Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 24 August 2020
Systems biology

OXFORD

Systems biology

Inferring Signaling Pathways with Probabilistic Programming

David Merrell [1,2], **Anthony Gitter** [1,2,3,*]

[1]Department of Computer Sciences, University of Wisconsin–Madison, USA
[2]Morgridge Institute for Research, Madison, Wisconsin, USA
[3]Department of Biostatistics and Medical Informatics, University of Wisconsin–Madison, USA.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Cells regulate themselves via dizzyingly complex biochemical processes called signaling pathways. These are usually summarized pictorially as a directed graph where nodes represent proteins and edges indicate their influence on each other. In order to understand diseases and therapies at the cellular level, it is crucial to have an accurate understanding of the signaling pathways at work. Because signaling pathways can be modified by disease, the ability to infer signaling pathways from condition- or patient-specific data is highly valuable.

A variety of techniques exist for inferring signaling pathways. We build on past works that formulate signaling pathway inference as a Dynamic Bayesian Network structure estimation problem on phosphoproteomic time course data. We take a Bayesian approach, using Markov Chain Monte Carlo to estimate a posterior distribution over possible Dynamic Bayesian Network structures. Our primary contributions are a novel proposal distribution that efficiently samples large, sparse graphs and the relaxation common restrictive modeling assumptions.

Results: We implement our method, named Sparse Signaling Pathway Sampling, in Julia using the Gen probabilistic programming language. Probabilistic programming is a powerful methodology for building statistical models. The resulting code is modular, extensible, and legible. The Gen language, in particular, allows us to customize our inference procedure for biological graphs and ensure efficient sampling.

We evaluate our algorithm on simulated data and the HPN-DREAM pathway reconstruction challenge, comparing our performance against a variety of baseline methods. Our results demonstrate the vast potential for probabilistic programming, and Gen specifically, for biological network inference.

Availability: The full codebase is available at https://github.com/gitter-lab/ssps

Contact: gitter@biostat.wisc.edu

1 Introduction

Signaling pathways enable cells to process information in order to rapidly respond to external environmental changes or intracellular cues. One of the core signaling mechanisms is protein phosphorylation. Kinases add phosphate groups to substrate proteins and phosphatases remove them. These changes in phosphorylation state can act as switches, controlling proteins' activity and function. A protein's phosphorylation status affects its localization, stability, and interaction partners (Newman *et al.*, 2014). Ultimately, phosphorylation changes regulate important biological processes such as transcription and cell growth, death, and differentiation (Hunter, 2009; Kholodenko *et al.*, 2010).

Pathway databases characterize the signaling relationships among groups of proteins but are not tailored to individual biological contexts. Even for well-studied pathways such as epidermal growth factor receptor-mediated signaling, the proteins significantly phosphorylated during a biological response can differ greatly from those in the curated pathway (Köksal *et al.*, 2018). The discrepancy can be due to context-specific

signaling (Hill *et al.*, 2017), cell type-specific protein abundances, or signaling rewiring in disease (Pawson and Warner, 2007). Therefore, there is a need to learn context-specific signaling pathway representations from observed phosphorylation changes. In the clinical setting, patient-specific signaling pathway representations may eventually be able to guide therapeutic decisions (Drake *et al.*, 2016; Halasz *et al.*, 2016; Eduati *et al.*, 2020).

Diverse classes of techniques have been developed to model and infer signaling pathways (Kholodenko *et al.*, 2012). They take approaches including Granger causality (Shojaie and Michailidis, 2010; Carlin *et al.*, 2017), information theory (Cheong *et al.*, 2011; Krishnaswamy *et al.*, 2014), logic models (Eker *et al.*, 2002; Guziolowski *et al.*, 2013; Gjerga *et al.*, 2020), differential equations (Schoeberl *et al.*, 2002; Molinelli *et al.*, 2013; Henriques *et al.*, 2017), non-parametric statistical tests (Zhang and Song, 2013), and probabilistic graphical models (Sachs *et al.*, 2005) among others. Some signaling pathway reconstruction algorithms take advantage of perturbations such as receptor stimulation or kinase inhibition. Although perturbing individual pathway members can causally link them to downstream phosphorylation changes, characterizing a complex pathway can require a large number of perturbation experiments. Inferring pathway structure from temporal phosphorylation data presents an attractive alternative. A single time series phosphorylation dataset can reveal important dynamics without perturbing individual pathway members. For instance, a kinase cannot phosphorylate substrates before it is activated.

An alternative approach to pathway reconstruction selects a context-specific subnetwork from a general background network. These algorithms can use phosphorylation data to assign scores to protein nodes in a protein-protein interaction network. They then select edges that connect the high-scoring nodes, generating a subnetwork that may explain how the induced phosphorylation changes arise from the source of stimulation. Extensions accommodate temporal scores on the nodes (Patil *et al.*, 2013; Budak *et al.*, 2015; Köksal *et al.*, 2018; Norman and Cicek, 2019).

Our present work builds on past techniques that formulate signaling pathway inference as a Dynamic Bayesian Network (DBN) structure estimation problem. This family of techniques relies on two core ideas: *(i)* we can use a DBN to model phosphorylation time series data; and *(ii)* the DBN's structure translates directly to a directed graph representing the signaling pathway. Rather than identifying a single DBN that best fits the data, these techniques take a Bayesian approach—they yield a *posterior distribution* over possible DBN structures. Some techniques use Markov Chain Monte Carlo (MCMC) to sample from the posterior (Werhli and Husmeier, 2007; Gregorczyk, 2010). Others use exact, enumerative inference to compute posterior probabilities (Hill *et al.*, 2012; Oates *et al.*, 2014; Spencer *et al.*, 2015).

We present a new Bayesian DBN-based technique, Sparse Signaling Pathway Sampling (SSPS). It improves on past MCMC methods by using a novel proposal distribution specially tailored for the large, sparse graphs prevalent in biological applications. SSPS also makes weaker modeling assumptions than other DBN approaches. As a result, SSPS scales to larger problem sizes and yields superior predictions in comparison to other DBN techniques.

We implement SSPS using the `Gen` probabilistic programming language. Probabilistic programming is a powerful methodology for building probabilistic models. It enables the programmer to build models in a legible, modular, reusable fashion. This flexibility was important for prototyping and developing the current form of SSPS and readily supports future improvements or extensions.

2 Materials and methods

2.1 Model formulation

SSPS makes specific modeling assumptions. We start with the DBN model of Hill *et al.* (2012), relax some assumptions, and modify it in other ways to be better-suited for MCMC inference.

Preliminary definitions. We first define some notation for clarity's sake. Let G denote a *directed graph* with vertices V and edges $E(G)$. Graph G will represent a signaling pathway with vertices V corresponding to proteins and edges $E(G)$ indicating their influence relationships. We use $\text{pa}_G(i)$ to denote the *parents* of vertex i in G.

Let X denote our time series data, consisting of $|V|$ variables measured at T timepoints. X is a $T \times |V|$ matrix where the jth column corresponds to the jth variable and the jth graph vertex. As a convenient shorthand, let X_+ denote the *latest* $T-1$ timepoints in X, and let X_- denote the *earliest* $T-1$ timepoints in X. Lastly, define $B_j \equiv X_{-,\text{pa}_G(j)}$. In other words, B_j contains the value of variable j's parents at the $T-1$ earliest timepoints. In general, B_j may also include columns of nonlinear interactions between the parents. We will only include linear terms, unless stated otherwise. The utility of this shorthand will become apparent when we define the model.

