation of the GAL genes in response to galactose. Other components in this subnetwork are MED2, GAL11, SRB4, SRB5 and SRB6, which are mediators that function as a bridge between the regulatory proteins and the basal polymerase II transcription machinery [23]. SPINE found a confident optimal solution, $S_{optimal}$, in which GAL4 is a confident activator, i.e. signing GAL4 as repressor reduces the expected number of explained knockout pairs. We checked the effect of signing GAL4 as a repressor by running SPINE again while fixing GAL4 as a repressor. Once an optimum was reached, we searched for a solution with the same optimal value, but with minimum changes compared to solution $S_{optimal}$. As shown in Figure 5.2(a), all the explanatory pathways that go through GAL4 are consistent when it is an activator, whereas when GAL4 is a repressor some of those pathways become inconsistent. Some genes, such as HAP5 maintain the same sign even when GAL4 is a repressor, due to

Type	Network	#Known	#Predicted	#Correct	Significance
Yeang *et al.* - Activators	The network in [34]	119	31	28	0.002
Yeang *et al.* - Repressors	The network in [34]	57	22	8	0.4
SPINE - Activators	The network in [34]	119	34	31	$5{\cdot}10^{-4}$
SPINE - Repressors	The network in [34]	57	22	8	0.4
SPINE - Activators	Our network	120	37	32	0.003
SPINE - Repressors	Our network	60	22	12	0.02

Table 5.4: Results of the comparison to known regulators, for both Yeang *et al.* and SPINE. Shown are the number of known activators and repressors that appear in the network, the number of sign predictions on this known set, the number of correct predictions and a hypergeometric p-value. The latter was computed separately for the activator and repressor predictions. The first two results are for running both Yeang *et al.* and SPINE for the network that appears in [34], and the third one is of the results of running SPINE on our network.

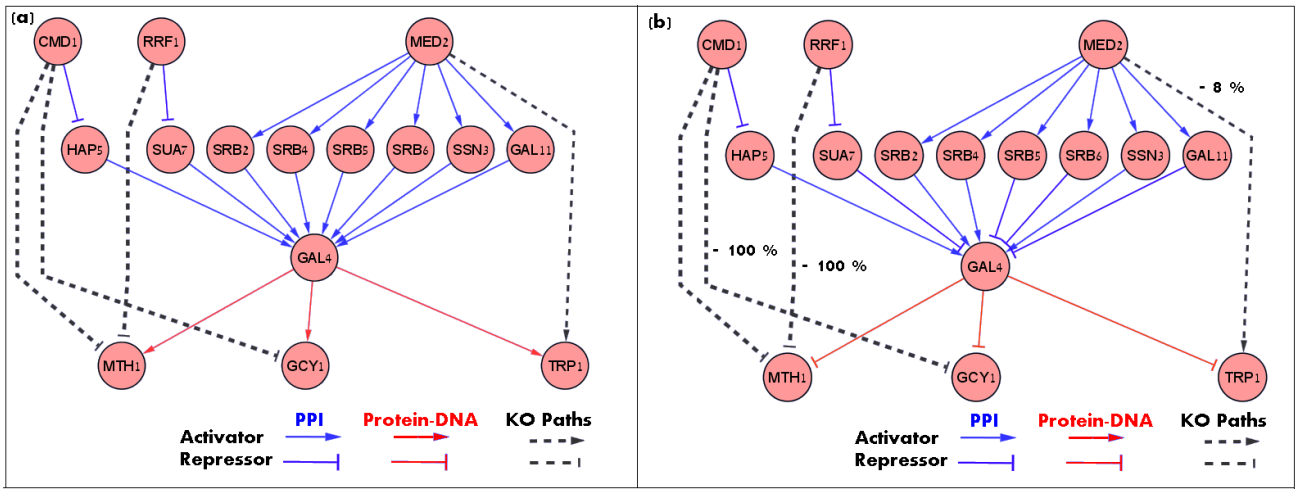

Figure 5.2: A GAL4-centered subnetwork. (a) GAL4 is correctly identified as an activator and all the depicted explanatory pathways are consistent. (b) A sign assignment in which GAL4 is forced to be a repressor, which is as close as possible to that chosen in (a). It yields several inconsistent pathways, implying a decrease in expectation. The decrease in the expectation percentage for each changed knockout pair appears alongside its edge.

their role in the different pathways, including pathways that are not part of the subnetwork. However, some genes do change their sign, such as SUA7. Figure 5.2(b) shows the decrease in the expectation of connected knockout pairs in the network.

