Explaining Genetic Knock-Out Effects Using Cost-Based Abduction

Emad A. M. Andrews and **Anthony J. Bonner**
Department of Computer Science, University of Toronto, Canada
{emad, bonner}@cs.toronto.edu

Abstract

Cost-Based Abduction (CBA) is an AI model for reasoning under uncertainty. In CBA, evidence to be explained is treated as a goal which is true and must be proven. Each proof of the goal is viewed as a feasible explanation and has a cost equal to the sum of the costs of all hypotheses that are assumed to complete the proof. The aim is to find the Least Cost Proof. This paper uses CBA to develop a novel method for modeling Genetic Regulatory Networks (GRN) and explaining genetic knock-out effects. Constructing GRN using multiple data sources is a fundamental problem in computational biology. We show that CBA is a powerful formalism for modeling GRN that can easily and effectively integrate multiple sources of biological data. In this paper, we use three different biological data sources: Protein-DNA, Protein–Protein and gene knock-out data. Using this data, we first create an un-annotated graph; CBA then annotates the graph by assigning a sign and a direction to each edge. Our biological results are promising; however, this manuscript focuses on the mathematical modeling of the application. The advantages of CBA and its relation to Bayesian inference are also presented.

1 Introduction

Since the word gene was coined in 1909, it had been a common belief that the higher an organism's complexity, the more genes it has. However, genome sequencing has revealed that the entire human genome contains only 23,000 to 40,000 genes, which is close to the number of genes in some types of worm [Baldi and Hatfield, 2003]. It is now known that it is not the number of genes, but gene interaction and regulation that are the sources of organism complexity. As a result, Genetic Regulatory Networks (GRN) have become one of the most interesting and challenging problems in computational biology and is expected to be the center of attention for a few decades to come [Baldi and Hatfield, 2003].

Other mathematical modeling and approaches to GRN include discrete models like Boolean Networks [Kauffman, 1993], continuous models like ordinary, stochastic and qualitative differential equations [Baldi and Hatfield, 2003], probabilistic and graphical models including Bayesian Networks (BN) [Beer and Tavazole, 2004; Friedman, 2004; Huttenhower *et al.*, 2006] and factor graphs [Yeang *et al.*, 2004] , and rule-based models including Induction Logic Programming [Ong *et al.*, 2007], and Abduction Logic Programming [Papatheodorou, 2007; Ray and Kakas, 2006]

Despite the advantage of being able to integrate different sources of biological knowledge in a single, homogeneous knowledge base, the area of rule-based models has received little attention in computational biology in general and in GRN modeling in particular. This is because there is no direct way to integrate an objective or a cost function with the knowledge base to measure solution quality. This issue is addressed to some extent by probabilistic inductive logic programming, which combines probability and logic programming [Raedt *et al.*, 2008]. In addition, there is no clear mathematical correspondence between rule-based models and other machine-learning approaches, such as probabilistic methods and Neural Networks (NN).

In this paper, we propose using Cost-Based Abduction (CBA) to model GRN by explaining the effects of genetic knock-out experiments. Because CBA is a rule-based system, it integrates different data sources efficiently and easily. In addition, it provides an associated cost for each explanation of the data; this cost serves as an objective function that we want to minimize. Interestingly, CBA has a clear, mathematical correspondence to BN, as we show later. Our results show that CBA is a promising tool in the field of computational biology.

This paper is organized as follows: the rest of the introduction gives the necessary background on CBA, its relation to Bayesian inference and NN, and describes the data sources used in the paper. We then describe how to use the data sources to create an un-annotated, or a skeleton graph. Finally, we describe our algorithm for building a CBA system from the skeleton graph. In particular, we show how to create rules and assign costs in a way that naturally enforces the constraints of genetic pathways and explains the effects of genetic knock-outs.

