OPEN ACCESS Freely available online

PLOS ONE

# CaSPIAN: A Causal Compressive Sensing Algorithm for Discovering Directed Interactions in Gene Networks

**Amin Emad*, Olgica Milenkovic**

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois, United States of America

## Abstract

We introduce a novel algorithm for inference of causal gene interactions, termed CaSPIAN (Causal Subspace Pursuit for Inference and Analysis of Networks), which is based on coupling compressive sensing and Granger causality techniques. The core of the approach is to discover sparse linear dependencies between shifted time series of gene expressions using a sequential list-version of the subspace pursuit reconstruction algorithm and to estimate the direction of gene interactions via Granger-type elimination. The method is conceptually simple and computationally efficient, and it allows for dealing with noisy measurements. Its performance as a stand-alone platform without biological side-information was tested on simulated networks, on the synthetic IRMA network in *Saccharomyces cerevisiae*, and on data pertaining to the human *HeLa* cell network and the SOS network in *E. coli*. The results produced by CaSPIAN are compared to the results of several related algorithms, demonstrating significant improvements in inference accuracy of documented interactions. These findings highlight the importance of Granger causality techniques for reducing the number of false-positives, as well as the influence of noise and sampling period on the accuracy of the estimates. In addition, the performance of the method was tested in conjunction with biological side information of the form of sparse "scaffold networks", to which new edges were added using available RNA-seq or microarray data. These biological priors aid in increasing the sensitivity and precision of the algorithm in the small sample regime.

**Citation:** Emad A, Milenkovic O (2014) CaSPIAN: A Causal Compressive Sensing Algorithm for Discovering Directed Interactions in Gene Networks. PLoS ONE 9(3): e90781. doi:10.1371/journal.pone.0090781

**Editor:** Daniele Marinazzo, Universiteit Gent, Belgium

**Received** December 4, 2013; **Accepted** February 5, 2014; **Published** March 12, 2014



**Funding:** This work was supported in part by an NSERC graduate fellowship, and NSF grants CCF 0809895, CCF 1218764 and the Emerging Frontiers for Science of Information Center, CCF 0939370. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

**Competing Interests:** The authors have declared that no competing interests exist.

* E-mail: emad2@illinois.edu

## Introduction

One of the unresolved open problems in systems biology is discovering causal relationships among different components of biological systems. Gene regulatory networks, protein-protein interaction networks, chemical signaling and metabolic networks are all governed by causal relationships between their agents that determine their functional roles. Discovering causal relationships through experiments is a daunting task due to the technical precision and output volumes required from the experiments and due to the large number of interconnected and dynamically varying components of the system. It is therefore of great importance to develop a precise analytical framework for quantifying causal connections between genes in order to elucidate the gene interactome based on limited and noisy experimental data. Statistically inferred interactions may be used to guide the experimental design process, helping with further refinement of the modeling framework [1,2]. Unfortunately, to date most reverse engineering algorithms have offered very few reliable outcomes for even moderately sized networks and were hardly ever experimentally tested – these and other shortcomings of existing inference techniques and models were described in detail in [3]. Consequently, algorithmic developments are focusing on small network components of *prokaryotic* or simple *eukaryotic* cell lines and on the more conservative – yet reliable – task of identifying a small number of highly accurate causal links.

One way to detect if a gene causally influences another gene is to monitor if changes in the expression of the potential regulator gene affect the expression of the target gene in the presence of at least one additional network component [4,5]. This type of analysis is frequently used in combination with expression clustering and classification techniques [6]. A number of authors also suggested the use of Bayesian networks [7–11], Boolean networks [12,13], differential equations [13,14], stochastic networks [15,16], finite state linear models [17] and other machine learning tools for network inference. Algebraic techniques were described in [18,19], while different information-theoretic approaches were proposed in [20–27]. For an overview of a variety of other methods for reverse engineering of gene regulatory networks, the interested reader is referred to [28–35].

Sparsity of gene regulatory networks was exploited in a number of different inference frameworks, including transcription factor interaction analysis [2,36–38]. Most of the proposed methods integrate sparsity priors through a form of Lasso penalty [39]. The algorithms reduce to an optimization problem that in its simplest form tries to minimize an objective function consisting of two terms: the first term is the $\ell_2$ norm of the reconstruction error, while the second term is a regularization term, equal to the $\ell_1$ norm of the sought solution. The main difficulties associated with the Lasso framework are solving a high-dimensional optimization problem and properly choosing the coefficient of the regularization term(s). In most cases, the regularization coefficient is either

chosen heuristically or using an optimization procedure which increases the complexity of the algorithm without providing provable performance guarantees. Parameter tuning issues also make the comparison of results generated by Lasso for different objective functions hard to accomplish in a fair manner.

An alternative to the Lasso approach is a greedy compressive sensing framework, which overcomes some of the shortcomings of Lasso while still utilizing the sparsity of the network. Compressive sensing (CS) is a dimensionality reduction technique with widespread applications in signal processing and optimization theory [40], [41]. CS allows for inferring sparse structures given a small number of linear measurements, usually generated in a random fashion. As such, it naturally lends itself for use in biological inference problems involving sparse interaction networks.

Motivated by recent advances in CS theory and its application in practice, we introduce the concept of *causal compressive sensing* and design new greedy list-reconstruction algorithms for inference of causal gene interactions; as part of the process, we generate two sparse models for each potential interaction pattern and infer causality by comparing the residual errors of the models using statistical methods. Furthermore, in CS, the most difficult task consists of finding the support (i.e. the nonzero entries) of a sparse signal. This is accomplished by inferring the subspace in which the vector of observation lies. As a result, the complicated process of choosing the regularization coefficient in Lasso is substituted by the more natural task of choosing a "consistency" level between the vector of observations and its representation in the estimated subspace.

The CS approach has not been widely used for gene regulatory network inference; to the best of the author's knowledge, only the methods in [42] and [43] described compressive sensing algorithms for linear models. Both papers deal with non-causal inference. In our work, we propose a method for identifying causal gene interactions based on a combination of two ideas: greedy CS reconstruction and Granger causality, or elimination analysis. The CS model is motivated by a technique for face recognition used in computer vision, first described in [44]. The crux of the approach is to efficiently find a sparse linear representation of an image of one individual in terms of images of that and other individuals, taken under many different conditions. One component of the setup is reminiscent to the method described in [43], where expression levels of genes taken under different experimental conditions (or under different gene knockout scenarios) are represented as vectors for which a sparse representation is sought. However, the results in [43] are based on an $\ell_1$ optimization method and only infer *non-causal* interaction among the genes. In addition, CS was used only as a preprocessing step; the obtained CS results were combined with extensive prior biological information, and the gene interactions were inferred through supervised learning performed by AdaBoost. It is worth mentioning that AdaBoost and similar boosters are highly susceptible to random classification noise, thereby limiting their applications in biological data analysis [45].

In this manuscript, we propose two causal CS inference approaches. In order to infer causality, we apply these approaches to two different combinations of gene expression profiles shifted in time, one of which contains a potential regulator and another, which does not contain a potential regulator. Each dataset gives rise to a certain representation error, and one may infer the level of influence of genes on each other based on the differences in the representation errors and by using the F-test [46]. The first method is an unsupervised learning scheme and therefore has the advantage that it does not need to be combined with learning steps involving biological side-information in order to produce good predictions. In the second method, in order to incorporate biological priors into the subspace pursuit process, we propose using an experimentally verified "scaffold network". This network consists of a small number of highly reliable edges, chosen based on the number of times they were reported in the literature, the number of different experimental methods used to verify them and similar considerations. The use of a scaffold network may resolve some ambiguities in the subspace selection process, which often lead to inference errors and hence improve the overall performance of causal CS methods.

At the core of our computational method is the subspace pursuit (SP) algorithm, which we described in [47]. We adapt this greedy approach into an algorithm termed list-SP. List-SP sequentially scans subspaces of measurements with different dimensions and creates an output that consists of the union of basis vectors for all identified subspaces. An advantage of the proposed algorithms is that one can take advantage of the prior information on the sparsity of the network, i.e. the in-degree of the nodes. If such information is not available, a rough estimate or an upper bound on the in-degree of the nodes is sufficient. In particular, the upper bound can be chosen large enough to ensure that it exceeds the largest in-degree of the network, and then the false-positives can be rejected throughout the F-test step of the inference algorithm.

The main finding of our analysis indicates that causal compressive sensing can infer a relatively large fraction of causal gene interactions with very small false-positive rates when applied to small and moderate size networks. This finding is supported by simulated data, synthetic data from the IRMA network in *Saccharomyces cerevisiae* [48], and biological data from the human HeLa cell network and the SOS network of *E. coli* [49]. The success probability is, as expected, highly influenced by the noise variance of the experiment and by the sampling time of the expressions. Our analysis of these phenomena adds to the understanding of the limitations of causal inference under imperfect measurement conditions, as well as the role of biological side information in reducing inference error rates. It also explains why available methods may not result in an improved detection probability upon adding as many time-shifted expression profiles as available, since gene expressions are usually measured at too widely separated times and have different time periods between the measurements.

## Methods

### Compressive Sensing

Compressive sensing (CS) is a technique for sampling and reconstructing sparse signals, i.e. signals that can be represented by $k \ll n$ significant coefficients over an $n$-dimensional basis. What distinguishes CS from other dimensionality reduction techniques is that it operates with a small number of measurements [40], [41] that allow for polynomial-time reconstruction of the sparse signal.

Assume that one is interested in finding a vector $\tilde{\mathbf{x}} \in \mathbb{R}^n$ using a (noisy) observation $\mathbf{y} \in \mathbb{R}^m$ obtained according to $\mathbf{y} = \mathbf{\Phi}\tilde{\mathbf{x}} + \mathbf{n}$, for a known sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$, with $m < n$; here, $\mathbf{n}$ denotes the noise vector. In general, the problem cannot be solved uniquely. However, if $\tilde{\mathbf{x}}$ is $k$-sparse, i.e., if it has up to $k$ nonzero entries, one may recover $\tilde{\mathbf{x}}$ uniquely if $m$ is large enough. This can be achieved by finding the sparsest signal consistent with the vector of measurements [40], i.e.

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{\Phi}\mathbf{x}\|_2^2 \leq \epsilon, \qquad (1)$$

where $||\mathbf{x}||_0$ denotes the $\ell_0$ norm of $\mathbf{x}$ (i.e., the number of non-zero entries of $\mathbf{x}$), while $\epsilon$ denotes a parameter that depends on the level of measurement noise. It can be shown that the $\ell_0$ minimization method can exactly reconstruct the original signal in the absence of noise using a properly chosen sensing matrix $\mathbf{\Phi}$ whenever $m \geq 2k$. However, $\ell_0$ minimization is a computationally hard combinatorial problem and cannot be performed efficiently.

On the other hand, it is known that an $\ell_1$ convex relaxation of (1) can accurately approximate the signal $\tilde{\mathbf{x}}$ in polynomial time if $\mathbf{\Phi}$ satisfies the so-called restricted isometry property (RIP) and provided that $m = O(k \log(n/k))$ [40], [41]. This optimization problem may be stated as

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{R}^n} ||\mathbf{x}||_1 \quad \text{subject to} \quad ||\mathbf{y} - \mathbf{\Phi}\mathbf{x}||_2^2 \leq \epsilon, \tag{2}$$

where, as before, $\epsilon$ depends on the noise variance. One should note that the $\ell_1$ minimization in Equation (2) is closely related to the previously mentioned Lasso problem [39]

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} ||\mathbf{y} - \mathbf{\Phi}\mathbf{x}||_2^2 + \lambda ||\mathbf{x}||_1, \tag{3}$$

where $\lambda$ is a regularization parameter. If $\epsilon$ and $\lambda$ in Equations (2) and (3) satisfy some special conditions, the two problems are equivalent; however, characterizing the relationships between $\epsilon$ and $\lambda$ is difficult, except for the special case of orthogonal sensing matrices $\mathbf{\Phi}$ [54]. In most other cases, it is easier and more natural to find an appropriate value of $\epsilon$ than $\lambda$ [51], since $\epsilon$ is proportional to the noise variance. As a result, the CS framework eliminates the computational issues of Lasso regarding parameter selection.

