Stable Differentiable Causal Discovery

Achille Nazaret [*1 2] **Justin Hong** [*1 2] **Elham Azizi** [1 2 3] **David Blei** [1 4]

Abstract

Inferring causal relationships as directed acyclic graphs (DAGs) is an important but challenging problem. Differentiable Causal Discovery (DCD) is a promising approach to this problem, framing the search as a continuous optimization. But existing DCD methods are numerically unstable, with poor performance beyond tens of variables. In this paper, we propose Stable Differentiable Causal Discovery (SDCD), a new method that improves previous DCD methods in two ways: (1) It employs an alternative constraint for acyclicity; this constraint is more stable, both theoretically and empirically, and fast to compute. (2) It uses a training procedure tailored for sparse causal graphs, which are common in real-world scenarios. We first derive SDCD and prove its stability and correctness. We then evaluate it with both observational and interventional data and in both small-scale and large-scale settings. We find that SDCD outperforms existing methods in convergence speed and accuracy, and can scale to thousands of variables.

1. Introduction

Inferring cause-and-effect relationships between variables is a fundamental challenge in many scientific fields, including biology (Sachs et al., 2005), climate science (Zhang et al., 2011), and economics (Hoover, 2006). Mathematically, a set of causal relations can be represented with a directed acyclic graph (DAG) where nodes are variables, and directed edges indicate direct causal effects. The goal of causal discovery is to recover the graph from the observed data. The data can either be interventional, where some variables were purposely manipulated, or purely observational, where there has been no manipulation.

The challenge of causal discovery is that searching for the true DAG underlying the data is an NP-hard problem. Exact methods are intractable, even for modest numbers of variables (Chickering, 1996). Yet datasets in fields like biology routinely involve thousands of variables (Dixit et al., 2016).

To address this problem, Zheng et al. (2018) introduced differentiable causal discovery (DCD), which formulates the DAG search as a continuous optimization over the space of all graph adjacency matrices. An essential element of this strategy is an acyclicity constraint, in the form of a penalty, that guides an otherwise unconstrained search toward acyclic graphs.

This optimization-oriented formulation often scales better than previous methods, and it has opened opportunities to harness neural networks (Lachapelle et al., 2019), incorporate interventional data (Brouillard et al., 2020), and use matrix approximation techniques (Lopez et al., 2022). But, while promising, existing DCD methods still struggle to scale consistently beyond tens of variables, or they rely on approximations that limit their applicability (see Section 5).

In this paper, we study the problems of DCD and improve on it, so that it can scale more easily and apply to many types of causal discovery problems. We trace the issues with DCD to the instability of its objective function; in particular, properties of the acyclicity constraint it uses to find a DAG solution. We formalize this notion of stability, show that previous DCD methods are unstable, and then formulate a method that is stable and scalable.

In details, this paper makes several contributions. First, we present a unifying theoretical view of existing acyclicity constraints, which explains their intrinsic numerical instability. We then employ a constraint, the *spectral acyclicity constraint* (Lee et al., 2019), that is both faster to compute and offers improved numerical stability. We prove its stability and corroborate its good properties with experiments.

Finally, we develop *Stable Differentiable Causal Discovery* (SDCD). SDCD is a two-stage optimization procedure for causal discovery that is stable and computationally efficient. In its first stage, it prunes edges without regard for acyclicity.

[*]Equal contribution [1]Department of Computer Science, Columbia University, New York, USA [2]Irving Institute for Cancer Dynamics, Columbia University, New York, USA [3]Department of Biomedical Engineering, Columbia University, New York, USA [4]Department of Statistics, Columbia University, New York, USA. Correspondence to: Elham Azizi <ea2690@columbia.edu>, David Blei <david.blei@columbia.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

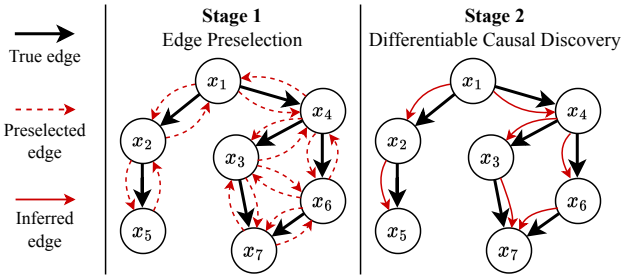

Figure 1: Visual representation of the SDCD method.

In the second stage, it performs DCD with the spectral acyclicity constraint described above. We prove that the first stage of SDCD does not remove true edges, and we show empirically that it is faster and more accurate than a single-stage optimization. SDCD removes key barriers that previously limited differentiable causal discovery to small problem sizes and application contexts.

In sum, the main contributions of this work are:

- We develop a theoretical analysis of the acyclicity constraints used in DCD, and their numerical instabilities.
- We motivate an alternative acyclicity constraint with superior stability, both theoretically and empirically.
- We propose the SDCD method for efficient DCD. It leverages the stable constraint within a two-stage optimization procedure designed for training robustness. We prove that SDCD does not compromise accuracy.
- We empirically study SDCD, and show that it efficiently solves problems involving thousands of variables. Compared to previous methods, SDCD achieves faster convergence and improved accuracy on observational and interventional data.
- Code is available at github.com/azizilab/sdcd.