1.1 Cost-Based Abduction

CBA was first introduced by Charniak *et al.* [Charniak and Shimony, 1990]. Formally, a CBA system is a 4-tuple

(H, R, c, G), where H is a set of hypotheses, or propositions, c is a function from H to a nonnegative real *c(h)* called the assumability cost of $h \in H$, R *i*s a set of rules of the form: $R{:}(p_{i_1} \wedge p_{i_2} ... \wedge p_{i_n}) \to p_{i_k}$ for all $p_{i_i},..,p_{i_n} \in H$, $p_{i_k} \in H$ and $G \subseteq H$ is the goal, or the evidence set [Abdelbar, 1998].

Our objective is finding the Least Cost Proof (LCP) for the goal. The cost of a proof is the sum of the costs of all the hypotheses that were assumed to complete the proof. Any given hypothesis $p_i \in H$ can be true either by proving it or by assuming it and paying its assumability cost. Hypotheses that can be assumed have finite assumability costs and are called "*assumables"*. Hypotheses that are proved from the assumables are called *"provables"*. A provable can be thought of as an assumable with an infinite assumability cost that we cannot afford. We can assume, without loss of generality, that all rule consequents are provables. To see this, suppose p_i is an assumable with assumability cost $c(p_i) = x$ and suppose it appears as the consequent of at least one rule, r_i. We can create a new hypothesis p_i' with an assumability cost equals to x, set the assumability cost of p_i to ∞ and add the rule: $p_i' \to p_i$. In addition, for the purpose of this paper, we call every hypothesis belonging to the goal set $G \subseteq H$ a sub-goal hypothesis. All sub-goals in our system are provables.

Finding the optimal solution in CBA has been shown to be NP-Hard [Charniak and Shimony, 1994]. The most significant approaches to finding LCP for CBA can be found at [Den, 1994; Ishizuka and Matsuo, 2002; Ohsawa and Ohsawa, 1997]. One of the most successful algorithms uses Integer Linear Programming (ILP) as developed by Santos [Santos Jr., 1994]. Abdelbar *et al.* showed how to solve CBA using High-Order Recurrent Neural Networks (HORN) [Abdelbar *et al.*, 2003; Abdelbar *et al.*, 2005].

Finding the LCP in a CBA system is equivalent to finding the *maximum a posteriori assignment* (MAP) in BN [Charniak and Shimony, 1990; Charniak and Shimony, 1994]. They showed that everything one can do using BN can also be done using CBA, and vice versa. However, despite their equivalence, CBA has many advantages. For instance, it is believed that finding LCP is more efficient than finding MAP, and it may be easier to find heuristics for CBA than for BN [Abdelbar, 1998; Charniak and Shimony, 1994]. Also, using costs instead of probabilities gives another perspective to the problem and is easier for some people to grasp. We have also found that the CBA knowledge representation in terms of rules is more natural for modeling genetic regulation. Finally, Santos has found necessary and sufficient conditions under which a CBA system is polynomially solvable [Santos Jr. and Santos, 1996]. In contrast, conditions for polynomial solvability of finding MAP in BN have not been found, even with applying restrictions on the graphical representation [Shimony, 1994] and even for trying to find an alternative next-best explanation [Abdelbar and Hedetniemi, 1998]. Andrews *et al.* provided a framework for finding MAP for BN using HORN and using CBA as an intermediate representation [Andrews and Bonner, 2009]; the theoretical foundation of that framework shows the strong equivalency between BN/CBA and HORN search spaces [Andrews and Bonner, 2011].

1.2 Biological Data Sources

In this paper, we illustrate the application of CBA to GRN by using it to model and learn the well-studied pheromone pathway in Yeast. We use three well-known data sets for this purpose. The first data set consists of protein-DNA interactions, also called location data or factor-binding data, from [Lee *et al.*, 2002]. The directionality of these interactions is known to be from a protein to a gene in the DNA. The second data set consists of protein-protein interactions from the well-known DIP database. The directions of the protein-protein interactions are not known a priori and are learned by CBA. Finally, we use the knock-out data from [Hughes *et al.*, 2000]. Knock-out data describes the effects of induced deletion mutation experiments on some genes. We do not use the knock-out data to build the skeleton graph. Instead, the effect of the knock-outs is what is being explained by the CBA system, and the resultant explanations annotate the entire graph, which in turn represents the learned GRN model. Both protein-DNA and protein-protein data indicate whether there is an interaction without specifying its sign. CBA uses the knock-out data to infer the sign of each interaction, as well as the directions of protein-protein interactions.