Chapter 6

Conclusions and Future Work

This thesis addresses the problem of inferring signaling-regulatory pathways explaining cause-effect experiments. We have translated the reconstruction problem into that of assigning an activation/repression attribute with each protein so as to explain (in expectation) a maximum number of the knockout effects observed. We provided an integer programming formulation for the latter problem and solved it using a commercial solver. The resulting framework, SPINE, was tested on both small- and large-scale networks, and provided highly accurate results, comparing favorably to a previous approach by Yeang *et al.* [33].

There are several important contributions of the current work: (i) a basic optimization criterion and a biologically-motivated scheme to solve it in practice via multi-level runs; (ii) a confidence measure for the signs inferred by the model; and (iii) a general approach for assigning both edge and node signs, providing a balance between the flexibility of the assignment and its confidence.

While our algorithm provides a valuable framework for analyzing gene expression data in the context of a physical network, several of its limitations should be acknowledged. First, the algorithm is computationally intensive due to the resulting large integer programs. A practical approach to analyze larger networks could be to limit the accuracy of the solver, or to use filtration strategies in a pre-processing phase. For example, we can filter pathways that are not enriched with a specific biological process annotation, using GO, or to filter pathways with probability below some threshold. Second, it could be beneficial to integrate into the framework information on protein complexes [9, 12], which could be perhaps viewed

as unified single nodes in the context of the current analysis. Finally, our algorithm does not consider the strength of the knockout effect (and the associated p-value). A refined optimization criterion that combines this information could improve the accuracy of the predictions.

We have tested our method on cause-effect data gathered from gene knockout experiments. Although gene knockouts are routine in model organisms such as yeast, they are more technically involved in mammalian species such as mouse. For these higher eukaryotes, other ways of genetically perturbing the cell are gaining in practice, such as RNA interference and eQTL analysis. In addition, although DNA microarrays are currently the most widespread technology for measuring global changes in cellular state, a variety of other measurement systems, such as mass spectrometry for monitoring protein abundance, protein post-translational modification, or small molecule abundance, are gaining in popularity. The general formulation of our model allows its application to those types of cause-effect data as well. Moreover, our model can be easily extended to support experiments in which there are multiple causes per an effect, as in, e.g, double knockout experiments.

Chapter 7

Implementation Issues

As mentioned earlier, SPINE is based in an integer-programming formulation of the problem. Thus, the amount of time it takes to reach an optimum solution tends to vary according to several parameters:

- The number of variables in the problem.
- The constraints of the problem - If the problem is well constrained then the number of possible solutions decreases. Every constraint added can reduce the level of freedom we give the solver, thus speeding up the solution process.
- The vendor and the version of the solver - The solver we used is CPLEX, version 7.5. Currently, there are newer versions of this solver (as to now, version 10.0 is the latest), and they contain different features and parameters that can speed the identification of an optimal solution. Moreover, a special feature enables the solver to start with input values for the variables that may not be a feasible solution, but it can help the solver to create a new solution based on those values. This can speed up the calculation of the confidence measure, because we can start from the optimal solution, and let CPLEX change it so the current measured variable is flipped. As for the vendor, there are other solvers, each with different parameters and approaches to solve MIP problems. Thus, different solvers will have different performance.
- CPLEX solver parameters: During the experiments held in the thesis I ran into some