Although the $\ell_1$ relaxation can be reformulated as a linear program (LP), for high-dimensional vectors it is desirable to use greedy algorithms as they offer significant reductions in computational complexity while ensuring performance comparable to that of LP methods. Another advantage of greedy methods is the ease of adding constraints and adapting the method to the problem at hand. For an in-depth discussion of one such greedy algorithm, Subspace Pursuit (SP), the interested reader is referred to [47].

## The List-SP algorithm

We next introduce the List-SP method, a modification of the SP algorithm [47] that is designed to increase the number of true positives found using SP while preserving the benefits of SP (including low complexity, ability to incorporate side information on the in-degree of the nodes, etc.).

Assume that for a vector $\mathbf{y}$, a sensing matrix $\mathbf{\Phi}$, and a fixed value of $\epsilon$, one is interested in finding a $k$-sparse estimate of $\tilde{\mathbf{x}}$, denoted by $\hat{\mathbf{x}}$, such that $||\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}||_2^2 \leq \epsilon$. The main challenge is to find the support of $\hat{\mathbf{x}}$, or the columns of $\mathbf{\Phi}$ indexed by $\hat{\mathbf{x}}$; given the support, the values of the non-zero entries may be obtained via pseudo-inversion [52]. List-SP is an iterative algorithm for solving this problem in $k$ iterations. In the $\kappa^{\text{th}}$ iteration, $1 \leq \kappa \leq k$, a set of $\kappa$ columns of $\mathbf{\Phi}$ is identified, such that the linear combination of these columns represents $\mathbf{y}$ with smallest possible error. The union of the sets of columns found during these $k$ iterations forms a subspace that with high probability contains the subspace spanned by the columns of $\mathbf{\Phi}$ indexed by $\hat{\mathbf{x}}$. In order to find the set of columns in the $\kappa^{\text{th}}$ iteration, SP first finds a set of $\kappa$ columns of the sensing matrix with highest correlation with $\mathbf{y}$; then, iteratively, the SP algorithm tests groups of columns in a greedy manner to augment the existing set of $\kappa$ columns of $\mathbf{\Phi}$ with $\kappa$ additional columns that together most likely span the subspace in which $\mathbf{y}$ lies. Upon finding such a set, SP updates the list of columns by discarding the $\kappa$ "least reliable" of them. The procedure continues until a desired accuracy is achieved [47]. Note that the reason for forming the union of subspaces in list-SP is that it may happen that some of the recovered columns for sparsity level $\kappa - 1$ are not present among the recovered columns for sparsity level $\kappa$. By combining the results obtained using SP for different values of $\kappa$, we reduce the chance of missing an important column of $\mathbf{\Phi}$, given that we do not know the sparsity level. In the context of gene interaction inference, we effectively reduce the probability of false-negatives. This significantly improves the performance of the List-SP compared to SP with respect to false-negatives, as we demonstrate in the results section. Note that the computational complexity of List-SP algorithm is $O(mnk^2)$, compared to the complexity of the classical SP, which equals $O(mnk)$ [47].

## Granger Causality

Only a few algorithms using signal sparsity were successfully integrated into causal inference models [2,36]. One such causality testing scheme, originally proposed in econometrics, is Granger causality [53]. In its original incarnation, Granger causality was presented as a heuristic statistical concept based on prediction. The method has the goal to determine if a time series of past observations of a process helps to predict the future values of another process. Granger causality only considers two stochastic stationary processes, in addition to an auxiliary process required as a "causal reference" [5]. An extension of this definition to more than two stationary processes, dubbed conditional Granger causality, was introduced in [54]. In the context of linear regression, this causal model may be described as follows. Assume that the value of the process $Y$ at time $t$ can be predicted using the values of the processes $Y$ and $Z$ at $d$ past time-points through the coefficients $a_i$ and $c_i$, $1 \leq i \leq d$. The *restricted model* takes the form

$$Y^{(1)}(t) = \sum_{i=1}^{d} a_i Y(t - iT) + \sum_{i=1}^{d} c_i Z(t - iT) + r^{(1)},$$

where $Z$ is the reference process, and $r^{(1)}$ represents the approximation residual. Augmenting this model with the $d$ past values of $X$ yields the *unrestricted model*

$$Y^{(2)}(t) = \sum_{i=1}^{d} a_i' Y(t - iT) + \sum_{i=1}^{d} b_i' X(t - iT) + \sum_{i=1}^{d} c_i' Z(t - iT) + r^{(2)},$$

where $a_i'$, $b_i'$, and $c_i'$, $1 \leq i \leq d$, are model coefficients, and $r^{(2)}$ represents the approximation residual. If the variance of $r^{(2)}$ is "significantly" smaller than the variance of $r^{(1)}$ in a suitable statistical framework, then $X$ Granger-causes $Y$ conditioned on $Z$, which we denote by $X \xrightarrow{G} Y|Z$.

In most cases, causality and co-integration are assessed via Augmented Dickey Fuller (ADF) tests or F-tests on the residual. In a nutshell, these tests determine the significance of the change in the value of the variance of the residuals [46]. Assume that one is given different realizations of the processes $X$, $Y$, and $Z$. These realizations can be used to form vector time-series models of the restricted and unrestricted models. To determine if $X \xrightarrow{G} Y|Z$, we assume the null hypothesis that $b_1' = b_2' = \ldots = b_d' = 0$. The F-statistic for this null hypothesis is

$$F=\frac{\left(\|\mathbf{r}_1\|^2-\|\mathbf{r}_2\|^2\right)/(n_2-n_1)}{\|\mathbf{r}_2\|^2/(m-n_2)}, \quad (4)$$

where $\|\mathbf{r}_1\|^2$ and $\|\mathbf{r}_2\|^2$ are the squared norms of the residuals in the restricted model and the unrestricted model, respectively; $n_1$ and $n_2$ are the total number of parameters in the restricted and unrestricted model, respectively, and $m$ is the total number of realizations of the processes at each time point. Under the null hypothesis, the F-statistic follows an F-distribution with $(n_2-n_1,m-n_2)$ degrees of freedom [46]. The null hypothesis is rejected if the F-statistic is greater than a critical value, calculated using the F-distribution for a desired significance level, $P_F$.

In what follows, the CS-Granger causality approach is applied on gene expression data sampled at a small number of time instances. There are two main issues to be addressed in this context: how to discover linear relationships between expression profiles that may be (and usually are) correlated with each other and how to adapt the sensing matrix $\mathbf{\Phi}$ to perform meaningful Granger-type tests. The processes of interest, $X$, $Y$ and $Z$ represent the regulated gene, the regulator gene and a *collection of candidate genes*, which contain both genes that causally influence $X$ (excluding $Y$) and genes that do not causally influence $X$. In other words, $Z$ is a vector random process $(Z_1,\ldots,Z_s)$, where $s$ denotes the number of candidate genes.

## The CaSPIAN algorithm for gene network inference

In order to combine CS techniques, in particular List-SP, with Granger causality, we assume that the gene expressions may be modeled via a linear regression. In addition, we assume that different realizations of the model are available through different experiments. These realizations are used to form a vector regression model for the gene expressions. More precisely, assume that one is interested in finding the directed graph corresponding to causal relationships of $n$ genes, denoted by $\mathcal{G}=\{g_1,g_2,\ldots,g_n\}$, using gene expression levels obtained under different experimental conditions. If the structure of the gene regulatory network does not change significantly under these conditions, one should be able to form expression profiles for each gene by concatenating the expression levels in different experiments. In particular, conditions that do not affect a subnetwork of the gene regulatory network can be used to infer the causal relationships among the genes of the subnetwork of interest. The procedure of forming the gene expression profiles is described in more detail at the end of this section.

Let $m$ denote the number of experiments and assume that in each experiment the expression level of each gene is given for $d$ time points. Let $\{\phi_{j,l}\}$ denote the set of expression profiles of genes, where $j\in\{1,2,\ldots,n\}$ and $l\in\{0,1,\ldots,d-1\}$; $\phi_{j,l}$ is a column-vector denoting the expression profile of $g_j$ corresponding to time $t=t_0-lT$, where $t_0$ is the latest available time-point and $T$ is the sampling period between the time-points. Assume that one is interested in finding the genes causally affecting a gene of interest, $g_i$, dubbed the *target gene*; in this case, we set $\mathbf{y}=\phi_{i,0}$, the expression profile of $g_i$ corresponding to time $t=t_0$. The idea is to relate the linear regression model governing the dynamics of gene expression level of $g_i$ to Equation (1) and use CS techniques to infer $\tilde{\mathbf{x}}$; the support of $\tilde{\mathbf{x}}$ can then be used to identify the genes affecting $g_i$. In addition, the sign of the nonzero entries of $\tilde{\mathbf{x}}$ and their values can be used to infer if a gene positively or negatively regulates another gene.

The sensing matrix is formed using the expression profiles of all the genes other than $g_i$, at all past time points, according to the formula:

$$\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}=[\phi_{1,1},\phi_{1,2},\ldots,\phi_{i-1,d-1},\phi_{i+1,1},\ldots,\phi_{n,d-1}]. \quad (5)$$

Note that in forming the matrix $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$, we did not use the expression profiles corresponding to time $t=t_0$. The reason behind this is that we assume that the profile of a gene at time $t=t_0$ depends on the profile of other genes at *previous* time-points. In addition, we did not include the expression profiles of $g_i$ corresponding to past time-points. This is justified by the fact that even if $\phi_{i,0}$ can be written as a linear combination of $\phi_{i,j}$, $1\le j\le d-1$, which is the case in most scenarios, it does not imply that the gene self-regulates; as a result, including such columns in the sensing matrix and treating them in the same way as profiles of other genes will result in false-positives and will also mask the effect of the true regulators of the target gene. One should also note that the sensing matrix formed in this way may not satisfy some of the properties reported in CS literature for analytical performance guarantees. For example, there is no guarantee that the matrix will satisfy an RIP-like condition; the RIP is a *sufficient condition* for recovery and may not be necessary for the algorithm to work. Our results, similar to the results pertaining to face recognition [44] show that many interactions can be inferred using this approach despite the fact that the sensing matrices may not satisfy the RIP.

List-SP infers gene interactions based on $\mathbf{y}$ and $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$, as summarized in Algorithm 1 (Table 1). This algorithm identifies genes that accurately "explain" the behavior of the target gene, which include, but are not restricted to, genes causally influencing the target gene. Hence, the set $\mathcal{R}_i$ may contain false-positives.