2 The Method

Our modeling method, described below, can be summarized as follows. We first select the elements in the genome that we are interested in, which we call the *elements of interest set*. In this paper, we select all genes and proteins that are known to participate in the pheromone pathway. We then create a skeleton graph. Each node in the skeleton graph represents a gene and its protein product, which is unique across all data sets. An edge in the skeleton graph represents a potential interaction between a pair of nodes. Using this graph, we enumerate all paths between each knocked-out gene and genes it is known to affect. Our CBA builder algorithm then uses these enumerated paths to build a CBA system that effectively annotates the entire graph while looking for the LCP of all the knock-out effects. The resulting annotated graph is the learned model for the GRN under study.

2.1 The Un-annotated (Skeleton) Graph

To build the skeleton graph, we first create a node for each element in our elements-of-interest set. Graph edges are then created in two phases. Phase 1 creates (directed) Protein-DNA edges: we search the location dataset for interactions between nodes of the graph; and for each such interaction, we create a directed edge from the protein to the DNA. Phase 2 creates (undirected) Protein-Protein edges: we search the DIP database for protein-protein interactions between nodes of the graph; and for each such interaction, we create an undirected edge between the two nodes. These two phases result in an un-annotated graph. We then use a recursive breadth-first algorithm to enumerate all possible paths between each knocked-out gene and genes it is known to affect. We call these paths *enumerated* or *potentially valid paths.*

The CBA system (described below) adds annotations to the skeleton graph. In the annotated graph, all edges have signs and existence labels, and in addition, undirected edges have directions. As in [Yeang*, et al.*, 2004], a valid path π_{ka} starts with a knocked-out gene g_k and ends with an observed (or affected) gene g_a, and must satisfy the following constraints:

1. All edges are in forward direction, from g_k to g_a.
2. The aggregate sign of the edges in the path must be consistent with the sign of the knock-out effect.
3. The path must end with a protein-DNA edge.
4. The path must be no longer than a predefined upper bound.
5. If an intermediate gene in the path is also knocked out, it must exhibit a knock-out effect on g_a.
6. Each edge must have one and only one direction and one and only one sign.

We use the potentially valid paths enumerated from the skeleton graph to build a CBA system whose LCP enforces these constraints to give us the best valid paths and a fully annotated graph.

2.2 Example

Figure 1 shows a skeleton graph that might be generated by the algorithm described above.

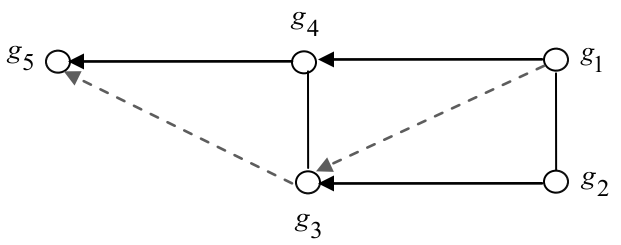

Figure 1: Location data suggests the potential existence of the directed edges. Protein-protein data suggests the potential existence of the undirected edges. Dashed arrows are the knock-out effects to be explained by CBA.

Let us assume that we have two knock-out effects to explain, $(\Delta g_1, g_3, -)$ and $(\Delta g_3, g_5, -)$. The notation $(\Delta g_i, g_j, -)$ means that deletion of gene g_i results in down-regulation of gene g_j. We say that a knock-out effect $(\Delta g_1, g_3, -)$ is explained if there is a valid path that connects both genes. The skeleton graph above suggests three potentially valid paths that might explain these knock-out effects: $g_1 \to g_2 \to g_3$, $g_1 \to g_4 \to g_3$, and $g_3 \to g_4 \to g_5$. Clearly, the direction of the edge between g_3 and g_4 is vital to any explanation of the data. In particular, the direction has to be from g_3 to g_4. If this direction is reversed, we will not be able to explain $(\Delta g_3, g_5, -)$. The two paths that explain all the data are $g_1 \to g_2 \to g_3$ and $g_3 \to g_4 \to g_5$. In addition, to be consistent with the knock-out effects, the edge signs along both paths must be either $(-,-)$ or $(+,+)$. (The aggregate sign of the edges in a valid path must be the opposite of the observed knock-out effect, since the deletion itself is negative.)