Related Work. Causal discovery methods mainly fall into two categories: constraint-based methods and score-based methods (Glymour et al., 2019).

Constraint-based methods identify causal relationships by testing for conditional independence among variables in the data. For example, the PC algorithm (Spirtes et al., 2000) finds the graphs which conform to all the independencies present in the data. COmbINE is an extension to support interventional data (Triantafillou & Tsamardinos, 2015).

On the other hand, score-based methods design a score $S(G)$ that is maximized by the true graph G^*, and they aim to find its maximizer $\hat{G} = \arg\max_{G \in \text{DAG}} S(G)$. Existing score-based methods differ in their choice of S and of the optimization method to maximize it. GES (Chickering, 2002) and GDS (Peters & Bühlmann, 2014) optimize the BIC score of a Gaussian linear model by greedily adding or removing edges. GIES modifies GES to support interventional data (Hauser & Bühlmann, 2012), and CAM supports non-linear additive models (Bühlmann et al., 2014).

Differentiable Causal Discovery (DCD), which our work extends, is a type of score-based approach that reformulates the search of the score maximizer into a continuous optimization problem. It uses a numerical criterion to distinguish acyclic graphs from cyclic ones (called the acyclicity constraint). It was initially introduced in (Zheng et al., 2018) as NO-TEARS, which uses linear models, augmented Lagrangian optimization, and a constraint based on the adjacency matrix exponential. Other works extend the methodology to incorporate polynomial regression (Lee et al., 2019), neural networks (Lachapelle et al., 2019; Zheng et al., 2020), and support for interventional data (Brouillard et al., 2020).

Alternative acyclicity constraints (Lee et al., 2019; Ng et al., 2020; Bello et al., 2022) have been proposed, as well as optimization schemes different than augmented Lagrangian (Ng et al., 2020; 2022; Bello et al., 2022). Deng et al. (2023) introduced a hybrid approach combining gradient optimization with combinatorial optimization, while Lippe et al. (2021) explored removing the acyclicity constraint entirely.

Despite a rich literature, DCD has difficulty scaling to a large number of variables, exhibiting long training times and numerical instability. Wei et al. (2020); Ng et al. (2024) identified those limitations, and Lee et al. (2019); Lopez et al. (2022) addressed these problems with elegant approximations, but they resulted in poor accuracy.

Here, we provide a theoretical understanding of some algorithmic issues with DCD and then use that understanding to develop a better DCD method. Our proposed method builds on ideas that have been partially studied in previous work, including the acyclicity constraint of Lee et al. (2019) and the penalty method of Ng et al. (2020). Lee et al. (2019) uses the spectral acyclicity constraint for computational efficiency but otherwise does not expand on its advantages. In this work, we provide a novel analysis of the constraint and incorporate it into a new strategy that is more accurate and scalable than existing DCD algorithms. Table 1 recapitulates research in DCD and how this work fits in.

2. Background and Notations

We review the Differentiable Causal Discovery (DCD) approach and define the notations used in the paper.

2.1. Background on Causal Discovery

Causal discovery is the task of learning cause-and-effect relationships among a set of variables. In this work, we

Table 1: Comparison of Differentiable Causal Discovery methods including our proposed SDCD method. *Expressive model class* refers to the capability to approximate any causal graph with non-linear structural equations.

Method	Stable Training	Scalable Constraint	Can Use Interventions	Expressive Model Class
SDCD	✓	$\boldsymbol{O(d^2)}$	✓	✓
DCDI	×	$O(d^3)$	✓	✓
DCDFG	×	$O(md)$	✓	×
DAGMA	×	$O(d^3)$	×	✓
NO-TEARS	×	$O(d^3)$	×	×
NO-BEARS	✓	$\boldsymbol{O(d^2)}$	×	×

consider variables that can be intervened on such that they no longer are affected by their causal parents. These interventions are called *structural* or *perfect* interventions (Eberhardt & Scheines, 2007).

Causal Graphical Models. Causal graphical models (CGMs) provide a mathematical framework for reasoning about causal relationships between variables. Consider a CGM over d variables and K possible interventions on it.

There are three components:

1. A directed acyclic graph (DAG), $G^* = (V, E)$, where each node, $j \in V$, represents a variable x_j, and each edge, $(j, k) \in E$, indicates a direct causal relationship from x_j to x_k.
2. A list of conditional distributions, $p_j^*(x_j \mid x_{\mathrm{pa}_j^{G^*}}; 0)$, which specify the distribution of each x_j given its causal parents $x_{\mathrm{pa}_j^{G^*}}$ without intervention (the 0 indicates no intervention).
3. A list of interventions, $\mathcal{I} = \{I_0, I_1, ..., I_K\}$, where $I_0 = \varnothing$ (no intervention) and the others $I_k \subset V$ define the target variables of intervention k. Alongside is a list of interventional distributions, $p_m^*(x_m; k)$, for each $k > 0$ and $m \in I_k$, which define the distributions over x_m after intervention k.