In order to identify which genes in the set $\mathcal{R}_i$ causally influence $g_i$, we perform an F-test for each $g_j\in\mathcal{R}_i$, for a given significance level, $P_F$. For this purpose, we need to calculate the vector of residuals in the unrestricted regression model and restricted regression model. Let $\mathbf{\Phi}_{\mathcal{T}_i}$ be a matrix formed using the columns in $\mathcal{T}_i^{(k)}$ recovered by Algorithm 1. The residual vector of representing $\mathbf{y}$ as a linear combination of the columns in $\mathcal{T}_i^{k}$ (i.e. the residual vector of unrestricted model), is calculated according to $\mathbf{r}_i=\mathbf{y}-\mathbf{\Phi}_{\mathcal{T}_i}\mathbf{\Phi}_{\mathcal{T}_i}^{\dagger}\mathbf{y}$, where "$\dagger$" denotes the Moore-Penrose pseudo-inverse [52]. For each $g_j\in\mathcal{R}_i$, we form $\mathbf{\Phi}_{\mathcal{T}_{i,j}}$ by removing the columns of $\mathbf{\Phi}_{\mathcal{T}_i}$ corresponding to $g_j$. The vector of residuals for each $g_j$ in the restricted model is computed according to $\mathbf{r}_{i,j}=\mathbf{y}-\mathbf{\Phi}_{\mathcal{T}_{i,j}}\mathbf{\Phi}_{\mathcal{T}_{i,j}}^{\dagger}\mathbf{y}$. These residual vectors are used in Equation (4), the result of which is subsequently used to reject or accept the hypothesis that $g_j$ is conditionally Granger-causal for $g_i$, for significance level $P_F$. The main steps of the procedure are

**Table 1.** Algorithm 1: List-SP.

| |
|---|
| **Input:** $i\in\{1,2,\ldots,n\}$, $\{\phi_{j,l}\}$, and $k\in\mathbb{N}$ |
| **Output:** $\mathcal{R}_i\subset\mathcal{G}$ |
| **Initialization:** $\mathbf{y}=\phi_{i,0}$, $\mathcal{S}_i^{(0)}=\varnothing$, $\mathcal{T}_i^{(0)}=\varnothing$; form $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$ |
| **For** $\kappa=1,2,\ldots,k$ **do** |
| • Run SP for vector $\mathbf{y}$, sensing matrix $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$, and sparsity $\kappa$ |
| • Form $\mathcal{T}_i^{(\kappa)}=\mathcal{T}_i^{(\kappa-1)}\cup\{\kappa$ *columns of* $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$ *recovered using SP*$\}$ |
| **End** |
| **Return** $\mathcal{R}_i=$ the set of genes corresponding to columns in $\mathcal{T}_i^{(k)}$ |

The pseudocode corresponding to the List-SP algorithm.
doi:10.1371/journal.pone.0090781.t001

**Table 2.** Algorithm 2: CaSPIAN.

| |
|---|
| **Input:** $i\in\{1,2,\ldots,n\}$, $\{\phi_{j,l}\}$, $k\in\mathbb{N}$, and $P_F$ |
| **Output:** $\mathcal{S}_i\subset\mathcal{G}$ |
| **Initialization:** $\mathbf{y}=\phi_{i,0}$, $\mathcal{S}_i^{(0)}=\varnothing$, $\mathcal{T}_i^{(0)}=\varnothing$; form $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$ |
| **For** $\kappa=1,2,\ldots,k$ **do** |
| • Run SP for vector $\mathbf{y}$, sensing matrix $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$, and sparsity $\kappa$ |
| • Form $\mathcal{T}_i^{(\kappa)}=\mathcal{T}_i^{(\kappa-1)}\cup\{\kappa \textit{ columns of } \mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}} \textit{recovered using SP}\}$ |
| **End** |
| • Form $\mathcal{R}_i=\{g_{i_1},g_{i_2},\ldots\}$ |
| • Form $\mathbf{\Phi}_{\mathcal{T}_i}$ using $\mathcal{T}_i^{(k)}$ and set $\mathbf{r}_i=\mathbf{y}-\mathbf{\Phi}_{\mathcal{T}_i}\mathbf{\Phi}_{\mathcal{T}_i}^{\dagger}\mathbf{y}$ |
| **For** $j=1,2,\ldots,\lvert\mathcal{R}_i\rvert$ **do** |
| • Form $\mathbf{\Phi}_{\mathcal{T}_{i,j}}$ and calculate $\mathbf{r}_{i,j}=\mathbf{y}-\mathbf{\Phi}_{\mathcal{T}_{i,j}}\mathbf{\Phi}_{\mathcal{T}_{i,j}}^{\dagger}\mathbf{y}$ |
| • Form the F-statistic using $\lVert\mathbf{r}_i\rVert$ and $\lVert\mathbf{r}_{i,j}\rVert$ |
| **If** the F-statistic is greater than critical value corresponding to $P_F$ |
| **Then** set $\mathcal{S}_i^{(j)}=\mathcal{S}_i^{(j-1)}\cup\{g_{i_j}\}$ |
| **Else** set $\mathcal{S}_i^{(j)}=\mathcal{S}_i^{(j-1)}$ |
| **End** |
| **Return** $\mathcal{S}_i=\mathcal{S}_i^{(\lvert\mathcal{R}_i\rvert)}$ |

The pseudocode corresponding to the CaSPIAN algorithm.
doi:10.1371/journal.pone.0090781.t002

summarized in Algorithm 2 (Table 2), and the method is termed Causal Subspace Pursuit for Inference and Analysis of Networks (CaSPIAN).

## CaSPIAN with prior subnetwork knowledge

In many practical situations, some directed edges in the network are known in advance, due to extensive experimental confirmations of their existence. In such cases, one can leverage this side-information to improve the performance of CaSPIAN, especially when the number of time points available for inference is small. For any $i\in\{1,2,\ldots,n\}$, let $\mathcal{A}_i\subset\mathcal{G}$ be the set of genes in the known subnetwork that causally influence $g_i$. The idea is to first remove the influence of the set of genes in $\mathcal{A}_i$ from $\phi_{i,0}$, and then to run CaSPIAN on the residual vector. Let $\mathbf{\Phi}_{\mathcal{A}_i}$ be a matrix formed by setting the profiles of genes in $\mathcal{A}_i$ as its columns. Using standard least square methods, the best representation of $\phi_{i,0}$ as a linear combination of the columns of $\mathbf{\Phi}_{\mathcal{A}_i}$ is equal to $\mathbf{\Phi}_{\mathcal{A}_i}\mathbf{\Phi}_{\mathcal{A}_i}^{\dagger}\phi_{i,0}$, where "†", as before, denotes the Moore-Penrose pseudo-inverse. As a result, we set $\mathbf{y}=\phi_{i,0}-\mathbf{\Phi}_{\mathcal{A}_i}\mathbf{\Phi}_{\mathcal{A}_i}^{\dagger}\phi_{i,0}$, and run CaSPIAN for this choice of $\mathbf{y}$ and the sensing matrix formed by all the profiles of genes in $\mathcal{G}\setminus(\mathcal{A}_i\cup\{g_i\})$. The steps of this method are presented in Algorithm 3 (Table 3).

Note that this approach of including the given side-information may not be optimal: one may try to include the information of existing edges into the subspace selection process of the SP method directly, at each of its iteration. Unfortunately, this approach may be computationally more demanding than using pseudo-inversion followed by List-SP and will not be discussed in this paper.

## Forming gene expression profiles

The gene expression profiles $\phi_{i,j}$, $1\le i\le n$ and $0\le j\le d-1$, can be formed in different ways. If a large number of experiments ($m$) is provided, and each experiment includes exactly $d$ time-points, one can form $\phi_{i,j}$ as a vector of length $m$ including all the expression levels of gene $g_i$ at time $t_0-jT$. This method was used in [2]. There are two main issues associated with forming $\phi_{i,j}$ this way. First, $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$ is a matrix of size $m\times(n-1)(d-1)$; since the number of genes is usually much larger than the number of experiments in a dataset, i.e. $n\gg m$, the number of rows in the sensing matrix is much smaller than the number of columns, which significantly deteriorates the performance of any CS-based (or Lasso-based) algorithm. This is due to the fact that the number of columns of $\mathbf{\Phi}_{\mathcal{G}\setminus\{g_i\}}$ is equal to the number of unknown variables (i.e. length of $\tilde{\mathbf{x}}$), and the number of its rows is equal to the number of known observations (i.e. length of $\mathbf{y}$). Second, in cases where the number of time-points varies from experiment to experiment, one may not be able to use the additional time-points available in some experiments; the number of "useful" time-points is limited to the minimum number of available time-points over all experiments.

In order to overcome these problems, we use a different method to form $\{\phi_{i,j}\}$. Assume that we want to test if the expression level of a gene $g_i$ at a given time $t$ is influenced by the expression levels of other genes at past time-points, considering a time-lag up to $(D-1)T$, with $D\ge 2$. Let $\mathcal{E}_\Delta$ denote the set of experiments for which at least $\Delta$ time-points are available, where $\Delta\ge D+1$. We denote by $m_\Delta$ the number of these experiments, i.e. $m_\Delta=\lvert\mathcal{E}_\Delta\rvert$. Also, let $d_l$, $1\le l\le m_\Delta$, denote the number of time-points available for the $l^{\text{th}}$ experiment in $\mathcal{E}_\Delta$. We form $\phi_{i,j}$, $0\le j\le D-1$ by concatenating the expression levels of $g_i$ corresponding to a subset of the time-points available in each experiment; the time points we use from the $l^{\text{th}}$ experiment are the expression levels from time $t_0-(d_l-D-j)T$ to $t_0-jT$, which significantly increases the length of $\mathbf{y}$ and uses all the available information in the expression data. This results in expression profile vectors of length $\sum_{l=1}^{m_\Delta}d_l-(D-1)m_\Delta\ge 2m_\Delta$. Consequently, the sensing matrix is of size $(\sum_{l=1}^{m_\Delta}d_l-(D-1)m_\Delta)\times(n-1)(D-1)$.

## Normalization of expression data

Two normalizations are performed on the vectors of gene expressions prior to running the algorithms. Given an expression profile, $\phi_{i,j}$, we first subtract the average expression level of this vector from each of its entries. Hence, the normalized profile contains both positive and negative entries. After this step, we normalize the values such that the $\ell_2$ norm of each profile is equal to one. The reason for these normalizations are to reduce the bias and non-uniformities in the expression level of different genes and ensure proper conditions for the operation of the list-SP method. In addition to normalization, we also add an all-one column to the sensing matrix. This step allows us to capture the effect of steady-state values of the profile of the target gene.

## Choosing the parameters of CaSPIAN

CaSPIAN has two input parameters, $P_F$ and $k$. The value of $P_F$ depends on the application at hand. As the parameter $P_F$ is used as a threshold in the F-test to infer Granger-causality, if $P_F$ is very small, the number of false-positives is very low. This increases the precision of the algorithm. In return, the sensitivity of the algorithm reduces as well. A well accepted range of values for $P_F$ is $0.01\le P_F\le 0.05$ which provides a balance between the sensitivity and precision. On the other hand, in applications where finding true positives is more important than finding all the edges, a significantly smaller value for the parameter may be chosen. We defer an in-depth discussion of this issue to the next section, where we compare results for three different choices of $P_F$ values.