3 The CBA Builder Algorithm

This section shows how we use the potentially valid paths to build a CBA system that finds valid paths that provide the best explanation of all the knock-out data and that fully annotate the graph.

3.1 Creating the CBA Hypotheses

For each potentially valid path π_{ka} for a knock-out pair $(\Delta g_k, g_a, \pm)$, we create the following hypotheses:

- k_{kj+}, k_{kj-} and k_{kj0}, for every intermediate gene g_j in π_{ka}, including the observed gene g_a. These hypotheses represent aggregate positive, negative and zero knock-out effects, respectively, from the knocked-out gene g_k to the intermediate gene g_j. k_{kj+} and k_{kj-} are provables while k_{kj0} is an assumable.
- o_{ka}. This is a sub-goal hypothesis representing the observed knock-out effect to be explained.

For each edge E_{ij} between g_i and g_j we create the following hypotheses:

- x_{ij1} and x_{ij0}, assumables representing the existence of the edge in the GRN (*i.e.,* whether or not it represents a real biological interaction).
- s_{ij+} and s_{ij-}, assumables representing the sign of the edge.
- $d_{i\to j}$ and $d_{j\to i}$, assumables representing the direction of the edge. (Used for undirected edges only.)
- s_{ij}, a sub-goal to determine the sign of the edge.
- d_{ij}, a sub-goal to determine the direction of the edge. (Used for undirected edges only.)
- y_{ij}, a sub-goal to determine whether the edge exists in the GRN.

3.2 Creating the CBA Rules

Given a potentially valid path π_{ka} for a knock-out pair $(\Delta g_k, g_a, \pm)$, let the genes on the path be $g_{j_0}, g_{j_1}, ..., g_{j_m}$, where $g_{j_0} = g_k$ and $g_{j_m} = g_a$. For the first edge E_{kj_1} on the path, we create two rules:

- $x_{kj_1 1} \wedge s_{kj_1 +} \wedge d_{k\to j_1} \to k_{kj_1 -}$
- $x_{kj_1 1} \wedge s_{kj_1 -} \wedge d_{k\to j_1} \to k_{kj_1 +}$

For every subsequent edge $E_{j_{n-1}j_n}$, we create four rules:

- $x_{j_{n-1}j_n 1} \wedge s_{j_{n-1}j_n +} \wedge k_{kj_{n-1}+} \wedge d_{j_{n-1}\to j_n} \to k_{kj_n +}$
- $x_{j_{n-1}j_n 1} \wedge s_{j_{n-1}j_n +} \wedge k_{kj_{n-1}-} \wedge d_{j_{n-1}\to j_n} \to k_{kj_n -}$
- $x_{j_{n-1}j_n 1} \wedge s_{j_{n-1}j_n -} \wedge k_{kj_{n-1}+} \wedge d_{j_{n-1}\to j_n} \to k_{kj_n -}$
- $x_{j_{n-1}j_n 1} \wedge s_{j_{n-1}j_n -} \wedge k_{kj_{n-1}-} \wedge d_{j_{n-1}\to j_n} \to k_{kj_n +}$

where the rules contain directionality assumables only if the edge is undirected.

The main feature of these rules is that for any two consecutive edges on the path, the consequents of the rules for the first edge are antecedents of the rules for the second edge. This makes it easy to enforce the second path constraint above, since the consequent of each rule represents the aggregate effect of all the edges along the path from the first edge up to the edge for the rule in question.