The joint distribution under intervention k writes:

$$p^*(x; k) = \prod_{j \in V \setminus I_k} p_j^*(x_j \mid x_{\mathrm{pa}_j^{G^*}}; 0) \prod_{j \in I_k} p_j^*(x_j; k). \quad (1)$$

Note $p^*(x; 0)$ is the joint on observational data.

Data. We observe n data points of the d variables $X = \{(x_1^i, ..., x_d^i)\}_{i=1}^n$ with labels $T = \{t_i\}_{i=1}^n$ where $t^i \in \{0, ..., K\}$ indicates which intervention was applied to x^i (0 indicating no intervention). For example, in genomics, Perturb-seq screens (Replogle et al., 2022) measure the expression of d genes across n cells, where each cell can be edited once to change the expression of one of its genes.

Causal discovery with score-based methods. The goal of causal discovery is to infer the graph G^* from the data (X, T). In particular, score-based methods assign a score $S(G)$ to every possible graph G, where the score function is designed so that it is maximized on the true graph G^*. Score-based methods aim to find the maximizer

$$\hat{G} = \arg\max_{G \in \mathrm{DAG}} S(G). \quad (2)$$

Fix a model class that defines the conditional (and interventional) distributions for each possible DAG G as $\{p(\cdot \mid G; \theta, k)\}_\theta$. We can define the score $S(G)$ to be the maximum log-likelihood that can be achieved under graph G with some regularization over the number of edges $|G|$ (Chickering, 2002):

$$S_{\mathrm{mle}}(G) = \sup_\theta \left[\frac{1}{n} \sum_{i=1}^n \log p(x^i \mid G; \theta, t^i) \right] - \lambda |G|. \quad (3)$$

In the limit of infinite samples ($n \to \infty$), and under a few assumptions, any maximizer $\hat{G}$ of Equation (3) is close to the true G^* (Brouillard et al., 2020). More precisely, $\hat{G}$ and G^* are $\mathcal{I}$-*Markov-equivalent*: they share the same skeleton, v-structures, and other restrictive properties at the intervened variables in $\mathcal{I}$ (Yang et al., 2018).

2.2. Differentiable Causal Discovery

The main challenge to a score-based method for causal discovery is how to search over the large space of DAGs. Differentiable Causal Discovery (DCD) reformulates this combinatorial search into a continuous optimization problem over the space of all graphs, including cyclic ones (Zheng et al., 2018). It introduces three key components.

Model Class with Implicit Graph. First, DCD defines a model class with no apparent underlying graph, where each variable is conditioned on all others as $p(\cdot \mid \theta, k) \propto \prod p_j(x_j | x_{-j}; \theta, k)$. Instead, θ defines the graph G implicitly, such that if θ renders $x_{-j} \mapsto p_j(x_j | x_{-j}; \theta, 0)$ invariant to some $x_\ell \subset x_{-j}$, then there is no edge from l to j. The

induced adjacency matrix is denoted A_θ. When θ induces an acyclic A_θ, then $p(\cdot \mid \theta, k)$ defines a valid CGM.

Acyclicity Function. Second, DCD introduces a differentiable function $h(A_\theta)$ that quantifies how "cyclic" A_θ is. $h(A_\theta)$ is high when A_θ contains cycles with large edge weights, it is low when A_θ contains cycles with small weights, and $h(A_\theta) = 0$ when A_θ contains no cycles.

Optimization. Finally, DCD reformulates Equation (3) into a constrained optimization problem only over θ.

$$\hat{\theta} = \underset{\text{s.t. } h(A_\theta)=0}{\arg\max_{\theta}} \quad S_{\alpha,\beta}(\theta). \tag{4}$$

It uses A_θ in place of G, uses the constraint $h(A_\theta) = 0$ in place of $G \in \mathrm{DAG}$, and uses a new objective $S_{\alpha,\beta}$:

$$S_{\alpha,\beta}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p(x^i; \theta, t^i) - \alpha\|A_\theta\|_1 - \beta\|\theta\|_2^2, \tag{5}$$

$S_{\alpha,\beta}$ is a relaxed version of S_{mle} (Equation (3)) where the discrete $|G|$ is changed into an L1 regularization of A_θ (for $\alpha \geq 0$) and an L2 regularization of θ is included (for $\beta \geq 0$). The $\sup_\theta$ in S_{mle} (Equation (3)) is now removed, as it merges with the $\arg\max_\theta$ of Equation (4).

With Equation (4) in hand, different methods for DCD solve the constrained optimization in different ways. Some approaches use the augmented Lagrangian method (Zheng et al., 2018; Lachapelle et al., 2019; Brouillard et al., 2020; Lopez et al., 2022), some use the barrier method (Bello et al., 2022), and others use h as a regularizing penalty (Ng et al., 2020).

In all these approaches, the choices of h and the optimization method dictate the optimization behavior and the ultimate quality of the inferred graph. In the next section, we highlight the importance of h.

3. Stable Acyclicity Constraint

In this section, we demonstrate how most existing acyclicity constraints can lead to unstable numerical behaviors during optimization, especially with large numbers of variables d. We then motivate an alternative constraint, which we show to be theoretically and empirically more stable.

3.1. Power Series Trace Constraints

We first introduce a family of constraints. It generalizes existing constraints and reveals their similarities.