Model derivation. In our setting, we aim to infer G from X. In particular, Bayesian approaches seek a *posterior distribution* $P(G|X)$ over possible graphs. From Bayes' rule we know $P(G|X) \propto P(X|G) \cdot P(G)$. That is, a Bayesian model is fully specified by its choice of *prior distribution* $P(G)$ and *likelihood function* $P(X|G)$.

We derive our model from the one used by Hill *et al.* (2012). They choose a prior distribution of the form

$$P(G \mid G', \lambda) \propto \exp\left(-\lambda |E(G) \setminus E(G')|\right) \tag{1}$$

parameterized by a *reference graph* G' and *inverse temperature* λ. This prior gives uniform probability to all subgraphs of G' and "penalizes" edges not contained in $E(G')$. λ controls the "importance" given to the reference graph.

Hill *et al.* choose a Gaussian DBN for their likelihood function. Intuitively, they assume linear relationships between variables and their parents:

$$X_{+,j} \sim \mathcal{N}(B_j \beta_j, \sigma_j^2) \qquad \forall j \in \{1 \dots |V|\}.$$

A suitable prior over the regression coefficients β_j and noise parameters σ_j^2 (Figure 1) allows us to integrate them out, yielding this *marginal likelihood function*:

$$P(X|G) \propto \prod_{j=1}^{|V|} (T+1)^{-\frac{|\text{pa}_G(j)|}{2}} \left(X_{+,j}^\top X_{+,j} - \frac{T}{T+1} X_{+,j}^\top (B_j \hat{\beta}_{ols}) \right)^{-\frac{T}{2}} \tag{2}$$

where $\hat{\beta}_{ols} = (B_j^\top B_j)^{-1} B_j^\top X_{+,j}$ is the ordinary least squares estimate of β_j.

In SSPS we use the same marginal likelihood function (Equation 2), but a different prior distribution $P(G)$. We obtain our prior distribution by decomposing Equation 1 into a product of independent Bernoulli trials over graph edges. This decomposition in turn allows us to make some useful generalizations. Define *edge existence variables* $z_{ij} \equiv \mathbb{1}[(i,j) \in E(G)]$. Let Z be the $|V| \times |V|$ matrix of all z_{ij}. Then, we can rewrite Equation 1 as follows:

$$P(G|G', \lambda) \equiv P(Z|G', \lambda) \propto \prod_{(i,j) \notin E(G')} e^{-z_{ij}\lambda}$$

$$= \prod_{(i,j) \in E(G')} \left(\frac{1}{2}\right)^{z_{ij}} \left(\frac{1}{2}\right)^{1-z_{ij}} \prod_{(i,j) \notin E(G')} \left(\frac{e^{-\lambda}}{1+e^{-\lambda}}\right)^{z_{ij}} \left(\frac{1}{1+e^{-\lambda}}\right)^{1-z_{ij}}$$

where the last line is a true equality—it gives a normalized probability measure. We see that the original prior is simply a product of Bernoulli variables parameterized by a shared inverse temperature, λ.

Rewriting the prior in this form opens the door to generalizations. First, we address a shortcoming in the way reference graph G' expresses prior knowledge. The original prior assigns equal probability to every edge of G'. However, in practice we can have differing levels of prior confidence in the edges. Expressing this requires *continuous-valued* prior edge confidences. We address this by allowing a real-valued prior confidence c_{ij} for each edge:

$$P(Z|C,\lambda) = \prod_{(i,j)} \left(\frac{e^{-\lambda}}{e^{-c_{ij}\lambda}+e^{-\lambda}}\right)^{z_{ij}} \left(\frac{e^{-c_{ij}\lambda}}{e^{-c_{ij}\lambda}+e^{-\lambda}}\right)^{1-z_{ij}} \quad (3)$$

where C is the matrix of all prior confidences c_{ij}, replacing G'. Notice that if every $c_{ij} \in \{0,1\}$, then Equation 3 is equivalent to the original prior. In effect, Equation 3 *interpolates* the original prior, permitting a continuum of confidences on the interval $[0,1]$.

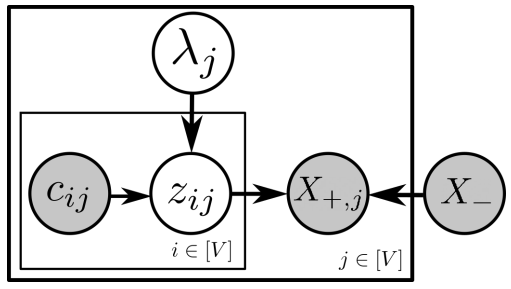
We make one additional change to the prior by replacing the shared λ inverse temperature variable with a collection of variables, $\Lambda = \{\lambda_j \mid j = 1,\ldots,|V|\}$, one for each vertex of the graph. Recall that the original λ variable determined the importance of the reference graph. In the new formulation, each λ_j controls the importance of the prior knowledge for vertex j and its parents:

$$P(Z|C,\Lambda) = \prod_{(i,j)} \left(\frac{e^{-\lambda_j}}{e^{-c_{ij}\lambda_j}+e^{-\lambda_j}}\right)^{z_{ij}} \left(\frac{e^{-c_{ij}\lambda_j}}{e^{-c_{ij}\lambda_j}+e^{-\lambda_j}}\right)^{1-z_{ij}} \quad (4)$$

We introduced Λ primarily to help MCMC converge more efficiently. Experiments with the shared prior revealed a multimodal posterior that tended to trap λ in high or low values. The introduction of vertex-specific λ_j variables yielded faster convergence with weaker modeling assumptions—an improvement in both respects.

We implicitly relax the model assumptions further via our inference procedure. For sake of tractability, the original exact method of Hill *et al.* (2012) imposes a hard constraint on the in-degree of each vertex. In contrast, we use a MCMC inference strategy with no in-degree constraints. Section 2.2 describes our strategy in detail.

In summary, our model departs from that of Hill *et al.* (2012) in three important respects. It permits real-valued prior confidences C; introduces vertex-specific inverse temperature variables Λ; and places no constraints on vertices' in-degrees. See the full model in Figure 1.



$$\lambda_j \sim \text{Uniform}(\lambda_{\min}, \lambda_{\max}) \qquad \forall j \in \{1\ldots|V|\}$$

$$z_{ij} \mid c_{ij}, \lambda_j \sim \text{Bernoulli}\left(\frac{e^{-\lambda_j}}{e^{-c_{ij}\lambda_j}+e^{-\lambda_j}}\right) \qquad \forall i,j \in \{1\ldots|V|\}$$

$$\sigma_j^2 \propto \frac{1}{\sigma_j^2} \qquad \forall j \in \{1\ldots|V|\}$$

$$\beta_j \mid \sigma_j^2 \sim \mathcal{N}\left(0, T\sigma_j^2 (B_j^\top B_j)^{-1}\right) \qquad \forall j \in \{1\ldots|V|\}$$

$$X_{+,j} \mid B_j, \beta_j, \sigma_j^2 \sim \mathcal{N}\left(B_j\beta_j, \sigma_j^2 I\right) \qquad \forall j \in \{1\ldots|V|\}$$

Fig. 1. Our generative model. (top) Plate notation. DBN parameters β_j and σ_j^2 have been marginalized out. (bottom) Full probabilistic specification. We usually set $\lambda_{\min} \simeq 3$ and $\lambda_{\max}$=15. If $\lambda_{\min}$>0 is too small, Markov chains will occasionally be initialized with very large numbers of edges, causing computational issues. The method is insensitive to $\lambda_{\max}$ as long as it's sufficiently large. Notice the improper prior $1/\sigma_j^2$. In this specification B_j denotes $X_{-,\text{pa}_Z(j)}$; that is, the parents of vertex j depend on edge existence variables Z.