For each undirected edge, we create two rules, which intuitively mean that the edge has two possible directions:

- $d_{j_{n-1}\to j_n} \to d_{j_{n-1}j_n}$ and $d_{j_n\to j_{n-1}} \to d_{j_{n-1}j_n}$

For every edge, we create two rules, which intuitively mean that the edge has two possible signs:

- $s_{j_{n-1}j_n+} \to s_{j_{n-1}j_n}$ and $s_{j_{n-1}j_n-} \to s_{j_{n-1}j_n}$

For every edge, we create two rules, which intuitively mean that the edge does or does not exist in the GRN:

- $x_{j_{n-1}j_n1} \to y_{j_{n-1}j_n}$ and $x_{j_{n-1}j_n0} \to y_{j_{n-1}j_n}$

For each knock-out pair $(\Delta g_k, g_a, \pm)$, we create the following three rules, which intuitively mean that the knock out can have three possible effects on the observed gene: up-regulation, down-regulation or leaving it unchanged.

- $k_{ka+} \wedge ck_{ka+} \to o_{ka}$, $k_{ka-} \wedge ck_{ka-} \to o_{ka}$, $k_{ka0} \to o_{ka}$

Here, ck_{ka+} and ck_{ka-} are auxiliary assumables that carry the costs of positive and negative knock-out effects.

Finally, all sub-goal hypotheses $\{y_{j_{n-1}j_n}, d_{j_{n-1}j_n}, s_{j_{n-1}j_n}, o_{ka}\}$ are added to the goal set.

3.3 Assigning Costs to Hypotheses

The last step in creating the CBA system is assigning the proper cost to each assumable. For all s_{ij+}, s_{ij-}, $d_{i\to j}$ and $d_{j\to i}$, we assign an equal and relatively high cost, to force the system to assume only one sign and only one direction while not being biased towards a particular choice. We calculate the cost of the existence hypotheses $\{x_{ij1}, x_{ij0}\}$ and the knock-out effect hypotheses $\{ck_{ka+}, ck_{ka-}\}$ based on observed experimental values reported in the datasets.

Calculating costs for edge existence assumables

The costs of the existence hypotheses $\{x_{ij1}, x_{ij0}\}$ for an edge depend on the likelihood that the edge represents a real biological interaction. These likelihoods are derived from the protein-DNA and protein-protein data. The likelihood itself is not reported in the data, but the p-value is.

We recover the likelihood using the same methods as [Yeang*, et al.*, 2004]. For example, for protein-DNA data, we assume the p-value comes from testing the null hypothesis H_0 ($x_{ij0}=1$, the interaction does not occur) against the alternative hypothesis H_1 ($x_{ij1}=1$, the interaction does occur). Accordingly, we recover the log likelihood that the interaction has occurred as follows:

$$\ell = \frac{P(D \mid x_{ij}=1)}{P(D \mid x_{ij}=0)} \approx \frac{P(D \mid H_1)}{P(D \mid H_0)} e^{\frac{d_0-d_1}{2\log(n)}} = e^{\frac{L}{2}-\frac{1}{2}\log(n)} \tag{1}$$

where D is the observed data, d_0 and d_1 are the degrees of freedom in H_0 and H_1, n is the sample size the p-value was computed from, which is 3 in our case, and L is recovered from the p-value using the following approximation:

$$L = 2\log\frac{P(D \mid H_1)}{P(D \mid H_0)} = F^{-1}(1-p) \tag{2}$$

where p is the reported p-value, and F is the cumulative χ^2 distribution with one degree of freedom. The costs of edge existence are then computed as follows:

$$c(x_{ij0}) = -\log(1/(1+\ell)) \quad \text{and} \quad c(x_{ij1}) = -\log(\ell/(1+\ell)).$$

Calculating costs for knock-out effect assumables

The likelihood of a knock-out interaction can be recovered using the same approximation mentioned above. If the likelihood of the knock-out effect k_{ka} between genes g_k and g_a is ℓ, then the cost of assuming no knock-out effect is

$$c(k_{ka0}) = -\log(1/(1+\ell+\varepsilon)).$$

The cost of assuming a positive knock-out effect k_{ka+} is

$$c(ck_{ka+}) = \begin{cases} -\log(\ell/(1+\ell+\varepsilon)) & \text{if } o_{ka} > 0 \\ -\log(\varepsilon/(1+\ell+\varepsilon)) & \text{if } o_{ka} \le 0 \end{cases} \tag{3}$$