On the other hand, the choice for $k$ depends on the available information regarding the network and its degree distribution. The compressive sensing SP algorithm [47] assumes that $k$ is given a

**Table 3.** Algorithm 3: CaSPIAN given a known subnetwork.

| |
|---|
| **Input:** $i\in\{1,2,\ldots,n\}$, $\mathcal{A}_i\subset\mathcal{G}$, $\{\phi_{j,l}\}$, $k\in\mathbb{N}$, and $P_F$ |
| **Output:** $\mathcal{S}_i\subset\mathcal{G}$ |
| **Initialization:** Form $\mathbf{\Phi}_{\mathcal{A}_i}$ and $\mathbf{\Phi}_{\mathcal{G}\setminus(\mathcal{A}_i\cup\{g_i\})}$; set $\mathbf{y}=\phi_{i,0}-\mathbf{\Phi}_{\mathcal{A}_i}\mathbf{\Phi}_{\mathcal{A}_i}^{\dagger}\phi_{i,0}$, $\mathcal{S}_i^{(0)}=\varnothing$, and $\mathcal{T}_i^{(0)}=\varnothing$ |
| **For** $\kappa=1,2,\ldots,k$ **do** |
| • Run SP for vector $\mathbf{y}$, sensing matrix $\mathbf{\Phi}_{\mathcal{G}\setminus(\mathcal{A}_i\cup\{g_i\})}$, and sparsity $\kappa$ |
| • Form $\mathcal{T}_i^{(\kappa)}=\mathcal{T}_i^{(\kappa-1)}\bigcup\{\kappa\ columns\ of\ \mathbf{\Phi}_{\mathcal{G}\setminus(\mathcal{A}_i\cup\{g_i\})} recovered\ using SP\}$ |
| **End** |
| • Form $\mathcal{R}_i=\{g_{i_1},g_{i_2},\ldots\}$ |
| • Form $\mathbf{\Phi}_{\mathcal{T}_i}$ using $\mathcal{T}_i^{(k)}$ and set $\mathbf{r}_i=\mathbf{y}-\mathbf{\Phi}_{\mathcal{T}_i}\mathbf{\Phi}_{\mathcal{T}_i}^{\dagger}\mathbf{y}$ |
| **For** $j=1,2,\ldots,\lvert\mathcal{R}_i\rvert$ **do** |
| • Form $\mathbf{\Phi}_{\mathcal{T}_{i,j}}$ and calculate $\mathbf{r}_{i,j}=\mathbf{y}-\mathbf{\Phi}_{\mathcal{T}_{i,j}}\mathbf{\Phi}_{\mathcal{T}_{i,j}}^{\dagger}\mathbf{y}$ |
| • Form the F-statistic using $\lVert\mathbf{r}_i\rVert$ and $\lVert\mathbf{r}_{i,j}\rVert$ |
| **If** the F-statistic is greater than the critical value corresponding to $P_F$ |
| **Then** set $\mathcal{S}_i^{(j)}=\mathcal{S}_i^{(j-1)}\bigcup\{g_{i_j}\}$ |
| **Else** set $\mathcal{S}_i^{(j)}=\mathcal{S}_i^{(j-1)}$ |
| **End** |
| **Return** $\mathcal{S}_i=\mathcal{S}_i^{(\lvert\mathcal{R}_i\rvert)}\bigcup\mathcal{A}_i$ |

The pseudocode corresponding to the CaSPIAN algorithm given a known subnetwork.
doi:10.1371/journal.pone.0090781.t003

priori. In the context of gene network inference, this is equivalent to knowing the in-degree of each gene in the network. However, by adapting this compressive sensing algorithm to gene network inference applications, the resulting List-SP and CaSPIAN do not require the knowledge of all in-degrees of nodes in the network. Instead, a somewhat tight upper bound suffices, which may be chosen to be the largest reported degree of a hub gene. Hubs in GRNs and protein-protein interactions may have degrees ranging between 5 and 20, as discussed in [55]. If the value of $k$ is chosen slightly smaller than the maximum in-degree, the performance of CaSPIAN does not deteriorate noticeably. On the other hand, a highly overestimated value of $k$ increases the number of false-positives in the list-SP algorithm; however, the F-test embedded in CaSPIAN reduces the number of false-positives. The effect of different choices of $k$ compared to the maximum in-degree of the network is discussed in the next section as well as in the supporting information.

## Results and Discussion

We evaluated the performance of the proposed algorithms with respect to the choice of different parameters such as the sparsity level $k$, the significance value $P_F$, the topology of network, the noise level, and the time-point sampling method. In addition, we compared the performance of these algorithms with that of other causal inference algorithms.

In order to evaluate the effect of different parameters on the performance of the algorithms, we employed synthetic (simulated) networks. Synthetic networks have tightly controlled design parameters, such as the maximum and minimum degree, degree distribution, gene expressions' dynamics, noise level, and sampling frequency. Hence, they allow for accurate assessment of the effect of different parameters on the performance of reverse engineering methods.

On the other hand, synthetic network models usually lack a number of complex features of biological networks that may be hard to model or unknown to the designer. As a result, a fair comparison of different reverse engineering methods require using biological networks. Consequently, we used the IRMA network in *Saccharomyces cerevisiae* [48], the human HeLa cell network, and the SOS network of *E. coli* [49]) to compare CaSPIAN with other known reverse engineering algorithms.

We also used IRMA and human HeLa cell networks to discuss the effect of side-information on the performance of Algorithm 3. In particular, we addressed the effect of employing a known *correct* subnetwork on the performance of CaSPIAN. In addition, we addressed the effect of using an *incorrect* subnetwork on the performance of this algorithm.

In our comparisons, we used four standard evaluation measures, "sensitivity" (recall), "precision", "$F$-measure" ($F$-score, not to be confused with the Granger $F$ statistics) and "accuracy". These measures are defined as $P=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$, $S=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$, $F=2\frac{P\times S}{P+S}$ and $A=\frac{TP+TN}{TP+TN+FP+FN}$, respectively; "TP" stands for the number of true-positives, "FP" stands for the number of false-positives, "TN" stands for the number of true-negatives, and "FN" stands for the number of false-negatives.

### Influence of different parameters on the performance of the algorithms

As described earlier, we used synthetic networks to evaluate the performance of CaSPIAN with respect to different parameters. The constructed synthetic networks follow a model representing a modification of the Erdös-Rényi model with controlled degree distributions and with additional features that allow its dynamics to converge to a steady-state. A detailed description of the model, which we subsequently refer to as the synthetic network, is given in the supporting information.

**The over-sampled regime with noise-free expressions.** We start by evaluating the performance of SP, List-SP, and CaSPIAN for different values of $P_F$, when the number of time-points is larger than the number of genes in the network. We randomly generated 200 synthetic networks comprising $n=10$ genes that are described with the accompanying performance plots. We selected these parameters given that all the biological networks analyzed in the subsequent sections will have a number of nodes of this order. In addition, given that the number of available time points for analysis of regulatory networks usually does not exceed a dozen, larger networks are unlikely to be inferred with any of the existing methods.

Figures S1 and S5 in Appendix S1 illustrate the average sensitivity of different algorithms with respect to $m$ (the length of the gene time-series profiles), for different values of $k$ and for two different distributions used for forming the gene expressions (as detailed in the supporting information). We ran CaSPIAN with $D=2$ (one time unit lag) and for three significance values: $P_F=10^{-2}$, $P_F=10^{-9}$, and $P_F=10^{-16}$. These widely different significance values allow us to evaluate the effect of $P_F$ on the performance of the method. As can be seen, the sensitivity of List-SP is higher than SP and CaSPIAN. This is a consequence of the fact that List-SP is specifically designed to increase the TP rate of SP, which consequently increases sensitivity. On the other hand, CaSPIAN uses an F-test to reduce the false-positive rate of List-SP, which may in turn reduce the sensitivity due to an increase in the false-negative rate. The sensitivity of CaSPIAN improves given more time-points (i.e., larger $m$), as expected. Another important observation is that the sensitivity of List-SP improves when increasing $k$. This is due to the iterative structure of List-SP: for any $K\geq 2$, the output of List-SP for $k=K$ is included in the output of List-SP for $k=K+1$. When $m$ is sufficiently large (roughly $m\geq 14$) and $k>d_{max}$, the sensitivity of CaSPIAN is not significantly affected by the choice of $k$. Therefore, one can choose $k$ based on a rough estimate of the largest in-degree of the network.

Figures S2 and S6 in Appendix S1 illustrate the average precision of our algorithms with respect to $m$. The precision of CaSPIAN is significantly better than the precision of List-SP and SP, even when a relatively large value of $P_F=10^{-2}$ is chosen. The precision of CaSPIAN increases as $P_F$ decreases, which is to be expected as a smaller value of $P_F$ implies a stricter condition for the F-test. However, choosing a very small value for $P_F$ decreases the sensitivity of CaSPIAN. Similar to the case of sensitivity analysis, it may be observed that for sufficiently large values of $m$ (e.g., $m\geq 14$) the precision of CaSPIAN is not significantly affected by the choice of $k$.

Figures S3 and S7 in Appendix S1 illustrate the average accuracy, while Figures S4 and S8 in Appendix S1 show the average $F$-measure of our algorithms with respect to $m$. The accuracy and $F$-measure of CaSPIAN are significantly better than those of SP and List-SP, and the accuracy and $F$-measure of CaSPIAN are not significantly affected by the choice of $k$ provided that $m$ is sufficiently large.

**The under-sampled regime with noise-free expressions.** We also evaluated the performance of the CaSPIAN algorithm when the number of genes is larger than the number of available time-points. Since CaSPIAN is based on a compressive sensing approach, in principle one may use this method when the number of genes is significantly larger than the number of measurements. The impediment of direct use of CaSPIAN on large biological networks is associated with noise in the measurements and the sampling irregularities of experimental data.

We evaluated CaSPIAN and related algorithms on 500 randomly generated networks comprising $n=100$ genes, and with expression profiles of size $m=75$, as described in the supporting information. Figures S9 and S11 in Appendix S1 show the average sensitivity, precision, accuracy and $F$-measure of SP, List-SP and CaSPIAN versus the sparsity level $k$, each corresponding to a different degree distribution. The standard deviations corresponding to these measures are provided in separate plots, as shown in Figures S10 and S12 in Appendix S1.

From Figure S9 in Appendix S1, we see that in the absence of noise, List-SP and CaSPIAN outperform SP; in particular, for $k\geq 5$, the sensitivity of List-SP and CaSPIAN is at least 85%, and it grows with $k$. The high sensitivities of these two algorithms imply that at least 85% of the directed edges in the network were detected for $k\geq 5$. Note that the maximum in-degree of the simulated network was set to $d_{max}=5$. As $k$ increases beyond $k=d_{max}$, the number of false-negatives decreases, and consequently the sensitivity increases. One can see that both SP and List-SP have low precision when compared to the CaSPIAN algorithm, as the latter applies a Granger-causality test that significantly reduces the number of false-positives. One measure that can capture the joint effects of sensitivity and precision is the $F$-measure, which we analyzed in addition to sensitivity and precision. As for the case of the oversampled regime, a rough estimate of $d_{\max}$ can be used as a bound on $k$ even in the under-sampled regime without significantly affecting the $F$-measure of CaSPIAN. Similar results are shown in Figure S11 in Appendix S1. In particular, all values of $k\in\{4,5,6,7\}$ result in nearly identical $F$-measures for the CaSPIAN algorithm in the under-sampled regime. Note that the networks randomly generated in Figure S11 in Appendix S1 have a maximum in-degree of $d_{max}=6$.

The results discussed up to this point were based on uniform sampling strategies for the synthetic network, i.e., on datasets obtained by measuring gene expressions at uniformly spaced times. Since in some available datasets measurements were generated using non-uniformly spaced time-points, we examined how the performance of our greedy algorithms is affected by the sampling strategy. For this purpose, 500 networks of $n=100$ genes were generated randomly as described in the supporting information section. For each network, 225 time-points were generated; subsequently, for each network, 75 measurements out of the existing 225 measurements were chosen uniformly at random. This is equivalent to a nonuniform sampling of the original time-points with an average rate of 1/3. Figures S13 and S14 in Appendix S1 show the effect of nonuniform sampling. As can be seen in these figures, the performance of all the algorithms significantly deteriorates, which suggests that uniform sampling should be practiced in place of nonuniform sampling whenever possible, given that in the model delays are assumed to be regular.

**The under-sampled and over-sampled regimes with noisy expressions.** In order to evaluate the performance of the proposed algorithms in the presence of noise, we considered 200 randomly generated networks of $n=100$ genes. White Gaussian noise with a variance equal to 5% of the average signal power was added to *all* gene expression profiles. The construction of the underlying networks is discussed in Appendix S1.