Definition 3.1 (The Power Series Trace Family)**.** For any non-negative coefficients $(a_k)_{k\in\mathbb{N}^*} \in \mathbb{R}_{\geq 0}^{\mathbb{N}^*}$, consider the power series $f_a(x) = \sum_{k=1}^{\infty} a_k x^k$.

Name	a_k	f_a	h_a	$\nabla h_a^\top$
$h_{\exp}$	$1/k!$	$\exp(x) - 1$	$\mathrm{Tr}\exp(A) - d$	$\exp(A)$
$h_{\log}$	$1/k$	$\log\frac{1}{1-x}$	$-\log\det(I - A)$	$(I - A)^{-1}$
h_{inv}	1	$\frac{1}{1-x}$	$\mathrm{Tr}(I - A)^{-1}$	$(I - A)^{-2}$
h_{binom}	$\binom{d}{k}$	$(1 + x)^d - 1$	$\mathrm{Tr}(I + A)^d - d$	$d(I + A)^{d-1}$
h_ρ	–	–	$\lvert\lambda_d(A)\rvert$	$v_d u_d^\top / v_d^\top u_d$

Table 2: (Top) Existing PST constraints with their power series and gradients. (Bottom) The spectral acyclicity constraint, which is not PST.

Then, for any matrix $A \in \mathbb{R}_{\geq 0}^{d\times d}$ with non-negative entries, we define the Power Series Trace (PST) function

$$h_a(A) = \mathrm{Tr}\left[f_a(A)\right] = \sum_{k=1}^{\infty} a_k \mathrm{Tr}\left[A^k\right].$$

The quantity $h_a(A)$ is closely related to the cycles in the graph represented by A. In $h_a(A)$, each $\mathrm{Tr}\left[A^k\right]$ equals the total weight of all length-k cycles in A – where the weight of a cycle is the product of its edge weights (Bapat, 2010). The next theorem generalizes the result of Wei et al. (2020) to show that most h_a can be used to characterize acyclicity.

Theorem 3.2 (PST constraint)**.** *For any sequence $(a_k)_{k\in\mathbb{N}^*} \in \mathbb{R}_{\geq 0}^{\mathbb{N}^*}$, if we have $a_k > 0$ for all $k \in [\![1, d]\!]$, then, for any matrix $A \in \mathbb{R}_{\geq 0}^{d\times d}$, we have*

$$\begin{cases} h_a(A) = 0 \Leftrightarrow A \text{ is acyclic}, \\ h_a(A) \geq 0, \\ \nabla h_a(A) = h_{a'}(A^\top) \text{ with } a'_k = (k+1)a_{k+1}. \end{cases}$$

We say that h_a is a PST constraint.

The proof is in Appendix A.1. In particular, sequences of strictly positive a_k satisfy the conditions for any d, so several standard power series are PST constraints.

For example, the sequence $a_k^{\exp} = \frac{1}{k!}$ recovers the penalty $h_{\exp}(A) = \mathrm{Tr}(\exp(A)) - d$ originally proposed in Zheng et al. (2018).

If we define $a_k^{\log} = \frac{1}{k}$, then $f_{a^{\log}} = \sum_{k=1}^{\infty} \frac{x^k}{k}$ is the power series of $x \mapsto -\log(1 - x)$. With the identity $\mathrm{Tr}\log A = \log\det A$ (Withers & Nadarajah, 2010), where $\log A$ is the matrix logarithm, we find that

$$h_{\log}(A) = \mathrm{Tr}(-\log(I - A)) = -\log\det(I - A).$$

This is precisely the constraint introduced in Ng et al. (2020); Bello et al. (2022). Hence, even though it uses the matrix determinant instead of the matrix trace, we uncover that $h_{\log}$ is also a PST constraint.

Table 2 shows that other constraints such as $h_{\mathrm{binom}} = \mathrm{Tr}((I + A)^d) - d$ (Yu et al., 2019), $h_{\mathrm{inv}} = \mathrm{Tr}((I - A)^{-1}) - d$ (Zheng et al., 2018) are also PST.

3.2. Limitations of PST constraints

In this section, we provide the criteria necessary for constraints to exhibit stable optimization behavior. We prove that PST constraints do not satisfy these criteria and show empirically that optimization with these constraints can be slow or fail. As a solution, we suggest an alternative, non-PST acyclicity constraint and demonstrate its stability.

Definition 3.3. An acyclicity constraint h is stable if these three criteria hold for almost every $A \in \mathbb{R}_{\geq 0}^{d\times d}$:

- **E-stable.** $h(sA) = O_{s\to\infty}(s)$
- **V-stable.** $h(A) \neq 0 \Rightarrow h(\varepsilon A) = \Omega_{\varepsilon\to 0^+}(\varepsilon)$
- **D-stable.** h and ∇h are defined almost everywhere.

E-stability ensures that h does not *explode* to infinity; *V-stability* ensures h does not *vanish* rapidly to 0; *D-stability* ensures that h and its gradient are well *defined*.