The cost of assuming a negative knock-out effect k_{ka-} is

$$c(ck_{ka-}) = \begin{cases} -\log(\ell/(1+\ell+\varepsilon)) & \text{if } o_{ka} \le 0 \\ -\log(\varepsilon/(1+\ell+\varepsilon)) & \text{if } o_{ka} > 0 \end{cases} \tag{4}$$

Here, o_{ka} is the observed effect of the knock-out on the expression level of gene g_a, and ε is a very small real number representing the possibility that the effect is due to something other than the knock-out experiment.

4 Results

We used integer linear programming (ILP) to solve the resultant CBA system. In particular, we used the popular public domain *LP-solve* to solve the CBA system after converting it to an equivalent ILP program.

In addition to toy examples like Figure 1, we applied our method to the well-studied Yeast pheromone pathway. The skeleton graph consisted of 37 protein-DNA interactions and 30 protein-protein interactions. We tried upper bounds of up to 10 nodes per path. In all cases, the paths discovered to explain the knock-out effects satisfied the valid path constraints mentioned above. Table 1 summarizes the result for different upper bounds:

Path length	2	3	4	5	6	7	8
Hypotheses	346	463	534	826	895	951	897
Rules	297	420	509	935	1281	1421	1465
ILP iterations	448	642	769	1538	1959	2237	2002
Potentially valid paths	31	40	48	159	582	1687	3911

Table 1: This table illustrates the effect of increasing path length on the size and complexity of the CBA instance created.

4.1 Comparison

Many of the elements of our work are based on [Yeang, *et al.*, 2004], which uses a probabilistic approach to infer a GRN of Yeast from multiple biological datasets. They use the same datasets as we do and create what they call a *physical network model*. The main difference is that they use factor graphs instead of CBA, and they compute MAP instead of LCP. We found that the CBA approach produced better results for the pheromone pathway. For instance, their method was only able to find approximate MAP configurations, while CBA was able to find an exact LCP. Moreover, while they used a maximum path length of 5, CBA could easily handle paths of length 10. Finally, because CBA computes the exact LCP, the solution does not have variant parts as in models that compute approximate MAPs.

A complete discussion and comparison of biological results is beyond the scope of the present paper. However, using CBA, we did recover almost all the confirmed signal transduction directionalities in the pheromone pathway, including STE11 $\rightarrow$ STE7, {STE5,STE7} $\rightarrow$ FUS3 and {FUS3,KSS1} $\rightarrow$ STE12. The 21 direct regulations of STE12 were also detected. Neither FUS3 nor KSS1 violated any path constraints. We recovered a consistent relation between STE12 and MCM1 in both signals and directionalities.

The next section elaborates on interesting features of the size of the CBA system produced by our build algorithm. These features explain why our model can handle path lengths longer than other probabilistic models.

4.2 CBA Size Features

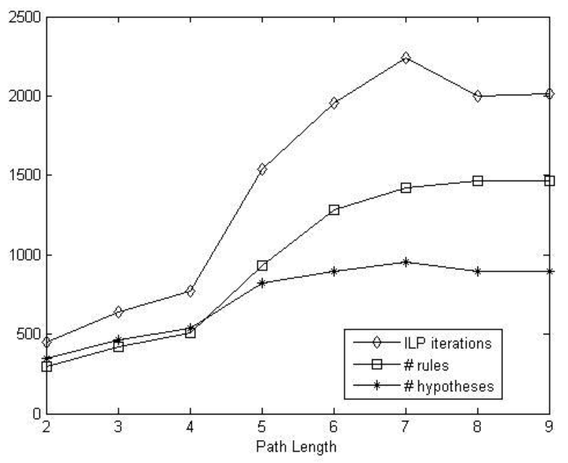

Figure 2: This graph illustrates the effect of increasing path length on the complexity and the size of the CBA instance.