Figures S15–S22 in Appendix S1 illustrate the mean and standard deviation of the proposed algorithms for different values of $k\in\{4,5,6,7\}$ versus the number of time points $m$; a range of $20\leq m\leq 180$ was considered for the number of time points to include both the under-sampled and over-sampled regimes. As can be seen in these figures, List-SP has a higher sensitivity compared to that of the other algorithms; however, CaSPIAN with $P_F=10^{-2}$ closely follows the sensitivity of List-SP. On the other

hand, CaSPIAN with $P_F = 10^{-9}$ and $P_F = 10^{-16}$ has much smaller sensitivity compared to List-SP.

The opposite behavior can be observed with respect to the precision of these algorithms: the precision of CaSPIAN with $P_F = 10^{-9}$ and $P_F = 10^{-16}$ is significantly better than the precision of CaSPIAN with $P_F = 10^{-2}$, SP, and List-SP. Figures S15–S22 in Appendix S1 show that CaSPIAN has a high accuracy, and the accuracy is not significantly affected by the choice of $P_F$. In addition, it can be observed that CaSPIAN with $P_F = 10^{-2}$ outperforms SP and List-SP with regards to the $F$-measure in both the under-sampled (i.e. $m < 100$) and the over-sampled ($m > 100$) regimes. On the other hand, for $P_F = 10^{-9}$ and $P_F = 10^{-16}$, the $F$-measure increases significantly as $m$ increases.

Next, we focus on the performance of these algorithms in the under-sampled regime in the presence of white Gaussian noise. Figures S23 and S24 in Appendix S1 demonstrate the mean and standard deviation of the sensitivity, precision, accuracy and $F$-measure of SP, List-SP, and CaSPIAN as a function of the ratio of the noise variance and the average signal power. As expected, the presence of noise deteriorates the performance of all the algorithms; in particular, since CaSPIAN rejects some recovered genes through an F-test, in the presence of noise this may cause some correctly identified edges recovered by List-SP to be rejected. This effect is more prominent for smaller values of $P_F$. Although the presence of noise increases the number of false-negatives in CaSPIAN, it is important to note that the *precision* of CaSPIAN remains high. In particular, for $P_F = 10^{-9}$ and $P_F = 10^{-16}$, a precision higher than 90% can be maintained for the chosen noise energy.

These findings demonstrate that changing the significance value $P_F$ results in a trade-off between sensitivity and precision when noise is present in the system: higher sensitivity may be achieved by choosing a larger value for $P_F$ – this is of importance for applications where reducing the false-negative rate is more important than reducing the false-positive rate. On the other hand, by choosing small values for $P_F$, one may be able to detect true-positives with very high reliability, while missing some existing edges; such choices of $P_F$ are particularly useful when finding *correct edges* is more important than finding *all edges*. On the other hand, when both sensitivity and precision are equally important, one should use the $F$-measure to evaluate the performance. In this case, $P_F = 10^{-2}$ outperforms other choices of $P_F$, which confirms that a range $0.01 \leq P_F \leq 0.05$ is the most appropriate choice in noisy scenarios, as previously suggested in the literature.

One can also take advantage of the edges recovered using different values of $P_F$ by assigning a confidence level to each edge depending on the $P_F$ value used to recover that edge. For example, when using the three values chosen for $P_F$ in this section, the edges recovered using $P_F = 10^{-16}$ have the highest confidence level; on the other hand, the edges that were recovered for the first time using $P_F = 10^{-9}$ (i.e. they were not present among the edges recovered using $P_F = 10^{-16}$) have the second highest confidence level, while the edges recovered for the first time using $P_F = 10^{-2}$ have the lowest confidence level among the others.

## Comparison of CaSPIAN with other reverse-engineering algorithms

As discussed at the beginning of this section, we used in vivo networks to compare the performance of CaSPIAN with other reverse-engineering algorithms. The results are discussed in what follows.

**The IRMA network in *Saccharomyces cerevisiae*.** We focus next on a network model that we believe to be a benchmark standard, given that it shares many features of synthetic networks – such as precise design parameters and controllability – while being part of a biological network in a living organism. The network in question is the IRMA (in vivo reverse-engineering and modeling assessment) network, a synthetic network of five genes, *CBF1*, *GAL4*, *SWI5*, *GAL80*, *ASH1*, embedded in *Saccharomyces cerevisiae* (yeast). The IRMA network has a fixed topology, and is constructed in such a way that its constituent genes are not regulated by other yeast genes. This network was introduced in [48].

For our analysis, we used the time-series gene expressions of "switch-on" experiments (shifting cells from glucose to galactose), measured via quantitative real-time PCR (q-PCR) every 20 minutes for up to 5 hours, within five experiments. The performance of two additional algorithms utilizing sparsity and causality, TSNI [56] and BANJO [57], were evaluated in [8] using the same dataset. TSNI is an algorithm based on modeling networks via ordinary differential equations (ODE), while BANJO is an algorithm based on Bayesian networks. The reconstructed networks corresponding to these algorithms are depicted in Figure 1.

We first ran CaSPIAN (with $P_F = 0.05$, $D = 2$, $k = 3$) on the time-series dataset, which consisted of the *average* of the gene expressions of the five experiments. As can be seen in Figure 1, CaSPIAN ($P = 0.67$, $S = 0.5$, $F = 0.57$) outperforms BANJO ($P = 0.3$, $S = 0.25$, 0.27); however, its precision and $F$-measure are not as good as that of the ODE-based algorithm: TSNI ($P = 0.8$, $S = 0.5$, $F = 0.62$). We next ran CaSPIAN using the combination of the gene expressions of the five experiments, employing the method described in the section "Forming gene expression profiles". As can be seen in Figure 1, CaSPIAN with $D = 2$ and $k = 3$ matches the performance of TSNI, while it outperforms TSNI with $D = 3$ and $k = 3$ ($P = 0.83$, $S = 0.625$, $F = 0.71$). This shows that using data averaged over different experiments causes a significant loss in the available information and deteriorates the performance of CaSPIAN. On the other hand, if one uses all the expressions to form profiles as previously described, the performance of CaSPIAN significantly improves. Note that CaSPIAN outperforms TSNI, in spite of the fact that TSNI uses a cumbersome and complicated procedure including a cubic smoothing spline filter and Principle Component Analysis (PCA) for dimensionality reduction.

As a concluding remark, we point out the interesting fact that the union of edges discovered by CaSPIAN and TSNI includes 7 out of 8 correct edges of the IRMA network and two false positives. This suggests an approach that we believe to be important for future investigation: fusing the outputs of a number of methods tuned to operate under different modeling assumptions. The topic of network fusion is beyond the scope of this paper.

**A HeLa cell line network.** In order to compare the performance of CaSPIAN with that of other algorithms on experimental biological data, we used a network consisting of 9 HeLa cell genes. The network corresponding to this set of genes was reported in [58] and was used in the literature as a benchmark for evaluating the performance of different algorithms such as CNET [58], Group Lasso (grpLasso) [59] and truncating adaptive Lasso (TAlasso) [2]. CNET is an algorithm for reverse engineering of causal gene interactions using microarray time series data; this algorithm tries to find the "best" directed graph among all the possible graphs, where the quality of a candidate graph is determined using a specific scoring function. On the other hand, grpLasso and TAlasso are penalty-based algorithms that rely on $\ell_1$ minimization. The grpLasso algorithm infers causal interactions

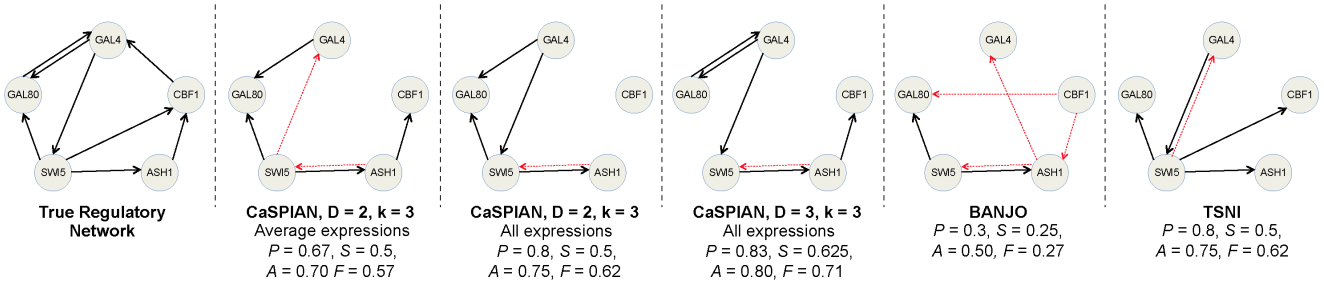


**Figure 1. The gene regulatory network corresponding to the IRMA network, obtained using different algorithms.** Solid arrows denote true-positives and dashed arrows denote false-positives. True-negatives and false-negatives are not depicted in the figures in order to avoid cluttering; however, they can be easily obtained by comparing the true regulatory network and the inferred networks. Precision is denoted by $P$, sensitivity by $S$, accuracy by $A$, and the $F$-measure by $F$.
doi:10.1371/journal.pone.0090781.g001

by evaluating the effect of different gene expressions *averaged* over different time-lags and therefore does not take advantage of all the information available in the time-series data. On the other hand, TAlasso does not suffer from the loss of information in grpLasso and employs an additional truncation method to simplify the model. However, as discussed in earlier sections, it suffers from the shortcomings intrinsic to $\ell_1$ minimization based Lasso approaches.

We used the expression data of HeLa cells from the third experiment performed in [60], consisting of 47 time-points. For each algorithm, we used the reconstructed graphs reported in the corresponding papers (as pointed out in Footnote 1 of [2], there appear to be some errors in the reported results of [59]; therefore, we used the corrected results of [2]). Since in [2] three time lags were used for TAlasso, we also used three time-lags in CaSPIAN (or equivalently, we set $D=4$). Since the true in-degree of the network is 5, we used $k=7$ as a rough upper bound. In addition, we set $P_F=0.05$ as a widely accepted significance value for $F$-tests. The results are shown in Figure 2, along with the precision, sensitivity and accuracy for each algorithm.

As can be seen in this figure, CaSPIAN outperforms all the other algorithms with respect to all three parameters: precision, sensitivity and $F$-measure. Note that this is in spite of the fact that CNET is a search-based algorithm and performs an extensive search to find the best graph. As a result, not only does CaSPIAN provide higher efficiency than the other algorithms, but it also outperforms them in terms of recovering the gene regulatory network. It is again interesting to point out that CaSPIAN and CNET tend to discover almost disjoint sets of true-positives, which suggests combining the methods as described for the IRMA network.

**The SOS genes in *E. coli*.** Our analysis of the IRMA and HeLA networks illustrated the performance of CaSPIAN and other causal inference algorithms for the case when the number of time-points was larger than the number of genes (corresponding to the over-sampled regime). As CaSPIAN is based on compressive sensing methods which allow for inference in the under-sampled regime, and as most inference problems operate in the under-sampled regime, it is instructive to test the performance of the method on a larger network. The performance of CaSPIAN is contrasted to that of the Lasso and truncating Lasso (Tlasso) [2].