These three criteria are all important for maximizing $S_{\alpha,\beta}(\theta)$ under the constraint $h(A_\theta) = 0$. D-stability and E-stability ensure the constraint remains well-defined and with bounded values throughout the optimization procedure. The V-stability is related to the nature of constrained optimization. Methods like augmented Lagrangian, barrier functions, and penalties use the constraint h to formulate an objective of the form $S_{\alpha,\beta}(\theta) - \gamma h(A_\theta) - \mu h(A_\theta)^2$. They then increase γ and μ until $h(A_\theta)$ reaches 0. These increments ensure that the penalty does not become negligible relative to $S_{\alpha,\beta}(\theta)$. But without V-stability, $h(A_\theta)$ can shrink quickly very close to 0, while A_θ remains far from a DAG. So, for full convergence, these methods must grow γ and μ to large values, which can be either inefficient (it requires more training epochs) or fail (as γ or $h(A_\theta)$ eventually reach the limit of machine precision). The studies in Section 5.2 demonstrate these failure modes happen in practice.

We now show that PST constraints are unstable, especially as d grows.

Theorem 3.4 (PST instability)**.** *For $d \geq 2$, any PST constraint h is both E-unstable and V-unstable. More precisely,*

- ***E-unstable.*** $\exists A \in \mathbb{R}_{\geq 0}^{d\times d}, h(sA) = \Omega_{s\to\infty}(s^d)$
- ***V-unstable.*** $\exists A \in \mathbb{R}_{\geq 0}^{d\times d}, h(\varepsilon A) = O_{\varepsilon\to 0^+}(\varepsilon^d)$

*Also, any PST constraint for which f_a has a finite radius of convergence is **D-unstable** (e.g., $h_{\log}, h_{\text{inv}}$).*

Theorem 3.4 is proved in Appendix A.2. It shows that the instability of the PST constraints worsens exponentially in d. Figure 2 empirically corroborates the theorem with two types of adjacency matrices encountered during DCD: a cycle and some uniformly random noise. It shows that all PST constraints escalate to infinity or vanish to zero as the scale of noise ε changes (Figure 2 left) or as the number of variables d increases (Figure 2 right), reflecting their E-instability and V-instability. In addition, the D-instability of

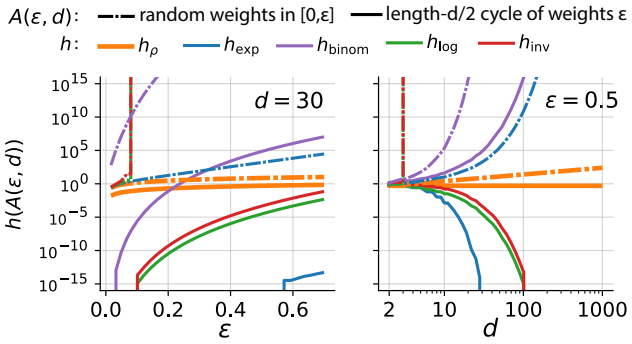

Figure 2: Constraint behaviors when evaluated on uniform random matrices in $[0, \epsilon]^{d\times d}$ (dashed) or a cycle of length $d/2$ with weight ε (solid). The y-axis shows the constraint's value, the x-axis is (left) the weights' scale ε (right) the number of variables d. Only the proposed h_ρ (orange) remains stable; others vanish to zero exponentially or escalate to infinity (as soon as $d > 10$). The vertical dotted lines indicate the constraint escaped its domain of definition. All these failures were encountered during DCD experiments.

$h_{\log}$ and h_{inv} appears even in small ϵ or d (vertical lines). We encounter all three instabilities during causal discovery experiments (Section 5.2), leading existing approaches to fail.

3.3. The Spectral Acyclicity Constraint

To overcome the limits of the PST constraint family, we propose to use another type of constraint, one based on the spectrum of A, which was first used in Lee et al. (2019). This constraint draws from a characterization of DAG matrices from graph theory - that A is acyclic if and only if all its eigenvalues are zero (Cvetković et al., 1980).

We write $\lambda_1(A) \in \mathbb{C}$ to $\lambda_d(A) \in \mathbb{C}$, the d eigenvalues of A, sorted from smallest to highest complex magnitude

Definition 3.5 (Spectral radius)**.** The spectral radius

$$h_\rho(A) = |\lambda_d(A)|,$$

is the largest eigenvalue magnitude of A.

The next theorem shows that the spectral radius can be used as an acyclicity constraint.

Theorem 3.6 (Cvetković et al. (1980); Lee et al. (2019))**.** *The spectral radius is an acyclicity constraint.*

$$h_\rho(A) = 0 \Leftrightarrow A \text{ is a DAG.}$$

We refer to it as the spectral acyclicity constraint. It is differentiable almost everywhere, with gradient

$$\nabla h_\rho(A) = v_d u_d^\top / v_d^\top u_d,$$

where u_d, v_d are respectively the right and left eigenvectors associated with $\lambda_d(A)$ (Magnus, 1985).

Theorem 3.6 is proved in Appendix A.3. It implies that h_ρ is D-stable. Next, we prove h_ρ is E-and-V-stable.

Theorem 3.7. *h_ρ is stable.*

We refer to Appendix A.4 for the proof.

Remark 3.8. As a corollary of Theorem 3.7, h_ρ is not another PST constraint (since it is stable).