2.2 Inference procedure

Our method uses MCMC to infer posterior edge existence probabilities. As described in Section 2.1, our model contains two classes of random variables: *(i)* the edge existence variables Z and *(ii)* the inverse temperature variables Λ. For each step of MCMC, we loop through these variables and update them in a Metropolis-Hastings fashion. Details are given below.

Main loop. At a high level, our MCMC procedure consists of a loop over the graph vertices, V. For each vertex j, we update its inverse temperature variable λ_j and then update its *parent set* $\text{pa}_G(j)$. All of these updates are Metropolis-Hastings steps; the proposal distributions are described below. Each completion of this loop yields one sample in the Markov chain.

Proposal distributions. For the inverse temperature variables we use a symmetric Gaussian proposal: $\lambda_j' \sim \mathcal{N}(\lambda_j, \xi^2)$. In practice, the method is insensitive to ξ; we typically set ξ=3.

The parent set proposal distribution is more complicated. There are two principles at work when we design a graph proposal distribution: *(i)* the proposal should efficiently traverse the space of directed graphs, and *(ii)* it should favor graphs with higher posterior probability. The most widely used graph proposal distribution selects a *neighboring* graph uniformly from the set of possible "add-edge," "remove-edge," and "reverse-edge" actions (Werhli and Husmeier, 2007; Gregorczyk, 2010). We'll refer to this traditional proposal distribution as the *uniform graph proposal*. In our setting, we expect sparse graphs to be much more probable than dense ones—notice how the marginal likelihood function (Equation 2) strongly penalizes $|\text{pa}_G(j)|$. However, the uniform graph proposal exhibits a preference toward *dense graphs*. It proposes "add-edge" actions too often. This motivates us to design a new proposal distribution tailored for sparse graphs—one which operates on our sparse *parent set* graph representation.

For a given graph vertex $j \in V$, the parent set proposal distribution updates $\text{pa}_G(j)$ by choosing from the following actions:

- `add-parent`. Select one of vertex j's non-parents uniformly at random, and add it to $\text{pa}_G(j)$.
- `remove-parent`. Select one of vertex j's parents uniformly at random, and remove it from $\text{pa}_G(j)$.
- `swap-parent`. A simultaneous application of `add-parent` and `remove-parent`. Perhaps surprisingly, this action is not made redundant by the other two. It plays an important role by yielding updates that maintain the size of the parent set. Because the marginal likelihood (Equation 2) changes steeply with $|\text{pa}_G(j)|$, Metropolis-Hastings acceptance probabilities will be higher for actions that keep $|\text{pa}_G(j)|$ constant.

These three actions are sufficient to explore the space of directed graphs, but we need another mechanism to bias the exploration toward *sparse* graphs. We introduce this preference via the *probability* assigned to each action. Intuitively, we craft the action probabilities so that when $|\text{pa}_G(j)|$ is too small, `add-parent` moves are most probable. When $|\text{pa}_G(j)|$ is too big, `remove-parent` moves are most probable. When $|\text{pa}_G(j)|$ is about right, `swap-parent` moves are given strong preference.

We formulate the action probabilities for vertex j as follows. As a shorthand, let $s_j = |\text{pa}_G(j)|$ and define the *reference size* $\hat{s}_j = \sum_{i=1}^{|V|} c_{ij}$. That is, $\hat{s}_j$ uses the prior edge confidences C to estimate an

appropriate reference size for the parent set. Then, the action probabilities are given by

$$p(\texttt{add-parent}|s_j, \hat{s}_j) \propto 1 - \left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)}$$

$$p(\texttt{remove-parent}|s_j, \hat{s}_j) \propto \left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)}$$

$$p(\texttt{swap-parent}|s_j, \hat{s}_j) \propto 2\left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)} \cdot \left(1 - \left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)}\right)$$

where $\gamma(\hat{s}_j) = 1/\log_2(|V|/\hat{s}_j)$. We use these functional forms only because they have certain useful properties: *(i)* when s_j=0, the probability of `add-parent` is 1; *(ii)* when s_j=$|V|$, the probability of `remove-parent` is 1; and *(iii)* when s_j=$\hat{s}_j$, all actions have equal probability. Beyond that, these probabilities have no particular justification.

Recall that Metropolis-Hastings requires us to compute the reverse transition probability for any proposal we make. This could pose a challenge given our relatively complicated parent set proposal distribution. However, `Gen` provides a helpful interface for computing reverse probabilities. Roughly speaking, the user can provide an *involution* function that returns the reverse of a given action. `Gen` then manages the reverse probabilities without further intervention. This makes it relatively easy to implement Metropolis-Hastings updates with unusual proposal distributions.

Termination, convergence, and inference. We follow the basic MCMC protocols described by Gelman *et al.* (2014). This entails running multiple (i.e., 4) Markov chains and discarding the first half of each chain as burnin. In all of our analyses, we terminate each Markov chain when it either *(i)* reaches a length of 100,000 iterations or *(ii)* the execution time exceeds 12 hours. These termination conditions are somewhat arbitrary but emulate a real-world setting where it may be acceptable to let the method run overnight.

Upon termination, we assess convergence with two diagnostics: Potential Scale Reduction Factor (PSRF) and effective number of samples (N_{eff}). PSRF identifies cases where the Markov chains fail to mix or achieve stationarity. N_{eff} provides a sense of "sample size" for our inferred quantities. It adjusts the number of MCMC samples by accounting for autocorrelation in each chain. For our purposes, we say a quantity has *failed to converge* if its PSRF ≥ 1.01 or N_{eff}<10. Note that satisfying these criteria does not guarantee convergence. However, failure to satisfy them is a reliable flag for non-convergence.

Assuming a quantity hasn't failed to converge, we estimate it by simply taking its sample mean from all samples remaining after burnin. In our setting we are primarily interested in *edge existence* probabilities; i.e., we compute the fraction of samples containing each edge.

2.3 Implementation: probabilistic programming

We implemented SSPS using the `Gen` probabilistic programming language. We briefly describe the probabilistic programming methodology and its advantages in our setting.

Probabilistic programming. Probabilistic programming is a methodology for building statistical models. It's based on the idea that statistical models are *generative processes*—sequences of operations on random variables. In probabilistic programming, we express the generative process as a *program* written in a *probabilistic programming language* (PPL). This program is then compiled to produce a log-probability function, which can be used in inference tasks. Probabilistic programming systems typically provide a set of generic inference methods for performing those tasks—e.g., MCMC or Variational Bayes.

PPL	Host language	Primary model class	Primary inference method
`Stan`	custom language	hierarchical, cont's vars	Black-box HMC
`Edward2`	Python/ TensorFlow	"deep", cont's vars	Black-box variational
`PyMC3`	Python/Theano	"deep", cont's vars	Black-box HMC
`Pyro`	Python/PyTorch	"deep", cont's vars	Black-box variational
`Gen`	Julia	discrete and cont's vars; highly flexible	Customizable MCMC

Table 1. A coarse comparison of some noteworthy PPLs. Most PPLs aim to provide a black-box interface for inference—the user is spared the difficulty of designing an inference procedure. However, this convenience comes at the cost of language restrictions. `Gen` is an exception. It provides greater expressiveness but requires the user to implement an inference program for their model. Cont's vars: continuous variables; HMC: Hamiltonian Monte Carlo.