useful parameters that can speed up the solution process: (1) probing - probing enables CPLEX to "learn" the problem before and while solving it. In this learning process it fixes some variables, and changes constraints, all in order to solve the problem faster. This parameter has 4 possible values: 0 - no probing. 1-3: different levels of probing, from minimum probing to maximum probing. Different parameters can be helpful for different problems. However, probing can cause performance degradation during the solution, but it usually converge to an optimum faster. (*set mip strategy probe X*). (2) Optimality vs. feasibility - in case we wish to find a solution and we do not care whether it is optimal or not, we can ask CPLEX to find a feasible solution. (*set mip emphasis X* - 0 for optimality and 1 for feasibility). (3) The gap from the maximum possible solution - in every iteration CPLEX solves the linear programming solution, and sets the optimum of that solution as a threshold for the maximum possible integer solution. By setting the mipgap parameter to higher values CPLEX will stop the solution process once the current solution is less than mipgap percent from the current possible optimum (*set mip tolerance mipgap X*).

A manual for CPLEX version 7.5 can be found at the web page in [1].

As noted earlier, the performance of SPINE is affected by the amount of variables in the optimization problem. Reducing this number can substantially improve the SPINE's performance. In future research one can consider filtering the data model before running SPINE, using, for example, one of the following criteria:

1. Filter pathways that are not enriched with a specific biological process annotation, using GO.

2. Filter pathways with probability below some threshold.

3. Filter edges whose reliability is below some threshold.

We have not used any filtration criterion in this work.

A summary of the running time of the solver on the two applications studied in this work appear in Table 7.1. Since integer programming is NP-Complete, the running time of the solver on larger networks with a large number of cause-effect pairs may grow substantially.

Network	#Interactions	#Cause-effect	Path length	#Variables	#Constraints	Time (sec)
Mating	60	103	5	600	400	5
Genomewide	3000	1000	3	6300	7700	1800

Table 7.1: The running time in seconds of the solver for the two applications. The table also shows the size of the networks, the number of cause-effect pairs, the maximal path length, the number of variables and the number of constraints of every application. The hardware used is an Intel Xeon server, with 3.06GHz CPU. The solver process was run on a single CPU.

Bibliography

[1] Ilog cplex 7.5 user's manual. Website. `http://karnak.upc.es/lceio/manuals/cplex75/doc/usermanccpp/onlinedoc/index.html`.

[2] M. Andrec, B.N. Kholodenko, R.M. Levy, and E.D. Sontag. Inference of signaling and gene regulatory networks by steady-state perturbation experiments: structure and accuracy. *Journal of Theoretical Biology*, 232(3):427–441, 2005.

[3] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25(1):25–29, May 2000.

[4] G. D. Bader and C. W. Hogue. Analyzing yeast protein-protein interaction data obtained from different sources. *Nat Biotech*, 20(10):991–997, October 2002.

[5] D. Bertsimas and Weismantel R. *Optimization Over Integers*. Dynamic Ideas, May 2005.

[6] C. M. Deane, L. Salwinski, I. Xenarios, and D. Eisenberg. Protein interactions: two methods for assessment of the reliability of high throughput observations. *Mol Cell Proteomics*, 1(5):349–356, May 2002.

[7] M. Deng, F. Sun, and T. Chen. Assessment of the reliability of protein-protein interactions and protein function prediction. *Pac Symp Biocomput*, pages 140–151, 2003.

[8] S. Fields and O. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–246, July 1989.

[9] A. C. Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars J. 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, March 2006.

[10] A. C. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A. M. Michon, C. M. Cruciat, M. Remor, C. Hofert, M. Schelder, M. Brajenovic, H. Ruffner, A. Merino, K. Klein, M. Hudak, D. Dickson, T. Rudi, V. Gnau, A. Bauch, S. Bastuck, B. Huhse, C. Leutwein, M. A. Heurtier, R. R. Copley, A. Edelmann, E. Querfurth, V. Rybin, G. Drewes, M. Raida, T. Bouwmeester, P. Bork, B. Seraphin, B. Kuster, G. Neubauer, and G. Superti-Furga. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, January 2002.

[11] N. Guelzim, S. Bottani, P. Bourgine, and F. Kepes. Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics*, 31:60–63, May 2002.