It is easy to verify that apart from the goal rule, our CBA builder will always produce rules that consist of at most 5 variables, including the consequent. This fixed length guarantees that the complexity of the graph will never cause an unexpectedly long constraint when using ILP or an unexpectedly high-order edge when using NN. This short rule length also makes the job of the CBA solver easier. However, the difficulty of a CBA instance depends on more than just rule length. Other factors include solution depth, the ratio of assumables to provables, and the ratio of the number of hypotheses to the number of rules.

In the probabilistic approach of [Yeang, *et al.*, 2004], the size of the factor graph depends on the number of *paths* in the GRN. In contrast, in the approach described here, the size of the CBA system depends only on the number of *edges* in the GRN. This is possible because, as a logical formalism, CBA very naturally deals with both AND and OR constraints and with constraints based on paths and subnetworks in a graph. Consequently, the hypotheses and rules for an edge are created only once, and reused for each path that the edge belongs to. As a result, increasing the number of paths or increasing path length does not result in an exponential increase in the size of the CBA system. Figure 2 illustrates the relation between path length and other aspects of CBA complexity for the pheromone pathway.

5 Concluding Remarks and Future Work

We applied CBA to the modeling of genetic regulatory networks (GRN). CBA can easily integrate biological data from multiple sources and explain the effects of gene knockout experiments. The size of the CBA instance does not increase exponentially with problem size. Our results suggest that CBA can be a useful tool for computational biology in general and for GRN modeling in particular. We tested our algorithm on the pheromone pathway in Yeast. In addition to testing our model on other pathways and larger GRNs, future work includes further optimization of the CBA instance. We need, for instance, to make the path enumeration phase more efficient, either by building the CBA rules directly from the edges without any path enumeration or by using a more efficient algorithm than recursive breadth-first search. It will also be interesting to use the node ordering of the CBA instance to create an equivalent BN and compare performance. Finally, because we use an ILP solver, we can study the characteristics of the resulting constraint matrix and check whether it meets the CBA polynomial-solvability conditions determined by Santos.

References

[Abdelbar, 1998] Ashraf M. Abdelbar. An algorithm for finding MAPs for belief networks through cost-based abduction. *Artificial Intelligence*, 104(1-2):331-338, 1998.

[Abdelbar, et al., 2003] Ashraf M. Abdelbar, Emad A. M. Andrews and Donald C. Wunsch II. Abductive reasoning with recurrent neural networks. *Neural Networks*, 16(5-6):665-673, 2003.

[Abdelbar, et al., 2005] Ashraf M. Abdelbar, Mostafa A. El-Hemely, Emad A. M. Andrews and Donald C. Wunch II. Recurrent neural networks with backtrack-points and negative reinforcement applied to cost-based abduction. *Neural Networks*, 18(5-6):755-764, 2005.

[Abdelbar and Hedetniemi, 1998] Ashraf M. Abdelbar and Sandra M. Hedetniemi. Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102(1):21-38, 1998.

[Andrews and Bonner, 2009] Emad A. M. Andrews and Anthony J. Bonner. Finding MAPs Using High Order Recurrent Networks. In *Proceedings of the 16th International Conference on Neural Information Processing: Part I*. C. S. Leung, M. Lee, and J. H. Chan (Eds.) pages 100-109: Springer-Verlag, Berlin / Heidelberg, 2009.

[Andrews and Bonner, 2011] Emad A. M. Andrews and Anthony J. Bonner. Finding MAPs using strongly equivalent high order recurrent symmetric connectionist networks. *Cognitive Systems Research*, doi:10.1016/j.cogsys.2010.12.013, 2011.

[Baldi and Hatfield, 2003] Pierre Baldi and G. Welesley Hatfield. *DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling*. Cambridge University Press, Cambridge, 2003.

[Beer and Tavazole, 2004] Micheal A. Beer and Saeed Tavazole. Predicting gene expression from sequence. *Cell*, 117(2):185-198, 2004.