Compare this with a more traditional approach, where the user must *(i)* derive and implement the log-probability function; and *(ii)* implement an inference method that operates on the log-probability function. This process of manual derivation and implementation is error-prone, and requires a high degree of expertise from the user. In contrast, probabilistic programming only requires the user to express their model in a PPL. The probabilistic programming system manages other details.

Probabilistic programming also tends to promote good software engineering principles such as abstraction, modularity, and legibility. Most PPLs organize code into functions, which can be reused by multiple statistical models.

Probabilistic programming languages. Several PPLs have emerged in recent years. Examples include `Stan` (Carpenter *et al.*, 2017), `Edward2` (Dillon *et al.*, 2017), `Pyro` (Bingham *et al.*, 2018), `PyMC3` (Salvatier *et al.*, 2016), and `Gen` (Cusumano-Towner *et al.*, 2019). PPLs differ in how they balance *expressive power* and *ease of use*. For example, `Stan` makes it easy to build hierarchical statistical models with continuous variables but caters poorly to other model classes. At the other extreme, `Gen` can readily express a large class of models—discrete and continuous variables with complex relationships—but requires the user to design a customized inference procedure.

Implementation in `Gen`. We use the `Gen` PPL precisely for its expressive power and customizable inference. While implementing SSPS, the customizability of `Gen` allowed us to begin with simple prototypes and then make successive improvements. For example, our model initially used a dense *adjacency matrix* representation for G, but subsequent optimizations led us to use a sparse *parent set* representation instead. Similarly, our MCMC method started with a naïve "add or remove edge" proposal distribution; we arrived at our sparse proposal distribution (described in Section 2.2) after multiple refinements. Other PPLs do not allow this level of control (Table 1).

2.4 Evaluation: simulation study

We use a simulation study to answer important questions about SSPS: How does its computational expense grow with problem size? Is it able to correctly identify true edges? What is its sensitivity to errors in the prior knowledge? Simulations allow us to answer these questions in a controlled setting where we have access to ground truth.

Data simulation process. We generate each simulated dataset as follows:

Parameter	Meaning	Values
$\|V\|$	Number of variables	40, 100, 200
T	Time course length	8
M	Number of time courses	4
r	Fraction of original edges removed	0.1, 0.5, 0.75, 1.0
a	Fraction of spurious edges added	0.1, 0.5, 0.75, 1.0

Table 2. These parameters define the grid of simulated datasets in our simulation study. There are $3{\times}4{\times}4{=}48$ distinct grid points. For each one, we generate $K{=}5$ replicates for a total of 240 simulated datasets. The graph corruption parameters, r and a, range from very little error (0.1) to total corruption (1.0).

1. Sample a random adjacency matrix $A \in \{0,1\}^{|V|\times|V|}$, where each entry is the outcome of a Bernoulli(p) trial. A specifies the *structure* of a DBN. We choose $p{=}5/|V|$ so that each vertex has an average of 5 parents. This approximates the sparsity we might see in signaling pathways. We denote the size of the original edge set as $|E_0|$.
2. Let the weights β for this DBN be drawn from a normal distribution $\mathcal{N}(0, 1/\sqrt{|V|})$. We noticed empirically that the $1/\sqrt{|V|}$ scale prevented the simulated time series from diverging to infinity.
3. Use the DBN defined by A, β to simulate M time courses of length T. We imitate the real datasets in Section 2.5 by generating $M{=}4$ time courses, each of length $T{=}8$.
4. Corrupt the adjacency matrix A in two steps: *(i)* remove $r \cdot |E_0|$ of the edges from A; *(ii)* add $a \cdot |E_0|$ spurious edges to the adjacency matrix. This corrupted graph simulates the *imperfect prior knowledge* encountered in reality. The parameters r and a control the "false negatives" and "false positives" in the prior knowledge, respectively.

We use a range of values for parameters $|V|, r$, and a, yielding a grid of simulations summarized in Table 2.

Performance metrics. We are primarily interested in SSPS's ability to correctly recover the structure of the underlying signaling pathway. The simulation study allows us to measure this in a setting where we have access to ground truth. We treat this as a probabilistic binary classification task, where the method assigns an *existence confidence* to each possible edge. We measure classification performance using area under the Precision-Recall curve (AUCPR). We use *average precision* to estimate AUCPR, as opposed to the trapezoidal rule (which tends to be overly-optimistic, see Davis and Goadrich (2006); Flach and Kull (2015)).

Our decision to use AUCPR over AUCROC or other scores is motivated by the sparseness of the graphs. For sparse graphs the number of edges grows linearly with $|V|$ while the number of possible edges grows quadratically. Hence, as $|V|$ grows the classification task increasingly becomes a "needle-in-haystack" scenario.

Performance measurements on simulated data come with many caveats. It's most instructive to think of simulated performance as a sanity check. Since our data simulator closely follows our modeling assumptions, poor performance would suggest serious shortcomings in our method.

2.5 Evaluation: HPN-DREAM network inference challenge

We measure SSPS's performance on experimental data by following the evaluation outlined by the HPN-DREAM Breast Cancer Network Inference Challenge (Hill *et al.*, 2016). Signaling pathways differ across contexts—e.g., cell type and environmental conditions. The challenge is to infer these context-specific signaling pathways from time course data.

The dataset. The HPN-DREAM challenge provides phosphorylation time course data from 32 biological contexts. These contexts arise from exposing 4 cell lines (BT20, BT549, MCF7, UACC812) to 8 stimuli. For each context there are approximately $M{=}4$ time courses, each about $T{=}7$ time points in length. Cell lines have differing numbers of phosphosite measurements (i.e., they have differing $|V|$), ranging from 39 (MCF7) to 46 (BT20).

Prior knowledge. Participants in the original challenge were free to extract prior knowledge from any existing data sources. As part of their analysis, the challenge organizers combined participants' prior graphs into a set of edge probabilities. These *aggregate priors* summarize the participants' collective knowledge.They were not available to participants in the original challenge, but we use them in our analyses of HPN-DREAM data. We provide them to each of the baseline methods (see Section 2.6), so the resulting performance comparisons are fair. We do not compare any of our scores to those listed by Hill *et al.* (2016) in the original challenge results.

Performance metrics. The HPN-DREAM challenge aims to score methods by their ability to capture causal relationships between pairs of variables. It estimates this by comparing predicted *descendant sets* against experimentally generated descendant sets. More specifically, the challenge organizers exposed cells to multiple mTOR inhibitors and observed the effects on other phosphosites. From this they determined a set of phosphosites *downstream* of mTOR in the signaling pathway. These include direct substrates of the mTOR kinase as well as indirect targets.

Comparing predicted descendants of mTOR against experimentally generated descendants of mTOR gives us a notion of *false positives* and *false negatives*. As we vary a threshold on edge probabilities, the predicted mTOR descendants change, which allows us to make a Precision-Recall curve. We use the resulting AUCPR (average precision) to score methods' performance on the HPN-DREAM challenge. Hill *et al.* (2016) provide more details for this descendant set AUCPR scoring metric.

Because SSPS is stochastic we run it $K{=}5$ times per context, yielding 5 AUCPR scores per context. Meanwhile the baseline methods (see Section 2.6) are all deterministic, requiring only one execution per context. We use a simple terminology to compare SSPS's scores against those of other methods. In a given context, we say SSPS *dominates* another method if its *minimum* score exceeds that of the other method. Conversely, we say the other method dominates SSPS if its score exceeds SSPS's *maximum* score. This *dominance* comparison has flaws—e.g., its results depend on the sample size K. However it errs on the side of strictness and suffices as an aid for summarizing the HPN-DREAM evaluation results.