[12] J. Hollunder, A Beyer, and T Wilhelm. Identification and characterization of protein subcomplexes in yeast. *Proteomics*, 8(5):2082–2089, May 2005.

[13] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, and Y. D. He. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, July 2000.

[14] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc Natl Acad Sci U S A*, 98(8):4569–4574, April 2001.

[15] M. Kanehisa and S. Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, January 2000.

[16] Boris N. Kholodenko, Anatoly Kiyatkin, Frank J. Bruggeman, Eduardo Sontag, Hans V. Westerhoff, and Jan B. Hoek. Untangling the wires: A strategy to trace functional interactions in signaling and gene networks. *PNAS*, 99(20):12841–12846, October 2002.

[17] Kschischang, Frey, and Loeliger. Factor graphs and the sum-product algorithm. *IEEETIT: IEEE Transactions on Information Theory*, 47, 2001.

[18] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804, October 2002.

[19] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol*, 14(13):1675–1680, December 1996.

[20] K. D. Macisaac, T. Wang, B. D. Gordon, D. K. Gifford, G. D. Stormo, and E. Fraenkel. An improved map of conserved regulatory sites for saccharomyces cerevisiae. *BMC Bioinformatics*, 7(1):113, March 2006.

[21] M Mann, R.C. Hendrickson, and A. Pandey. Analysis of proteins and proteomes by mass spectrometry. *Annual Review of Biochemistry*, 70:437–473, October 2001.

[22] H.W. Mewes, K. Albermann, K. Heumann, S. Lieb, and F. Pfeiffer. Mips: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, 25(1):28–30, 1997.

[23] L. C. Myers and R. D. Kornberg. Mediator of transcriptional regulation. *Annual Review of Biochemistry*, 69:729–749, 2000.

[24] O. Ourfali, T. Shlomi, T. Ideker, E. Ruppin, and Sharan R. Spine: A framework for signaling-regulatory pathway inference from cause-effect experiments. *Bioinformatics*, 23(13):i359–i366, July 2007.

[25] B. Ren, F. Robert, J. J. Wyrick, O. Aparicio, E. G. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Hannett, E. Kanin, T. L. Volkert, C. J. Wilson, S. P. Bell, and R. A. Young. Genome-wide location and function of dna binding proteins. *Science*, 290(5500):2306–2309, December 2000.

[26] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg. The database of interacting proteins: 2004 update. *Nucleic Acids Research*, 32:D449, 2004.

[27] T. Shlomi, D. Segal, E. Ruppin, and R. Sharan. Qpath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, 7, 2006.

[28] S. Sobhanifar. The yeast two-hybrid assay: an exercise in experimental eloquence. *The science creative quarterly*, 2, 2003.

[29] S. Suthram, T. Shlomi, E. Ruppin, R. Sharan, and T. Ideker. A direct comparison of protein interaction confidence assignment schemes. *BMC Bioinformatics*, 7:360, July 2006.

[30] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–627, February 2000.

[31] C. T. Workman, C. H. Mak, S. Mccuine, J. B. Tagne, M. Agarwal, O. Ozier, T. J. Begley, L. D. Samson, and T. Ideker. A systems approach to mapping dna damage response pathways. *Science*, 312(5776):1054–1059, May 2006.

[32] N. Yalamanchili, D.E. Zak, B.A. Ogunnaike, J.S. Schwaber, A. Kriete, and B.N. Kholodenko. Quantifying gene network connectivity in silico: scalability and accuracy of a modular approach. *Systems Biology (Stevenage)*, 153(4):236–246, 2006.

[33] C. H. Yeang, T. Ideker, and T. Jaakkola. Physical network models. *Journal of Computational Biology*, 11(2/3):243–262, 2004.

[34] C. H. Yeang, H. C. Mak, S. McCuine, C. Workman, T. Jaakkola, and T. Ideker. Validation and refinement of gene-regulatory pathways on a network of physical interactions. *Genome Biology*, 6(7):R62, 2005.

[35] C. H. Yeang and M. Vingron. A joint model of regulatory and metabolic networks. *BMC Bioinformatics*, 7:332, July 2006.