We used the gene expression data of *E. coli* from the Many Microbe Microarrays Database (M3D) [61]. To evaluate the performance of CaSPIAN, we focused on the SOS subnetwork. The main documented genes in this subnetwork are *dinI*, *lexA*, *recA*, *recF*, *rpoD*, *rpoH*, *rpoS*, *ssb*, *umuC/D*, as described in [49]. We denote these genes by $\mathcal{G}_{SOS}$. The known links in the SOS network of *E. coli* are available in Table S4 of [22]; for completeness, and as a reference for our comparisons, we provided this information in Appendix S2. The M3D database contained the expression levels of 4292 genes from $m_\Delta=22$ experiments with at least $\Delta=3$ time-points. The number of time-points in these 22 experiments was at least 3 and at most 5. Using the method for forming gene expression lists outlined in the previous section, we combined the data of the 22 experiments producing profiles of length $m=72$. In order to evaluate the performance of these algorithms in the *under-sampled* regime, we randomly selected a set of 90 genes from the 4283 genes other than $\mathcal{G}_{SOS}$ and formed a set of $n=100$ genes as

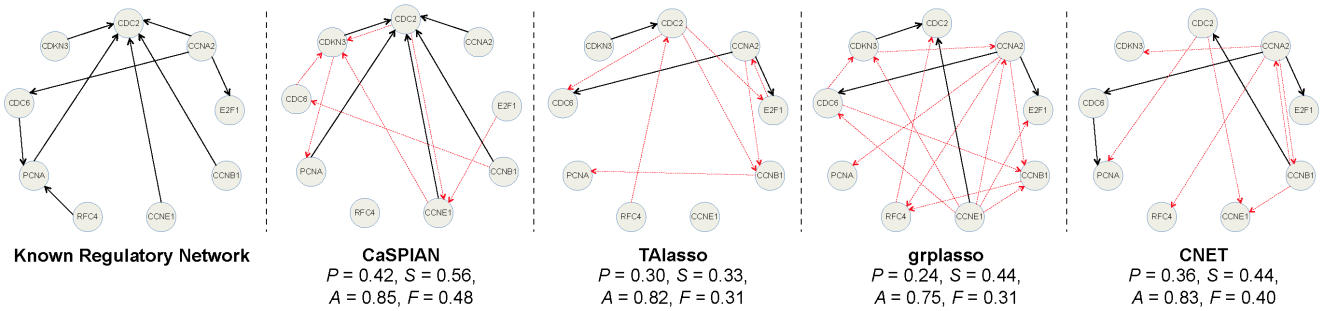


**Figure 2. A gene regulatory network of HeLa cell genes, reconstructed using different causal inference algorithms.** Solid arrows denote true-positives and dashed arrows denote false-positives. True-negatives and false-negatives are not depicted in the figures to avoid cluttering; however, they can be easily obtained by comparing the known regulatory network and the inferred network. Precision is denoted by $P$, sensitivity by $S$, accuracy by $A$ and the $F$-measure by $F$.
doi:10.1371/journal.pone.0090781.g002

the union of this set and $\mathcal{G}_{SOS}$. We repeated this procedure 10 times: we ran List-SP, CaSPIAN, Lasso, and TLasso on 10 sets, each containing $n=100$ genes, formed as the union of $\mathcal{G}_{SOS}$ and a set of 90 other genes chosen independently and uniformly at random. Executing each algorithm more than once was motivated by the idea of introducing the multiplicity ratio (MR) as a measure of confidence for the recovered links. MR represents the number of runs of the algorithm that produced a given correct edge, divided by the total number of runs of the algorithm.

We ran List-SP (Algorithm 1) for $k=4,5,6$ using the gene expression profiles formed according to the method of the previous section, and with $D=2$. Appendix S2 provides the tables of MRs for the recovered directed edges corresponding to $k=4$, $k=5$, and $k=6$. Figure 3 shows the network reconstructed using List-SP (with $k=5$ and MR at least 0.2). Comparing this figure with the results in [49] reveals that List-SP is capable of correctly identifying 19 edges in the SOS network (note that there exists a number of feedback loops between two vertices that are counted as two separate edges). However, List-SP misses some edges in the network due to the choice of $k$ and noise in the expression data. Using a larger value of $k$ for List-SP allows us to find more existing links in the network, but the number of false-positives increases as well, which is expected (see tables in Appendix S2). We ran CaSPIAN for 5 different values of $P_F$ on the generated 10 sets of genes. The corresponding MR values of the links found using CaSPIAN are shown in tables in Appendix S2. The reconstructed network for $P_F=10^{-2}$ is shown in Figure 3. It is important to note that *all* the links found using CaSPIAN are correct links (i.e., the regulatory relationships have been reported in the literature).

We ran Lasso and Tlasso on the same set of genes, both of which are combinations of Granger causality with different versions of a Lasso penalty (the codes for these algorithms are available at http://www.biostat.washington.edu/~ashojaie). We used the *error based method* for choosing the regularization coefficient. In [2], the parameter $\alpha$ was chosen between 0.05 and 0.2, and it was stated that different values of $\alpha$ do not affect the performance of the algorithm significantly. As a result, we picked $\alpha=0.1$ (the default value in the algorithm) and $\alpha=0.2$. For a detailed description of the error based method and the definition of $\alpha$, see the original paper [2]. Default values were used for all other parameters in these two algorithms. Lasso was unable to find even a single correct link between the SOS genes; similarly, Tlasso failed to find any links among the genes for $\alpha=0.1$. On the other hand, for $\alpha=0.2$, it correctly identified a link from *umuC/D* to *lexA* with MR equal to 0.9; however, no other link was found using this method. Comparing these results with List-SP and CaSPIAN reveals that both of these algorithms outperform Lasso and Tlasso in the undersampled regime. As another illustrative example, CaSPIAN with $P_F=10^{-2}$ found 13 true positives and 0 false-positives, while Tlasso only found one true positive.

## CaSPIAN with scaffolding subnetworks

We examined the performance of CaSPIAN assuming that some directed edges in the network are known in advance (Algorithm 3). In particular, we evaluated the improvement/change in the performance of CaSPIAN given that a *correct* subnetwork is known. In addition, we assessed the degradation/change in the performance of CaSPIAN given that an *incorrect* subnetwork is assumed. Since the exact regulatory network of IRMA is known, in our study we mainly focus on the IRMA network.

**Application to the IRMA network.** In order to evaluate the performance of Algorithm 3, we used the averaged expressions of the genes in the IRMA network corresponding to the "switch-on" experiments. We applied Algorithm 3 with parameters $D=2$ and $k=3$ to different known subnetworks consisting of 1, 2, and 3 correct edges. Note that we used averaged data and suboptimal CaSPIAN parameters, as the optimal performance of CaSPIAN may not be further improved for the IRMA network given prior information. The results are illustrated in Figure 4. As can be seen in this figure, knowing even one correct edge can improve the performance of CaSPIAN significantly, although in these and most other tests we performed, the number of false positives remained unchanged. As an illustrative example, CaSPIAN without side-information recovers a false-positive edge from *SWI5* to *GAL4*. However, if the information that an edge exists from *CBF1* to *GAL4* is provided a priori, CaSPIAN correctly concludes that no edge exists from *SWI5* to *GAL4*.

In order to address the second question, we considered 9 different incorrect edges as a given subnetwork, and applied Algorithm 3 ($D=2$ and $k=3$) to averaged IRMA expressions. As before, we used suboptimal performance parameters in order to be able to compare the drawbacks/advantages of using incorrect/correct side information, as compared to using no side information, given one and the same dataset. The results are shown in Figure 5. When an incorrect subnetwork is provided, Algorithm 3 forces CaSPIAN to build a network around this structure. Still, when the wrong side-information represents an edge outside the true network, most correctly identified edges remain accurate, but a large number of new false-positives arise.

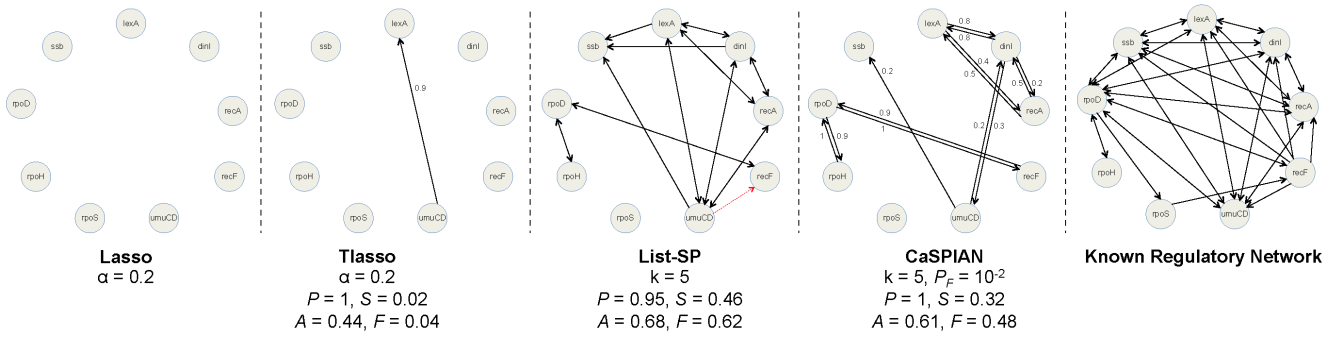


**Figure 3. The SOS network reconstructed using Lasso, TLasso, Algorithm 1 (List-SP) and CaSPIAN.** Black solid arrows correspond to true positives and red dashed arrows correspond to false-positives. The numbers above edges describe their multiplicity ratios (MRs); in order to avoid cluttering, we did not plot the MRs for the results of Algorithm 1. Note that only links with MR at least 0.2 are shown. Precision is denoted by $P$, sensitivity by $S$, accuracy by $A$ and the $F$-measure by $F$.
doi:10.1371/journal.pone.0090781.g003

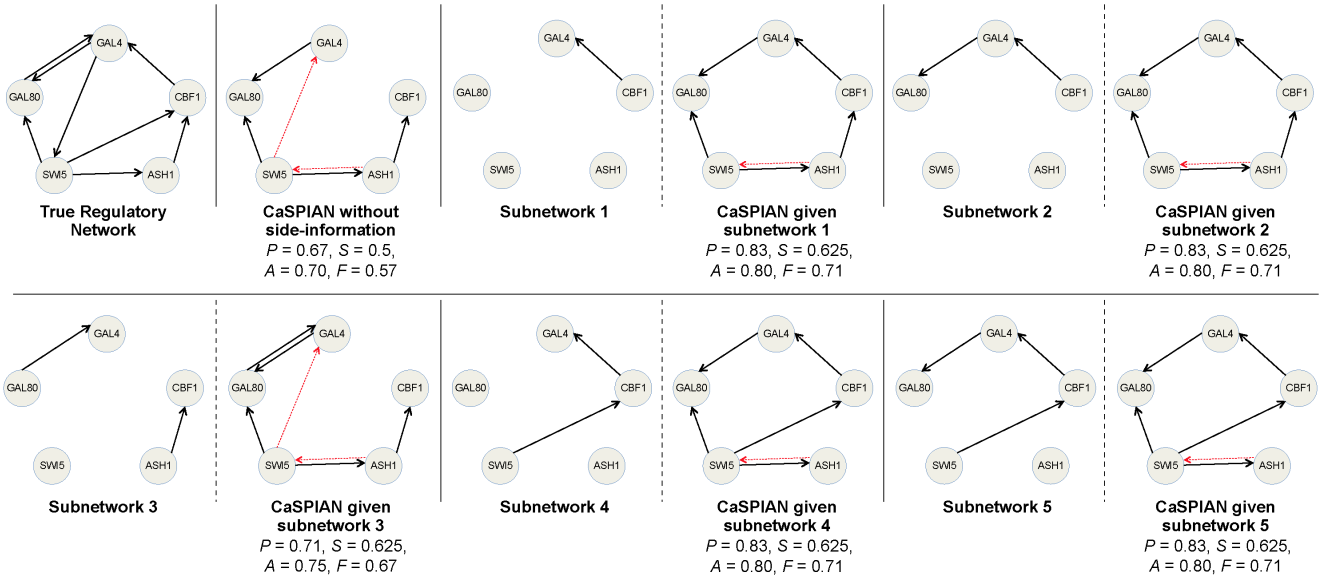


**Figure 4. The network corresponding to IRMA obtained by applying Algorithm 3 to various subnetworks.** Parameters of this algorithm were chosen as $D=2$ and $k=3$ and average gene expressions were used. Solid arrows denote true-positives and dashed arrows denote false-positives. Precision is denoted by $P$, sensitivity by $S$, accuracy by $A$, and the $F$-measure by $F$.
doi:10.1371/journal.pone.0090781.g004