2.6 Baseline pathway inference algorithms

Our evaluations compare SSPS against a diverse set of baseline methods, which we describe here.

Exact DBN (Hill et al.*, 2012).* This method was an early inspiration for SSPS and is most similar to SSPS (Section 2.1). However, the exact DBN method encounters unique practical issues when we run it on real or simulated data. The method's computational expense increases rapidly with problem size $|V|$, and becomes intractable unless the "max-indegree" parameter is set to a small value. For example, we found that the method used more than 32GB of RAM on problems of size $|V|{=}100$, unless max-indegree were set ≤ 3. Furthermore, the exact DBN method only admits prior knowledge in the form of Boolean *reference edges*, rather than continuous-valued edge confidences. We overcame this by using a threshold to map edge confidences to 1 or 0. We chose a threshold of 0.25 for the HPN-DREAM challenge evaluation because it yielded a reasonable number of prior edges.

`FunChisq` *(Zhang and Song, 2013).* This method is based on the notion that two variables X, Y have a causal relationship if there exists a *functional dependence* $Y{=}f(X)$ between them. It detects these dependencies using a chi-square test against the "no functional dependence" null hypothesis. `FunChisq` was a strong competitor in the HPN-DREAM challenge, despite the fact that it uses no prior knowledge.

In order to use `FunChisq`, one must first discretize their time course data. We followed Zhang and Song's recommendation to use 1D k-means clustering for discretization. Detailed instructions are given in the HPN-DREAM challenge supplementary materials (Hill *et al.*, 2016).

LASSO. We included a variant of LASSO regression as a simple baseline. It incorporates prior knowledge into the typical primal formulation:

$$\hat{\beta_j} = \text{argmin}_\beta \left\{ \|X_{+,j} - B_j\beta\|_2^2 \; + \; \alpha \sum_{i=1}^{V} e^{-c_{ij}} |\beta_{ij}| \right\}$$

where c_{ij} is the prior confidence (either Boolean or real-valued) for edge (i, j). That is, the method uses *penalty factors* $e^{-c_{ij}}$ to discourage edges with low prior confidence. The method selects LASSO parameters, α, using the Bayesian Information Criterion described by Zou *et al.* (2007). We utilize the `GLMNet` implementation (Friedman *et al.*, 2010) via the Julia wrapper located at https://github.com/JuliaStats/GLMNet.jl.

Prior knowledge baseline. Our most straightforward baseline simply reports the prior edge probabilities, performing no inference at all. Ideally, a Bayesian method should do no worse than the prior—new time course data should only *improve* our knowledge of the true graph. In reality, this improvement is subject to caveats about data quality and model fit.

2.7 SSPS software availability

We provide the code for SSPS via GitHub, distributed under a MIT license (https://github.com/gitter-lab/ssps). It includes a Snakemake workflow (Koster and Rahmann, 2012) for our full evaluation pipeline, enabling the reader to reproduce our results.

3 Results

We describe evaluation results from the simulation study and HPN-DREAM network inference challenge. We find that SSPS competes well against the baselines, with superior scalability to other DBN-based approaches.

3.1 Simulation study results

We compare our method to the baselines listed in Section 2.6. We focus especially on the exact DBN method of Hill *et al.* (2012), as SSPS shares many modeling assumptions with it.

Computational expense. Because SSPS uses MCMC, the user may allow it to run for an arbitrary amount of time. With this in mind, we summarize SSPS's time expense with two numbers: *(i)* N/cpu-hr, the number of MCMC samples per CPU-hour; and *(ii)* N_{eff}/cpu-hr, the *effective* number of samples per CPU-hour. We also measure the memory footprint per Markov chain, subject to our termination conditions. We measured these numbers for each simulation in our grid (see Table 2).

Table 3 reports average values of N/cpu-hr, N_{eff}/cpu-hr, and memory footprint for each problem size. As we expect, N/cpu-hr and N_{eff}/cpu-hr both decrease approximately with the inverse of $|V|$. In contrast, the non-monotonic memory usage requires more explanation. It results from two causes: *(i)* our termination condition and *(ii)* the sparse data structures we use to store samples. On small problems ($|V|$=40), the Markov chain terminates at a length of 100,000—well within the 12-hour limit. On larger problems ($|V|$=100 or 200) the Markov chain terminates at the 12-hour timeout. This accounts for the 500MB gap between small and large problems. The *decrease* in memory usage between $|V|$=100 and 200 results from our sparse representations for samples. Roughly speaking, the sparse format only stores *changes* in the variables. So the memory consumption of a Markov chain depends not only on $|V|$, but also on the *acceptance rate* of the Metropolis-Hastings proposals. The acceptance rate is smaller for $|V|$=200, yielding a net decrease in memory usage.

$\lvert V\rvert$	N/cpu-hr	N_{eff}/cpu-hr	MB per chain
40	70000	400	500
100	9000	140	1200
200	3000	60	1000

Table 3. Computational expense of SSPS as a function of problem size $|V|$. N is the number of iterations completed by a Markov chain. N_{eff} accounts for burnin and autocorrelation in the Markov chains, giving a more accurate sense of the method's progress (see Section 2.2). The last column gives the approximate memory footprint of each chain. The non-monotonic memory usage is an artifact of the chain termination conditions (N>100,000 or time >12 hours).

$\lvert V\rvert$	max indeg	"linear"	"full"
40	4	66s	210s
	5	770s	3900s
	6	6700s	**TIMEOUT**
	7	**OOM**	**OOM**
100	3	250s	520s
	4	**OOM**	**OOM**
200	2	53s	140s
	3	**OOM**	**OOM**

Table 4. Computational expense of the exact DBN method of Hill et al. (2012) measured in CPU-seconds, as a function of problem size $|V|$ and various parameter settings. The method imposes an in-degree constraint on each vertex, shown in the "max indeg" column. The columns "linear" and "full" correspond to different regression modes, i.e., which interaction terms are included in the DBN's conditional probability distributions. "OOM" (Out Of Memory) indicates that the method exceeded a 32GB memory limit. "TIMEOUT" indicates that the method failed to complete within 12 hours.

Recall that SSPS differs from more traditional MCMC approaches by nature of its parent set proposal distribution, which is specially designed for sparse graphs (see Section 2.2). When we modify SSPS to instead use a naïve uniform graph proposal, we see a striking difference in sampling efficiency. The uniform graph proposal distribution attains N_{eff}/cpu-hr of 100, 10, and 0.2 for $|V|$=40, 100 and 200, respectively—drastically smaller than those listed in Table 3 for the parent set proposal. It's possible that the traditional proposal could achieve higher N_{eff}/cpu-hr by simply running faster. However, the more important consideration is how N_{eff}/cpu-hr changes with $|V|$. Our parent set proposal distribution's N_{eff}/cpu-hr decays approximately like $1/|V|$. Meanwhile, the traditional proposal distribution's N_{eff}/cpu-hr decays faster than $1/|V|^2$. This makes an enormous difference for sampling efficiency on large problems.

Table 4 summarizes the computational expense of the exact DBN method (Hill *et al.*, 2012). The method quickly becomes impractical as the problem size increases unless we enforce increasingly strict modeling assumptions. In particular, the exact DBN method suffers from exponential memory cost as a function of its "max in-degree" parameter. The growth becomes increasingly sharp with problem size. When $|V| = 200$, increasing maximum in-degree from 2 to 3 makes the difference between terminating in <1 minute and exceeding 32GB of memory. Such low bounds on in-degree are unrealistic, and will likely result in poor inference quality. In comparison, SSPS makes no constraints on in-degree, and its memory usage scales well with problem size.