We complete Figure 2 with the empirical behavior of h_ρ. As theoretically expected, h_ρ retains non-extreme values and is suitable for constraint-based optimization.

To further understand the impact of the constraints' stability on optimization, we empirically study the optimization path of the augmented Lagrangian and the penalty method with each constraint in Appendix Figure 5. The instabilities of PST constraints effectively slow their convergence and require increasing γ and μ to excessively large values. In contrast, the optimization paths with h_ρ take the least number of iterations to converge, especially with the penalty method. Moreover, the computation of h_ρ can be done in $O(d^2)$ time (See Appendix B.2), contrary to the PST constraints whose computations scale in $O(d^3)$.

We are ready to perform DCD with the stable h_ρ.

4. Stable Differentiable Causal Discovery

With the stable acyclicity constraint h_ρ in hand, we now introduce Stable Differentiable Causal Discovery (SDCD). SDCD efficiently learns causal graphs in two stages.

4.1. The SDCD method

To solve the optimization problem (4) with the spectral acyclicity constraint h_ρ, SDCD optimizes the following objective with gradient-based optimization:

$$\hat{\theta} = \arg\max_{\theta} S_{\alpha,\beta}(\theta) - \gamma \cdot h_\rho(A_\theta), \tag{6}$$

where h_ρ is used as a penalty with coefficient γ. SDCD proceeds in two stages (See Figure 1).

Stage 1: Edge Preselection. First, SDCD solves Equation (6) without the constraint, by setting $\gamma = 0$.

$$\hat{\theta}_1 = \arg\max_{\theta \atop \forall j, A_{\theta,jj}=0} S_{\alpha_1,\beta_1}(\theta) \tag{7}$$

This stage amounts to solving simultaneously d independent prediction problems of each variable given the others (the constraint $A_{\theta,jj} = 0$ prevents self-loops). The goal is to identify nonpredictive edges and remove them in stage 2, akin to *feature selection*.

SDCD selects the *removed edges* as $\hat{R}_1 = \{(j,l) \in [\![1,d]\!]^2 \mid A_{\hat{\theta}_1,jl} < \tau_1\}$ where τ_1 is a threshold.

Stage 2: Differentiable Causal Discovery. Next, SDCD re-solves Equation (6), this time with the constraint and with masking the removed edges from stage 1,

$$\hat{\theta}_2 = \arg\max_{\theta \atop \forall (j,l) \in \hat{R}_1, A_{\theta,jl}=0} S_{\alpha_2,\beta_2}(\theta) - \gamma_2 h_\rho(A_\theta). \tag{8}$$

The term γ_2 is initialized at 0 and is increased by a constant, γ_δ, after each epoch. Like other DCD methods, SDCD forms the final graph $\hat{G}_{\text{SDCD}}$ by selecting the edges in $A_{\hat{\theta}_2}$ with weight above a threshold τ_2. The details of the algorithm can be found in Appendix B.2.

Remark 4.1. In both stages, the constraints $A_{\theta,jl} = 0$ are straightforward to enforce by masking the elements in θ corresponding to $A_{\theta,jl}$ (i.e., fixing them at 0).

Compared to other methods, SDCD innovates in two ways: (1) by using the constraint h_ρ with the penalty method and (2) by using a two-stage optimization that preselects edges in stage 1 and optimize the DCD objective only on those in stage 2. Without explicit masking, stage 2 would be similar to the barrier or penalty method (Ng et al., 2020; Bello et al., 2022), with stage 1 only providing a warm start.

Motivations for stage 1. Dedicating stage 1 to removing unlikely edges is motivated by the hypothesis that real-life causal graphs are sparse. For example, individual genes in biological systems are typically regulated by a few other genes rather than all other genes (Lambert et al., 2018). A similar hypothesis underlies work in sparse mechanism shift (Schölkopf et al., 2021). Hence, stage 1 will likely remove many false edges and facilitate stage 2. Alternative approaches for variable selection (e.g., markov boundaries (Loh & Bühlmann, 2014; Wang & Chan, 2012), skeletons (Tsamardinos et al., 2006), preliminary neighborhood selection (Bühlmann et al., 2014; Lachapelle et al., 2019)) can be motivated for the same reasons. Here, we found a simple modification to the objective function can effectively serve this purpose. In practice, we find that stage 1 improves convergence speed and accuracy (Section 5, Table 7). Notably, we find that stage 1 improves the stability of the training in stage 2, even when PST constraints are used in place of the spectral one (Table 8). For this reason, stage 1 may also serve as a beneficial preprocessing step for other causal discovery methods. In Theorem 4.2 below, we prove that stage 1 does not remove true causal parents.

4.2. Theoretical guarantees

We analyze SDCD's time complexity in Appendix B.2.3. We now provide correctness guarantees for the two stages of SDCD. We show that theoretically, stage 1 does not remove true causal parents, and so, stage 2 returns an optimal graph.

As done in the field (e.g., Chickering (2002); Brouillard et al. (2020)), the results focus on the "theoretical" $\hat{G}$ that

would be obtained with infinite data and if Equations (7) and (8) were solved exactly, in their non-relaxed form. We study SDCD in practice in Section 5.