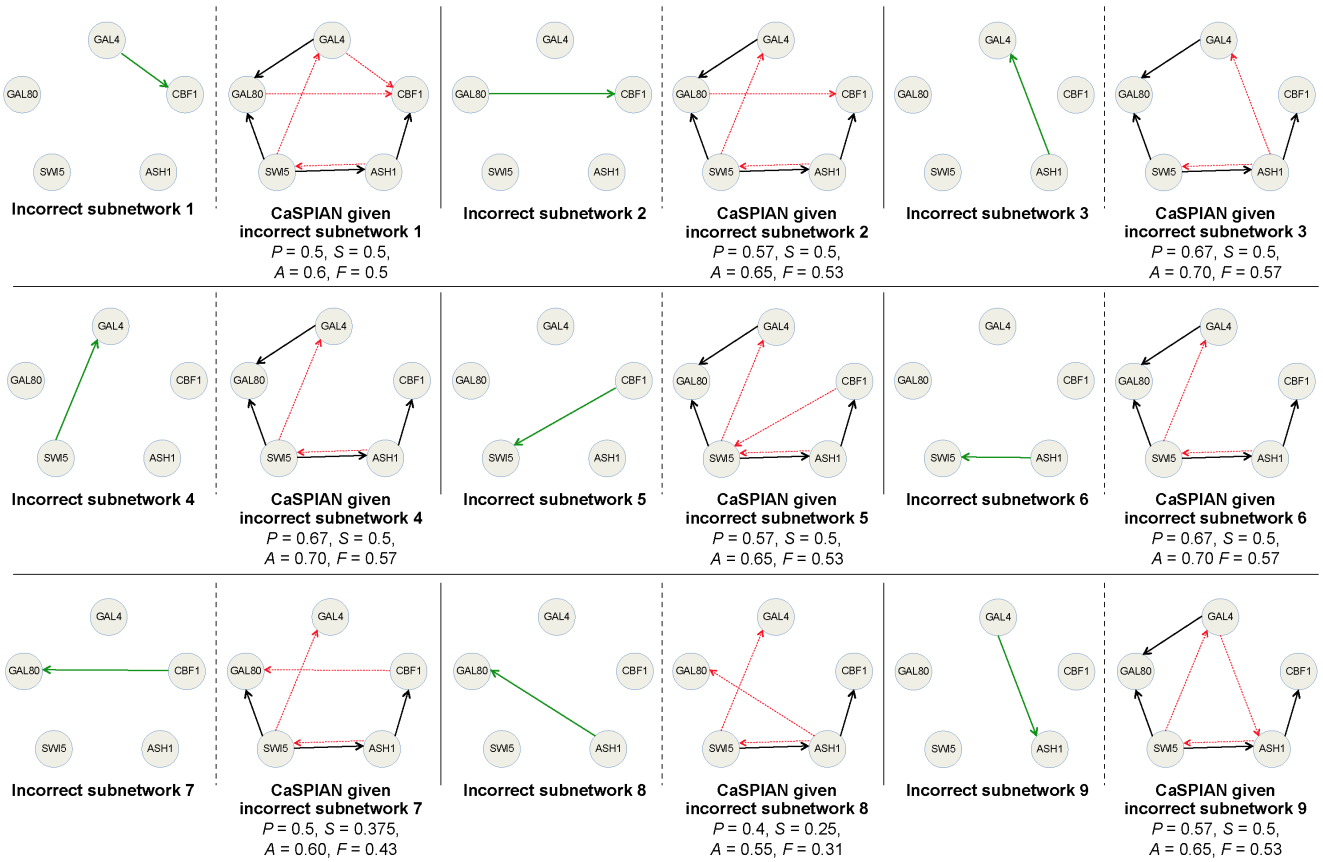


**Figure 5. The network corresponding to IRMA obtained by applying Algorithm 3 to various incorrect subnetworks.** Parameters of this algorithm were chosen as $D=2$ and $k=3$ and average gene expressions were used. Solid arrows denote true-positives and dashed arrows denote false-positives. Green solid arrows correspond to the incorrect edges in the subnetwork. Precision is denoted by $P$, sensitivity is denoted by $S$, accuracy by $A$, and the $F$-measure by $F$.
doi:10.1371/journal.pone.0090781.g005

**Application to HeLa cell genes.** Since the exact regulatory network of *HeLa* is not completely known, we only address the first question raised in the introduction of the section.

Figure 6 shows the results of Algorithm 3 applied to different known subnetworks consisting of 1, 2, 3, and 4 edges. As can be seen, knowing a subnetwork consisting of 4 edges suffices to achieve sensitivity equal to $S=1$. Also, it is important to note that the number of false-positive edges corresponding to the genes claimed to influence *CDKN3* cannot be reduced, since the in-degree of this gene in the known regulatory network is 0. Therefore, the edges found using CaSPIAN for this target gene do not change given the different subnetworks.

## Conclusions

The CaSPIAN approach for network inference represents a new attempt to connect the fields of compressive sensing, causal inference and bioinformatics. Compressive sensing (CS) is a technique for sampling and recovering sparse signals using optimization and/or greedy approaches. In spite of its widespread applications in different areas such as signal processing, image and video processing, communications, etc., its utility has not yet been fully explored in the field of bioinformatics and gene network inference. Although [42] and [43] employ the CS framework to infer gene networks, there are crucial differences between CaSPIAN and the algorithms discussed in these papers. First, unlike CaSPIAN, none of these algorithms are capable of inferring causal (i.e. directed) edges in the gene network. Second, CaSPIAN is an unsupervised learning algorithm that uses CS as the core of its approach; however, [43] introduced an approach which employs CS only as a preprocessing step and combines the obtained results with extensive prior biological information to infer gene networks using a supervised learning algorithm performed by Adaboost. However, due to the susceptibility of Adaboost to random classification noise [45], application of such booster algorithms in biological network inference is limited. Finally, CaSPIAN employs a low-complexity greedy CS approach, which significantly reduces its computational complexity compared to the $\ell_1$ minimization approach used in [43].

Another line of work explored in the literature integrates the sparsity of gene networks using some version of the LASSO penalty [2,36–38]. There are two main difficulties associated with the LASSO approach. First, such methods require performing high dimensional $\ell_1$ optimization, which has a high computational complexity compared to greedy CS-based algorithms such as CaSPIAN. In addition, in order to enforce the sparsity criterion, these approaches require a proper choice for the coefficient of the regularization term. Unlike CaSPIAN, in which the parameter $k$ is directly related to the sparsity of the network (i.e. the in degree of the nodes), the regularization coefficient in the LASSO penalty does not directly correspond to the sparsity of the network and therefore is usually chosen using some heuristics or by performing optimization, which increase the complexity of these algorithms without providing provable performance guarantees. On the other hand, by using CaSPIAN, one is able to directly employ information about the degree of the nodes in the network; as we showed in the results, if such information is not available, in most cases an upper bound on the in-degree is sufficient.

CaSPIAN is a reverse engineering algorithm that is based on the list-SP algorithm: a low-complexity greedy search method, which scans the expression profiles for low-dimensional subspaces. Although the computational advantage of the method may not be of great relevance for current small network inference problems, with the ever-increasing volumes of data, this issue will become important in the near future. Furthermore, CaSPIAN does not require fine parameter tuning or supervised learning methods as was shown using both synthetic and real biological data. In addition, in Algorithm 3 we provided a version of CaSPIAN that incorporates biological side-information in the form of scaffolding subnetworks of accurate edges. Our analysis shows that correct edges may significantly improve the performance of CaSPIAN. Given that wrong side-information may severely deteriorate the performance of the inference methods, it may be advisable to use prior information with caution - for example, only if the links were experimentally verified using different techniques by at least several different research labs.

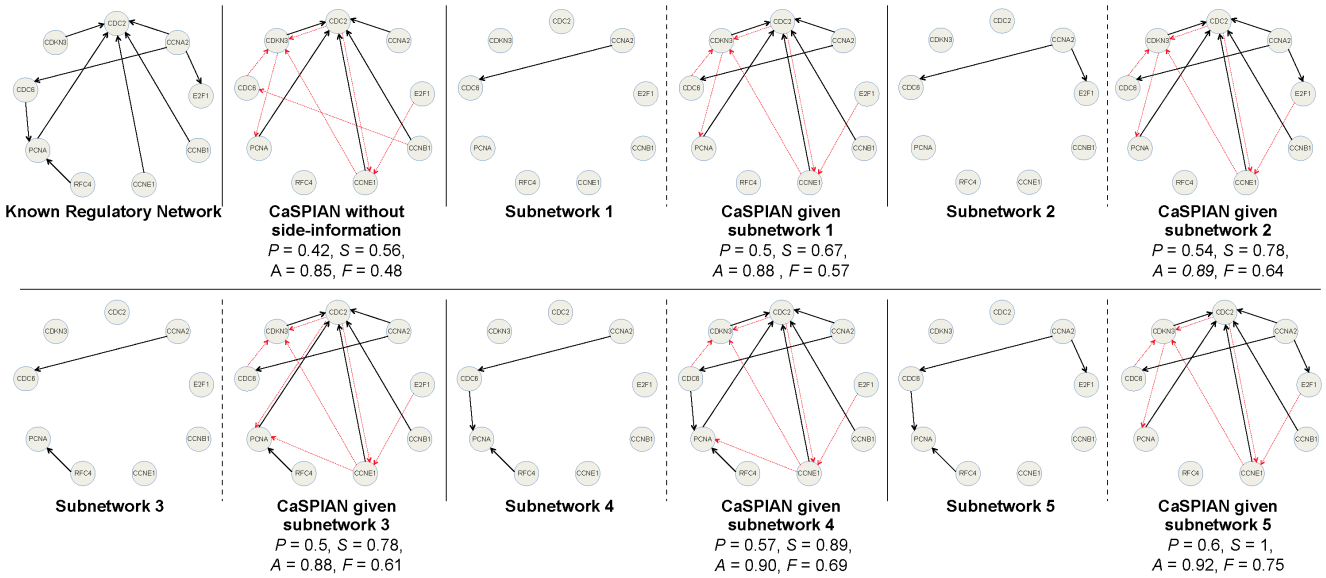


**Figure 6. The HeLA network inferred by CaSPIAN with D = 4 and k = 7.** Solid arrows denote true-positives and dashed arrows denote false-positives. Precision is denoted by $P$, sensitivity is denoted by $S$, accuracy by $A$, and the $F$-measure by $F$.
doi:10.1371/journal.pone.0090781.g006

We also evaluated the influence of different parameters such as network topology, degree distribution, size, number of measurements, sampling method, noise, and the value of $k$ and $P_F$ on the performance of the proposed algorithms using synthetic (simulated) networks. In particular, we showed that there exists a tradeoff between the sensitivity and precision of CaSPIAN that is mainly controlled by the choice of $P_F$. As a result, in applications in which finding a few highly reliable edges is of main interest, one can choose the value of $P_F$ to be small; however, in applications where both precision and sensitivity are of interest, a value of $P_F$ between 0.05 and 0.01 is more appropriate. Another approach to take advantage of this tradeoff is to assign a reliability score to each edge based on the smallest value of $P_F$ for which CaSPIAN recovered that edge.