[Charniak and Shimony, 1990] Eugene Charniak and Solomon E. Shimony. Probabilistic semantics for cost-based abduction. In *Proceedings of the 8th National conference on Artificial intelligence*, pages 106-111, Boston, Massachusetts, 1990. AAAI Press.

[Charniak and Shimony, 1994] Eugene Charniak and Solomon Eyal Shimony. Cost-based abduction and MAP explanation. *Artificial Intelligence*, 66:345-374, 1994.

[Den, 1994] Yasuharu Den. Generalized Chart Algorithm: An Efficient Procedure for Cost-Based Abduction. In *Proceedings of the 32nd annual Meeting of the Association for Computational Linguistics*, pages 218-225, Las Cruces, New Mexico, 1994. Association for Computational Linguistics.

[Friedman, 2004] Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799-805, 2004.

[Hughes, et al., 2000] Timothy R. Hughes, Matthew J. Marton, Allan R. Jones, Christopher J. Roberts, Roland Stoughton, Christopher D. Armour, Holly A. Bennett, Ernest Coffey, Hongyue Dai, Yudong D. He, Matthew J. Kidd, Amy M. King, Michael R. Meyer, David Slade, Pek Y. Lum, Sergey B. Stepaniants, Daniel D. Shoemaker, Daniel Gachotte, Kalpana Chakraburtty, Julian Simon, Martin Bard and Stephen H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

[Huttenhower, et al., 2006] Curtis Huttenhower, Matt Hibbs, Chad Myers and Olga G. Troyanskaya. A scalable method for integration and functional analysis of multiple microarray datasets. *Bioinformatics*, 22(23):2890-2897, 2006.

[Ishizuka and Matsuo, 2002] Mitsuru Ishizuka and Yutaka Matsuo. SL Method for Computing a Near-Optimal Solution Using Linear and Non-linear Programming in Cost-Based Hypothetical Reasoning. *Knowledge-based systems*, 15(7):369-376, 2002.

[Kauffman, 1993] Stuart Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press., New York, 1993.

[Lee, et al., 2002] Tong Ihn Lee, Nicola J. Rinaldi, Franc¸ois 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*, 298:799–804, 2002.

[Ohsawa and Ohsawa, 1997] Yukio Ohsawa and Mitsuru Ohsawa. Networked bubble propagation: a polynomial-time hypothetical reasoning method for computing near-optimal solutions. *Artificial Intelligence*, 91(1):131-154, 1997.

[Ong, et al., 2007] Irene M. Ong, Scott E. Topper, David Page and Vitor Santos Costa. Inferring Regulatory Networks from Time Series Expression Data and Relational Data Via Inductive Logic Programming. In *Lecture Notes in Artificial Intelligence*. S. Muggleton, R. Otero, and A. Tamaddoni-Nezhad (Eds.) pages 366-378: Springer-Verlag, 2007.

[Papatheodorou, 2007] Irene V. Papatheodorou. *Inference of Gene Relations from Microarray Experiments by Abductive Reasoning*. University of London, PhD thesis, 2007.

[Raedt, et al., 2008] Luc De Raedt, Paolo Frasconi, Kristian Kersting and Stephen Muggleton (Eds). *Probabilistic inductive logic programming: theory and applications*. Springer-Verlag, 2008.

[Ray and Kakas, 2006] Oliver Ray and Antonis Kakas. ProLogICA: a practical system for Abductive Logic Programming. In *Proceedings of the 11th International Workshop on Non-monotonic Reasoning*, pages 304-312, 2006.

[Santos Jr., 1994] Eugene Santos Jr. A linear constraint satisfaction approach to cost-based abduction. *Artificial Intelligence*, 65(1):1-27, 1994.

[Santos Jr. and Santos, 1996] Eugene Santos Jr. and Eugene S. Santos. Polynomial Solvability of Cost-Based Abduction. *Artificial Intelligence*, 86:157-170, 1996.

[Shimony, 1994] Solomon Eyal Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399-410, 1994.

[Yeang, et al., 2004] Chen-Hsiang H. Yeang, Trey Ideker and Tommi Jaakkola. Physical network models. *Journal of computational biology*, 11(1-2):243–262, 2004.