The other baseline methods—`FunChisq` and LASSO—are much less computationally expensive. Both finish in seconds and require less than 100MB of memory for each simulated task. This highlights the computationally intense nature of Bayesian approaches. Not every scenario

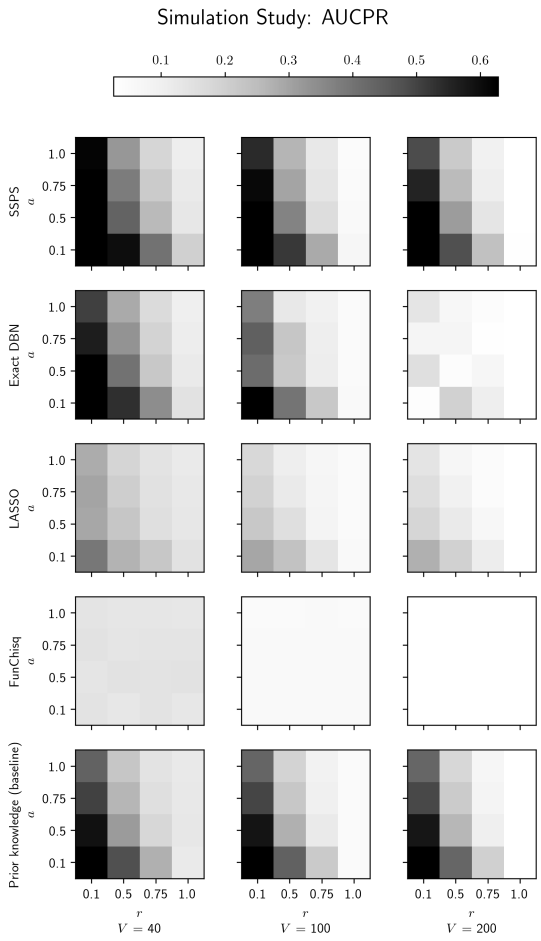


Fig. 2. Heatmap of AUCPR values from the simulation study. Both DBN-based techniques (SSPS and the exact method) score well on this, since the data is generated by a DBN. On large problems the exact DBN method needs strict in-degree constraints, leading to poor prediction quality. LASSO and `FunChisq` both perform relatively weakly.

calls for Bayesian inference. However, Bayesian inference is valuable in scientific settings where we're concerned with uncertainty quantification.

Predictive performance. The simulation study provides a setting where we have access to "ground truth"—the true simulated graph. We use AUCPR (see Section 2.4) to score each method's ability to recover the true graph's edges.

Figure 2 shows the AUCPR scores over our grid of simulations [1]. Each heat map shows AUCPR as a function of graph corruption parameters, r and a. The heat maps are arranged by method and problem size $|V|$. Each AUCPR value is an average over 5 replicates. We see SSPS maintains fairly consistent performance across problem sizes. In contrast, the other methods' scores decrease with problem size. For the exact DBN method, this is partially due to the imposition of small in-degree constraints on the large problems. The exact DBN method is forced to trade model accuracy for tractability.

[1] There is a slight discrepancy between the simulation procedure in Section 2.4 and the results in Figures 2 and 3. The original adjacency matrix was provided as the prior. The corrupted adjacency matrix was used to generate the time series data and evaluate the predicted networks. The simulation still accurately assesses how each method performs at different levels of error in the prior knowledge, but the interpretation of r and a differs. This does not affect any conclusions about the methods' runtime, memory usage, AUCPR performance, and relative merits; or the HPN-DREAM evaluation. We will post corrected simulation results in a preprint that will be linked from https://github.com/gitter-lab/ssps.

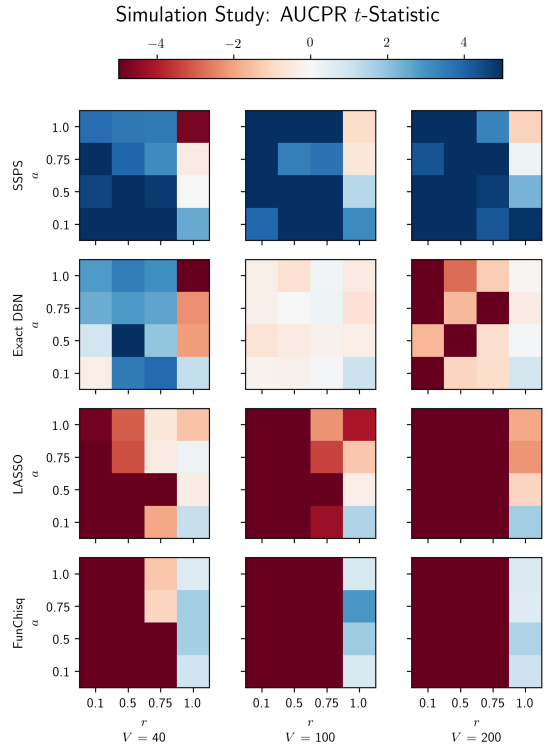


Fig. 3. Heatmap of differential performance against the prior knowledge, measured by AUCPR paired t-statistics. SSPS consistently outperforms the prior knowledge across problem sizes, and shows robustness to errors in the prior knowledge.

Figure 3 reveals further insights into these results. It plots *differential* performance with respect to the prior knowledge, in a layout analogous to Figure 2. Specifically, it plots the t-statistic of each method's AUCPR scores, paired with the prior baseline's AUCPR scores. As long as the prior graph has some informative edges, SSPS outperforms the prior baseline. We also see that SSPS performs as bad or worse than the prior whenever the prior contains *no* true edges (i.e., r=1). Under those circumstances `FunChisq` is a better choice. Since it doesn't rely on prior knowledge at all, it outperforms the other methods when the prior is totally corrupted. However, we argue that in most realistic settings there exists partially-accurate prior knowledge, in which case we expect `FunChisq` to perform worse than SSPS.

These results confirm SSPS's ability to identify the true network, given partially-accurate prior knowledge and time series data consistent with the modeling assumptions. They suggest that SSPS is fairly robust with respect to the quality of the prior and demonstrate its ability to perform consistently across different problem sizes.

3.2 HPN-DREAM challenge results

We evaluated SSPS on experimental data from the HPN-DREAM challenge. The HPN-DREAM challenge includes time series phosphorylation data from 32 biological contexts, the product of 4 breast cancer cell lines and 8 stimuli (see Section 2.5). Methods are scored on their ability to correctly identify the experimentally derived descendants of mTOR. Figure 4 shows barcharts comparing the methods' AUCPR scores in a given context.