With infinite data, Equation (7)'s unrelaxed version writes,

$$\tilde{\theta}_1 = \arg\max_{\substack{\theta \\ \forall j, A_{\theta,jj}=0}} \sum_{\substack{k=0 \\ I_k \not\ni j}}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} [\log p_j(x_j | x_{-j}; \theta, 0)] - \lambda |A_\theta|, \quad (9)$$

where π_k is the proportion of data coming from intervention k. The next theorem characterizes the graph $\tilde{G}_1 = A_{\tilde{\theta}_1}$ in terms of Markov boundaries in the true graph G^*. A Markov boundary for j is a minimal set of variables that render j independent of all the others. In a causal graph, each j has a unique Markov boundary, consisting of j's parents, j's children, and j's children's parents (Neapolitan et al., 2004).

Theorem 4.2. *Under regularity assumptions detailed in Appendix A.5, the candidate parents* $\mathrm{pa}_j^{\tilde{G}_1}$ *of* j *selected by stage 1 are precisely the Markov boundary of* j *in the true graph* G^**, That is,* $\mathrm{pa}_j^{\tilde{G}_1} = \mathrm{pa}_j^{G^*} \cup \mathrm{ch}_j^{G^*} \cup \mathrm{pa}_{\mathrm{ch}_j^{G^*}}^{G^*}$.

The assumptions of Theorem 4.2 and its proof are detailed in Appendix A.5. The assumptions are reasonable: p^* should be in the model class $\{p_\theta\}$, the expectations should be well defined, and "faithfulness" should hold (that is, G^* doesn't have superfluous edges).

Theorem 4.2 gives two guarantees: (1) stage 1 does not remove causal parents and (2) stage 1 returns only a subset of the edges, not all of them. For instance, if G^* is sparse such that each node has at most k parents, then only $O(dk^2)$ edges are returned, which is essentially linear in d for small k (see Appendix A.7).

Theorem 4.2 implies that Brouillard et al. (2020, Theorem 1) still applies, and we deduce that stage 2 remains optimal under the stated assumptions (see Appendix A.6).

The theoretical results are reassuring. In the next section, we study SDCD's empirical performance to examine the impact of finite data, nonconvex optimization, and relaxations.

5. Empirical studies

We compare SDCD to state-of-the-art baselines on multiple datasets. We find that SDCD achieves significantly better scores in both observational and interventional settings, particularly excelling at recovering sparser graphs. SDCD is the only method to scale to thousands of variables without sacrificing accuracy.

5.1. Evaluation Setup

Baselines for interventional data. For datasets with interventional data, we compare SDCD against DCDI (Brouillard et al., 2020), DCD-FG (Lopez et al., 2022), and GIES (Hauser & Bühlmann, 2012).

Baselines for observational data. When the dataset contains only observational data, we include the interventional methods and further compare against NO-TEARS (Zheng et al., 2018), NO-BEARS (Lee et al., 2019), DAGMA (Bello et al., 2022), and SCORE (Rolland et al., 2022). In addition, we report *sortnregress* (Reisach et al., 2021), a trivial baseline that should be outperformed (see Robustness Checks). We further included NOCURL (Yu et al., 2021) and AVICI (Lorch et al., 2022) in Appendix D.

Metrics. We evaluate performance using the structural Hamming distance (SHD) between the true G^* and each method's output graph. SHD is standard in causal discovery. It quantifies the minimum number of edge additions, deletions, and reversals needed to transform one graph into the other. Lower SHDs indicate better reconstructions of G^*.

Robustness Checks. Previous works detailed common issues with the SHD metric (Tsamardinos et al., 2006) and data simulation processes (Reisach et al., 2021). We include additional metrics and baselines recommended by previous works to ensure our evaluation is robust. Further details and results are detailed in Appendix C.4.

Data. We simulate observational and interventional data for a wide range of d (number of variables), varying the graph density with s (the average number of parents per node), and varying the number of variables that are intervened on. The simulations proceed as done in Brouillard et al. (2020); Bello et al. (2022), by sampling a random graph, modeling its conditionals with random neural networks, setting its interventional distribution to Gaussian, and drawing samples from the obtained model. More details are in Appendix C.1. In all experiments, the number of observational samples is fixed at $10,000$, and an additional 500 samples are added for each perturbed variable.

To further validate the results against the strongest baseline, we evaluate SDCD on the simulated data generated in Brouillard et al. (2020) (DCDI) and compare our results against their reported SHD values.

Setting. Consistent with prior work (e.g., DAGMA, NOTEARS), we do not conduct hyperparameter optimization for the experiments. Instead, we fix a single set of parameters for all experiments (see Appendix C.2). The training time on CPU is measured on an AMD 3960x with 4-core per method; on GPU on an AMD 3960x with 16-core and an Nvidia A5000.