In addition, we compared the performance of CaSPIAN with that of other algorithms using real biological (in vivo) networks and showed that in many cases CaSPIAN outperforms other algorithms in spite of their extensive use of resources and side-information and their high complexity. Although we provided extensive comparisons with other models and illustrated that under almost all performance criteria used by the reverse engineering community, CaSPIAN outperforms these methods, we cannot argue that there exists one "optimal approach" for all problems. As an example, we did not test the performance criteria used in the DREAM2 challenge, where teams were asked to provide rank-ordered list of edges believed to exist in the network, according to their reliability. Those lists were truncated to top-$T$ candidates, for different values of $T$, and tested for accuracy and precision. In a different approach, precision and accuracy were calculated at the point of the $n$-th correct prediction. Each performance criteria is bound to give slightly, if not vastly different answers. It would therefore be of interest to investigate optimal fusion strategies of different methods based on different evaluation criteria in order to recognize the strengths and weaknesses of current methods and fully utilize them in the reverse engineering process. For example, combining CaSPIAN with reverse engineering algorithms based on a completely different framework, such as Bayesian networks or differential equations, may result in a method that can outperform each individual algorithm in a noisy, non uniformly under-sampled regime.

### Availability

The implementation of these algorithms in Matlab will be offered upon request. Please contact the following email address: emad2@illinois.edu.

## Supporting Information

**Appendix S1 CaSPIAN applied to synthetic networks.**
(PDF)

**Appendix S2 Tables of multiplicity ratios (MRs) for the *E. coli* SOS network.**
(PDF)

## Acknowledgments

The authors would like to thank Mo Deng for his contribution to the initial development of some of the ideas and running some preliminary simulations. Those preliminary results were presented at the Statistical Signal Processing (SSP) Workshop, Ann Arbor, 2012 [62].

## Author Contributions

Conceived and designed the experiments: AE OM. Performed the experiments: AE. Analyzed the data: AE OM. Contributed reagents/materials/analysis tools: AE. Wrote the paper: AE OM.

## References

1. Rao A, Hero AO, States DJ, Engel JD (2008) Using directed information to build biologically relevant inuence networks. J Bioinfo Comput Biol 6: 493–519.
2. Shojaie A, Michailidis G (2010) Discovering graphical granger causality using the truncating lasso penalty. Bioinformatics 26: i517–i523.
3. Stolovitzky G, Prill RJ, Califano A (2009) Lessons from the dream2 challenges. Annals of the New York Academy of Sciences 1158: 159–195.
4. Sima C, Hua J, Jung S (2009) Inference of gene regulatory networks using time-series data: a survey. Curr Genomics 10: 416–429.
5. Pearl J (2000) Causality: models, reasoning and inference, volume 29. Cambridge Univ Press.
6. Xu M, Zhu M, Zhang LX (2008) A stable iterative method for refining discriminative gene clusters. BMC Genomics 9(Suppl 2): S18.
7. Friedman N, Linial M, Nachman I, Peer D (2000) Using bayesian network to analyze expression data. J Comput Biol 7: 601–620.
8. de Jong H (2002) Modeling and simulation of genetic regulatory systems: a literature review. J Comput Biol 9: 67–103.
9. Husmeier D (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. Bioinformatics 19: 2271–2282.
10. Perrin BE, Ralaivola L, Mazurie A, Bottani S, Mallet J, et al. (2003) Gene networks inference using dynamic bayesian networks. Bioinformatics 19: ii138–ii148.
11. Zou M, Conzen SD (2005) A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data. Bioinformatics 21: 71–79.
12. Kauffman S (1969) Homeostasis and differentiation in random genetic control networks. Nature 224: 177–178.
13. Liu W, Lähdesmäk H, Dougherty E, Shmulevich I (2008) Inference of boolean networks using sensitivity regularization. EURASIP J Bioinform Syst Biol 2008: 780541.
14. Chen T, He HL, Church GM (1999) Modeling gene expression with differential equations. Pac Symp Biocomput 4: 29–40.
15. Samad HE, Khammash M, Petzold L, Gillespie D (2005) Stochastic modelling of gene regulatory networks. Int J Robust Nonlinear Control 15: 691–711.
16. Chen BS, Chang CH, Wang YC, Wu CH, Lee HC (2011) Robust model matching design methodology for a stochastic synthetic gene network. Math Biosci 230: 23–36.
17. Ruklisa D, Brazma A, Viksna J (2005) Reconstruction of gene regulatory networks under the finite state linear model. Genome inform 16: 225–236.
18. Laubenbacher R, Stigler B (2004) A computational algebra approach to the reverse engineering of gene regulatory networks. J Theor Biol 229: 523–537.
19. Dingel J, Milenkovic O (2009) List-decoding methods for inferring polynomials in finite dynamical gene network models. Bioinformatics 25: 1686–1693.
20. Butte A, Kohane I (2000) Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. Pac Symp Biocomp 5: 415–426.
21. Margolin A, Nemenman I, Basso K, Wiggins C, Stolovitzky G, et al. (2006) Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics 7: S7.
22. Dougherty J, Tabus I, Astola J (2008) Inference of gene regulatory networks based on a universal minimum description length. EURASIP J Bioinform Syst Biol 8: 1–11.
23. Meyer PE, Lafitte F, Bontempi G (2008) minet: A r/bioconductor package for inferring large transcriptional networks using mutual information. BMC Bioinformatics 9: 461.
24. Liang K, Wang X (2008) Gene regulatory network reconstruction using conditional mutual information. EURASIP J Bioinform Syst Biol 2008: 253894.
25. Zhao W, Serpendin E, R DE (2008) Inferring connectivity of genetic regulatory networks using information-theoretic criteria. IEEE/ACM Trans Comput Biol Bioinform 5: 262–274.
26. Watkinson J, Liang KC, Wang X, Zheng T, Anastassiou D (2009) Inference of regulatory gene interactions from expression data using three-way mutual information. Ann N Y Acad Sci 1158: 302–313.
27. Altay G, Emmert-Streib F (2011) Structural inuence of gene networks on their inference: analysis of c3net. Biol Direct 6: 31.
28. Werhli A, Grzegorczyk M, Husmeier D (2006) Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. Bioinformatics 22: 2523–2531.
29. Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D (2007) How to infer gene networks from expression profiles. Mol Syst Biol 3: 78.
30. Margolin A, Califano A (2007) Theory and limitations of genetic network inference from microarray data. Ann N Y Acad Sci 1115: 51–72.

31. Olsen C, Meyer P, Bontempi G (2009) On the impact of entropy estimator in transcriptional regulatory network inference. EURASIP J Bioinform Syst Biol 2009: 308959.
32. de Smet K Rand Marchal (2010) Advantages and limitations of current network inference methods. Nat Rev Microbiol 8: 717–729.
33. Penfold CA, Wild DL (2011) How to infer gene networks from expression profiles, revisited. Interface Focus 1: 857–870.
34. Emmert-Streib F, Glazko G, Gökmen A, De Matos Simoes R (2012) Statistical inference and reverse engineering of gene regulatory networks from observational expression data. Front Genet 3.
35. Marbach D, Costello JC, Kuffner R, Vega NM, Prill RJ, et al. (2012) Wisdom of crowds for robust gene network inference. Nature Methods 9: 796–804.
36. Fujita A, Sato JR, Garay-Malpartida HM, Yamaguchi R, Miyano S, et al. (2007) Modeling gene expression regulatory networks with the sparse vector autoregressive model. BMC Systems Biol 1: 39.
37. Mukhopadhyay N, Chatterjee S (2007) Causality and pathway search in microarray time series experiment. Bioinformatics 23: 442–449.
38. Cai X, Bazerque JA, Giannakis GB (2013) Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations. PLOS Computational Biology 9: e1003068.
39. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Statist Soc B 58: 267–288.
40. Donoho DL (2006) Compressed sensing. IEEE Trans Inf Theory 52: 1289–1306.
41. Candès EJ, Wakin MB (2008) An introduction to compressive sampling. IEEE Signal Process Magazine 25: 21–30.
42. Hang X, Dai W, Wu FX (2009) Subspace pursuit for gene profile classification. In: IEEE Int. Workshop on Genomic Signal Processing and Statistics (GENSIPS). pp. 1–4.
43. Prat Y, Fromer M, Linial N, Linial M (2007) Recovering key biological constituents through sparse representation of gene expression. Bioinformatics 27: 655–661.
44. Wright J, Ma Y, Mairal J, Sapiro G, Huang T, et al. (2010) Sparse representations for computer vision and pattern recognition. Proc IEEE 98: 1031–1044.
45. Long PM, Servedio RA (2010) Random classification noise defeats all convex potential boosters. Mach Learn 78: 287–304.
46. Ewens WJ, Grant GR (2004) Statistical Methods in Bioinformatics: An Introduction (Statistics for Biology and Health). New York: Springer Science press, 2nd edition.
47. Dai W, Milenkovic O (2009) Subspace pursuit for compressive sensing signal reconstruction. IEEE Trans Inf Theory 55: 2230–2249.
48. Cantone I, Marucci L, Iorio F, Ricci MA, Belcastro V, et al. (2009) A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. Cell 137: 172–181.
49. Gardner TS, di Bernardo D, Lorenz D, Collins JJ (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. Science 301: 102–105.
50. Van Den Berg E, Friedlander MP (2008) Probing the pareto frontier for basis pursuit solutions. SIAM J Sci Comput 31: 890–912.
51. Becker S, Bobin J, Candès E (2011) Nesta: A fast and accurate first-order method for sparse recovery. SIAM J Imaging Sci 4: 1–39.
52. Penrose R (1955) A generalized inverse for matrices. Math Proc Cambridge Philos Soc 51: 406–413.
53. Granger CWJ (1969) Investigating causal relations by econometric models and cross-spectral methods. Econometrica 37: 424–438.
54. Geweke JF (2003) Measures of conditional linear dependence and feedback between time series. J Am Stat Assoc 79: 907–915.
55. Vallabhajosyula RR, Chakravarti D, Lutfeali S, Ray A, Raval A (2009) Identifying hubs in protein interaction networks. PLoS One 4: e5344.
56. Della Gatta G, Bansal M, Ambesi-Impiombato A, Antonini D, Missero C, et al. (2008) Direct targets of the trp63 transcription factor revealed by a combination of gene expression profiling and reverse engineering. Genome Res 18: 939–948.
57. Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004) Advances to bayesian network inference for generating causal networks from observational biological data. Proc IEEE 20: 3594–3603.
58. Sambo F, Camillo BD, Toffolo G. Cnet: an algorithm for reverse engineering of causal gene networks. NETTAB2008, Varenna, Italy 2008.
59. Lozano AC, Abe N, Liu Y, Rosset S (2009) Grouped graphical granger modeling for gene expression regulatory networks discovery. Bioinformatics 25: i110.
60. Whitfield ML, Sherlock G, Saldanha AJ, Murray JI, Ball CA, et al. (2002) Identification of genes periodically expressed in the human cell cycle and their expression in tumors. Mol Biol Cell 13: 1977–2000.
61. Faith JJ, Driscoll ME, Fusaro VA, Cosgrove EJ, Hayete B, et al. (2008) Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. Nucleic Acids Res 36: D866–D870.
62. Deng M, Emad A, Milenkovic O (2012) Casual compressive sensing for gene network inference. Proceedings of IEEE Statistical Signal Processing Workshop : 696–6999.