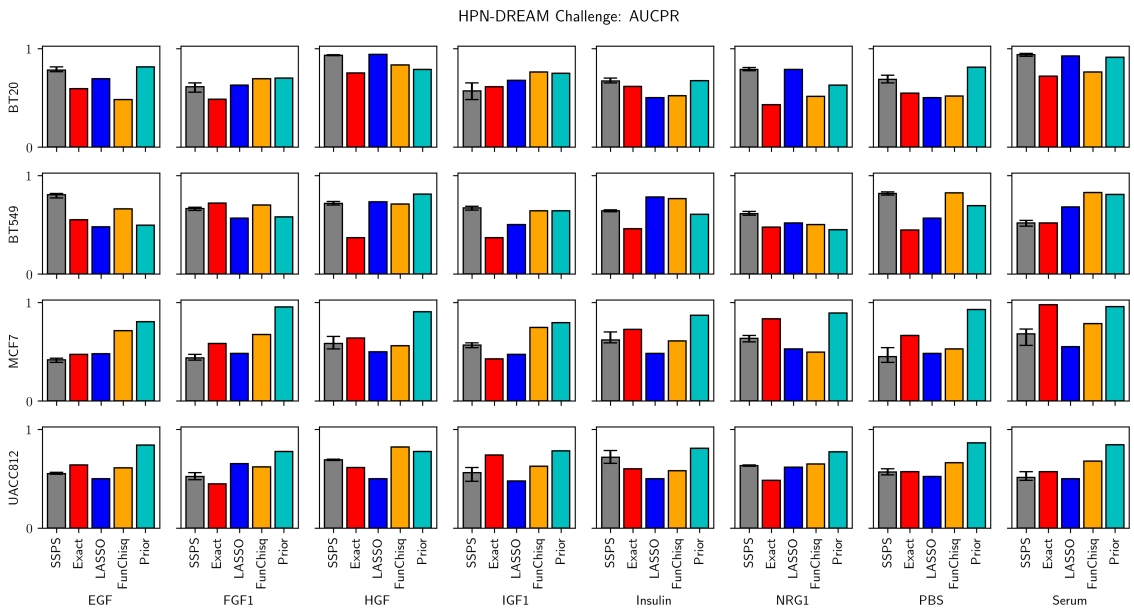


Fig. 4. Comparison of methods' performances across contexts in the DREAM Challenge. Notice the error bars for SSPS. Since MCMC is stochastic, we run the method 5 times; the error bars show the range of AUCPR scores. The other methods are all deterministic and require no error bars.

SSPS performs satisfactorily on this task overall. Employing terminology from Section 2.5, SSPS dominates the exact DBN method in 18 of the 32 contexts, whereas the exact DBN method dominates SSPS in only 9 contexts. Meanwhile, SSPS dominates `FunChisq` in 11 contexts and is dominated by `FunChisq` in 15. This is not surprising because `FunChisq` was among the top competitors in the original challenge. LASSO, on the other hand, performs poorly. SSPS dominates LASSO in 18 contexts and is dominated in only 6.

More puzzling is the strong performance of the prior knowledge baseline. SSPS dominates the aggregate prior in only 9 contexts and is dominated in 21. This is not isolated to our method. `FunChisq` dominates and is dominated by the prior knowledge in 11 and 21 contexts, respectively. The aggregate prior's strong performance is consistent with results from the original HPN-DREAM challenge; the aggregate prior outperformed all individual challenge submissions (Hill *et al.*, 2016). There are different ways to interpret this. We suspect that context specificity, in reality, plays a weak role in this task and that the time series are relatively uninformative.

4 Discussion

We presented SSPS, a signaling pathway reconstruction technique based on DBN structure estimation. It uses MCMC to estimate the posterior probabilities of directed edges, employing a parent set proposal distribution specially designed for sparse graphs. SSPS is a Bayesian approach. It takes advantage of prior knowledge with edge-specific confidence scores and can provide uncertainty estimates on the predicted pathway relationships, which are valuable for prioritizing experimental validation.

SSPS scales to large problems more efficiently than past DBN-based techniques. On simulated data, SSPS yields superior edge predictions with robustness to flaws in the prior knowledge. Our HPN-DREAM evaluation shows SSPS performs comparably to established techniques on a community standard task. In other words, the challenge *fails to discard* SSPS as unsuitable for real applications. It is difficult to make stronger statements in the HPN-DREAM setting because the prior knowledge baseline performs so well and we can only evaluate the predicted mTOR descendants, not the entire pathway. However, SSPS's scalability among Bayesian methods, strong results in the simulation, and competitive performance in the HPN-DREAM challenge make it an attractive option for further investigation of real phosphorylation datasets.

There are several potential limitations of SSPS relative to alternative pathway signaling models. Prior knowledge is not available in some organisms or biological conditions, minimizing one of the advantages of our Bayesian approach. Although SSPS is more scalable than related DBN techniques, it would struggle to scale to proteome-wide phosphoproteomic data measuring thousands or tens of thousands of phosphosites. For very large datasets, we recommend running SSPS on a pruned version that includes only the highest intensity or most variable phosphosites. SSPS, like most DBN techniques, models only observed variables. It will erroneously exclude important pathway members, such as scaffold proteins, that are not phosphorylated. Latent variable models or background network-based algorithms are better suited for including unphosphorylated proteins in the pathway. Background network methods can also impose global constraints on the predicted pathway structure, such as controlling the number of connected component or proteins' reachability from relevant receptors (Köksal *et al.*, 2018).

There are many possible ways to improve SSPS. For example, it could be extended to jointly model related pathways in a hierarchical fashion, similar to Oates *et al.* (2014) and Hill *et al.* (2017). Alternatively, SSPS could be modified to accommodate causal assumptions via Pearl's intervention operators; see the model of Spencer *et al.* (2015) for a relevant example. Combining temporal and interventional data (Cardner *et al.*, 2019) is another rich area for future work. On the algorithmic side, we could improve our MCMC procedure by adaptively tuning the parameters of its proposal distributions, as described by Gelman *et al.* (2014). Because SSPS is a probabilistic program, it is naturally extensible.

Funding

This work was funded by the National Institutes of Health (award T32LM012413) and National Science Foundation (award DBI 1553206).

Acknowledgements

We thank UW-Madison's Biomedical Computing Group for generously providing compute resources; the teams developing Snakemake and HTCondor (Thain *et al.*, 2005) for empowering us to use those resources effectively; and the `Gen` team (Cusumano-Towner *et al.*, 2019) for designing a uniquely powerful probabilistic programming language.

References

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. (2018). Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*.

Budak, G., Ozsoy, O. E., Son, Y. A., Can, T., and Tuncbag, N. (2015). Reconstruction of the temporal signaling network in Salmonella-infected human cells. *Frontiers in Microbiology*, **6**, 730.

Cardner, M., Meyer-Schaller, N., Christofori, G., and Beerenwinkel, N. (2019). Inferring signalling dynamics by integrating interventional with observational data. *Bioinformatics*, **35**(14), i577–i585.

Carlin, D. E., Paull, E. O., Graim, K., Wong, C. K., Bivol, A., Ryabinin, P., Ellrott, K., Sokolov, A., and Stuart, J. M. (2017). Prophetic granger causality to infer gene regulatory networks. *PLOS ONE*, **12**(12), e0170340.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, **76**(1).

Cheong, R., Rhee, A., Wang, C. J., Nemenman, I., and Levchenko, A. (2011). Information Transduction Capacity of Noisy Biochemical Signaling Networks. *Science*, **334**(6054), 354–358.

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., and Mansinghka, V. K. (2019). Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, New York, NY, USA. ACM.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. (2017). Tensorflow distributions.

Drake, J. M., Paull, E. O., Graham, N. A., Lee, J. K., Smith, B. A., Titz, B., Stoyanova, T., Faltermeier, C. M., Uzunangelov, V., Carlin, D. E., Fleming, D. T., Wong, C. K., Newton, Y., Sudha, S., Vashisht, A. A., Huang, J., Wohlschlegel, J. A., Graeber, T. G., Witte, O. N., and Stuart, J. M. (2016). Phosphoproteome Integration Reveals Patient-Specific Networks in Prostate Cancer. *Cell*, **166**(4), 1041–1054.

Eduati, F., Jaaks, P., Wappler, J., Cramer, T., Merten, C. A., Garnett, M. J., and Saez-Rodriguez, J. (2020). Patient-specific logic models of signaling pathways from screenings on cancer biopsies to prioritize personalized combination therapies. *Molecular Systems Biology*, **16**(2), e8664.

Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., and Sonmez, K. (2002). Pathway logic: symbolic analysis of biological signaling. *Pacific Symposium on Biocomputing*, pages 400–412.

Flach, P. A. and Kull, M. (2015). Precision-recall-gain curves: PR analysis done right. In *Advances in neural information processing systems*.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, **33**.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014). *Bayesian Data Analysis*. CRC Press, 3 edition.

Gjerga, E., Trairatphisan, P., Gabor, A., Koch, H., Chevalier, C., Ceccarelli, F., Dugourd, A., Mitsos, A., and Saez-Rodriguez, J. (2020). Converting networks to predictive logic models from perturbation signalling data with CellNOpt. *bioRxiv*, page 2020.03.04.976852.

Gregorczyk, M. (2010). An introduction to gaussian bayesian networks. In Q. Yan, editor, *Systems Biology in Drug Discovery and Development: Methods and Protocols*, chapter 6, pages 121–147. Springer.

Guziolowski, C., Videla, S., Eduati, F., Thiele, S., Cokelaer, T., Siegel, A., and Saez-Rodriguez, J. (2013). Exhaustively characterizing feasible logic models of a signaling network using Answer Set Programming. *Bioinformatics*, **29**(18), 2320–2326.

Halasz, M., Kholodenko, B. N., Kolch, W., and Santra, T. (2016). Integrating network reconstruction with mechanistic modeling to predict cancer therapies. *Science Signaling*, **9**(455), ra114.

Henriques, D., Villaverde, A. F., Rocha, M., Saez-Rodriguez, J., and Banga, J. R. (2017). Data-driven reverse engineering of signaling pathways using ensembles of dynamic models. *PLOS Computational Biology*, **13**(2), e1005379.

Hill, S. M., Lu, Y., Molina, J., Heiser, L. M., Spellman, P. T., Speed, T. P., Gray, J. W., Mills, G. B., and Mukherjee, S. (2012). Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics*, **28**, 2804–2810.

Hill, S. M., Heiser, L. M., Cokelaer, T., Unger, M., Nesser, N. K., Carlin, D. E., Zhang, Y., Sokolov, A., Paull, E. O., Wong, C. K., Graim, K., Bivol, A., Wang, H., Zhu, F., Afsari, B., Danilova, L. V., Favorov, A. V., Lee, W. S., Taylor, D., Hu, C. W., Long, B. L., Noren, D. P., Bisberg, A. J., Consortium, H.-D., Gray, J. W., Kellen, M., Norman, T., Friend, S., Qutub, A. A., Fertig, E. J., Guan, Y., Song, M., Stuart, J. M., Spellman, P. T., Koeppl, H., Stovolitzky, G., Saez-Rodriguez, J., and Mukherjee, S. (2016). Inferring causal molecular networks: empirical assessment through a community-based effort. *Nature Methods*.

Hill, S. M., Nesser, N. K., Johnson-Camacho, K., Jeffress, M., Johnson, A., Boniface, C., Spencer, S. E., Lu, Y., Heiser, L. M., Lawrence, Y., Pande, N. T., Korkola, J. E., Gray, J. W., Mills, G. B., Mukherjee, S., and Spellman, P. T. (2017). Context specificity in causal signaling networks revealed by phosphoprotein profiling. *Cell Systems*, pages 73–83.

Hunter, T. (2009). Tyrosine phosphorylation: thirty years and counting. *Current Opinion in Cell Biology*, **21**(2), 140–146.

Kholodenko, B., Yaffe, M. B., and Kolch, W. (2012). Computational Approaches for Analyzing Information Flow in Biological Networks. *Science Signaling*, **5**(220), re1.

Kholodenko, B. N., Hancock, J. F., and Kolch, W. (2010). Signalling ballet in space and time. *Nature Reviews Molecular Cell Biology*, **11**(6), 414–426.

Köksal, A. S., Beck, K., Cronin, D. R., McKenna, A., Camp, N. D., Srivastava, S., MacGilvray, M. E., Bodik, R., Wolf-Yadlin, A., Fraenkel, E., Fisher, J., and Gitter, A. (2018). Synthesizing signaling pathways from temporal phosphoproteomic data. *Cell Reports*, **24**, 3607–3618.

Koster, J. and Rahmann, S. (2012). Snakemake–a scalable bioinformatics workflow engine. *Bioinformatics*, **28**(19), 2520–2522.

Krishnaswamy, S., Spitzer, M. H., Mingueneau, M., Bendall, S. C., Litvin, O., Stone, E., Pe'er, D., and Nolan, G. P. (2014). Conditional density-based analysis of T cell signaling in single-cell data. *Science*, **346**(6213), 1250689.

Molinelli, E. J., Korkut, A., Wang, W., Miller, M. L., Gauthier, N. P., Jing, X., Kaushik, P., He, Q., Mills, G., Solit, D. B., Pratilas, C. A., Weigt, M., Braunstein, A., Pagnani, A., Zecchina, R., and Sander, C. (2013). Perturbation biology: inferring signaling networks in cellular systems. *PLOS Computational Biology*, **9**.

Newman, R. H., Zhang, J., and Zhu, H. (2014). Toward a systems-level view of dynamic phosphorylation networks. *Frontiers in Genetics*, **5**, 263.

Norman, U. and Cicek, A. E. (2019). ST-Steiner: a spatio-temporal gene discovery algorithm. *Bioinformatics*, **35**(18), 3433–3440.

Oates, C. J., Korkola, J., Gray, J. W., and Mukherjee, S. (2014). Joint estimation of multiple related biological networks. *The Annals of Applied Statistics*, **8**, 1892–1919.

Patil, A., Kumagai, Y., Liang, K.-c., Suzuki, Y., and Nakai, K. (2013). Linking Transcriptional Changes over Time in Stimulated Dendritic Cells to Identify Gene Networks Activated during the Innate Immune Response. *PLOS Computational Biology*, **9**(11), e1003323.

Pawson, T. and Warner, N. (2007). Oncogenic re-wiring of cellular signaling pathways. *Oncogene*, **26**(9), 1268–1275.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, **308**(5721), 523–529.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, **2**, e55.

Schoeberl, B., Eichler-Jonsson, C., Gilles, E. D., and Müller, G. (2002). Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology*, **20**(4), 370–375.

Shojaie, A. and Michailidis, G. (2010). Discovering graphical Granger causality using the truncating lasso penalty. *Bioinformatics*, **26**(18), i517–i523.

Spencer, S. E., Hill, S. M., and Mukherjee, S. (2015). Inferring network structure from interventional time-course experiments. *The Annals of Applied Statistics*, **9**, 507–524.

Thain, D., Tannenbaum, T., and Livny, M. (2005). Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, **17**(2-4), 323–356.

Werhli, A. V. and Husmeier, D. (2007). Reconstructing gene regulatory networks with bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical Applications in Genetics and Molecular Biology*, **6**.

Zhang, Y. and Song, M. (2013). Deciphering interactions in causal networks without parametric assumptions.

Zou, H., Hastie, T., and Tibshirani, R. (2007). On the "degrees of freedom" of the lasso. *The Annals of Statistics*, **35**, 2173–2192.