SDCD Modeling Assumptions. We use neural networks (NNs) to parameterize the model class, as done in Lachapelle et al. (2019); Zheng et al. (2020). Each $p_j(x_j \mid x_{-j}; \theta, k)$ is a Gaussian distribution over x_j with

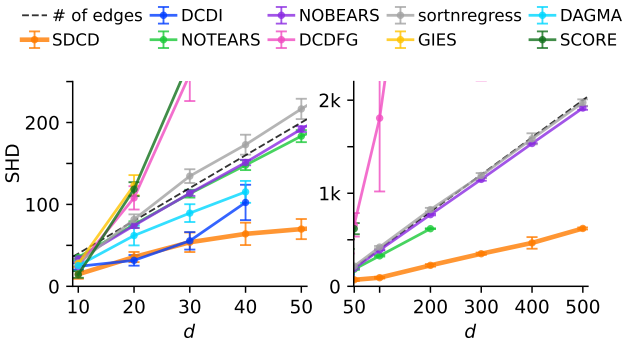

Figure 3: SHD across simulations on observational data with increasing numbers of variables d. SDCD achieves the best SHDs. It is the only method scaling above 200 variables with nontrivial SHD. Missing data points imply the method failed to run. Error bars indicate std on 30 random datasets for $d \leq 50$ and five for $d > 50$ (175 total). Lower is better.

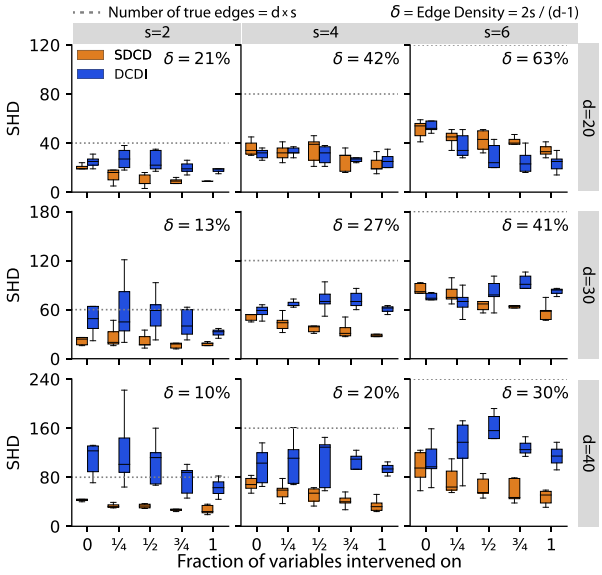

Figure 4: SHD across simulations with an increasing proportion of variables intervened on, varying the total number of variables d (columns) and average edges per variable s (rows). SDCD is the only method to consistently improve with interventional data and has the best SHDs for sparse graphs (edge density $\delta \leq 45\%$). Each boxplot over 5 random datasets (45 datasets total).

mean and variance given by an NN as a function of all the other x_{-j}. More details about the NN architecture are in Appendix B.1. Also, SDCD is amenable to other model classes, such as normalizing flows (Brouillard et al., 2020).

5.2. Observational Data Experiments

We evaluate all eight methods on a wide range of number of variables d, with a fixed average number of edges per variable $s = 4$, and repeat the experiments over 30 random datasets. Figure 3 reports the results and detailed tables are provided in Appendix D with additional baselines.

SDCD outperforms the other methods in accuracy at every scale and speed. It can be explained by SDCD's stability.

Failures of other methods. DCDI is competitive on small d but crashes for $d > 40$ – as discussed in 3.2, for $d = 50$, NaNs appear during training when h_{exp} underflows due to V-instability; for $d > 50$ NaNs appear right at initialization when h_{exp} overflows due to E-instability. DAGMA fails to converge within 6 hours for as few as 30 variables due to the learned adjacency matrix escaping the domain of definition of $h_{\log}$, caused by D-instability. DAGMA attempts to stay within the domain of definition of $h_{\log}$ by reducing the learning rate near the singularities, but this is often not sufficient and it significantly slows down training. NO-TEARS and NO-BEARS perform similarly to the trivial baseline *sortnregress*, confirming the findings of Reisach et al. (2021). DCD-FG scales well but has exceptionally high SHD due to predicting very dense graphs – which we attribute to its low-rank approximation.

Finally, we note that most methods outperform the SHD of the empty graph (it is the number of edges, as dashed line).

To show that SDCD's performance is robust to a comprehensive set of scenarios, we provide additional metrics for these experiments in Appendix C.4 and Figures 7 and 10.

The runtimes associated with Figure 3 are presented in Appendix Figure 6. SDCD-GPU runs under 15 minutes for all values of d in Figure 3 experiments (e.g., $d = 500$). In Appendix Figure 8, we further demonstrate that SDCD can scale up to 4,000 variables under 2h45.

5.3. Interventional Data Experiments

Next, we compare SDCD, DCDI, DCD-FG, and GIES over datasets with an increasing proportion of intervened variables. We show the results for SDCD and DCDI in Figure 4 and all methods in the Appendix (DCD-FG and GIES performed consistently worse). As expected, the methods generally improve with more interventional data, although SDCD is the only method to do so consistently. We find that SDCD performs the best in most scenarios, particularly on sparser graphs. We characterize the edge density of a graph, δ, as the ratio of true edges to the maximum number of edges possible in a DAG.

5.4. Ablation Experiments

We performed ablation studies to judge the impact of each innovation implemented in SDCD. We evaluated modifications of SDCD where (1) only stage 2 is performed without

stage 1 and where (2) in stage 2, the spectral constraint is substituted for alternative PST acyclicity constraints. As the results show in Tables 7 and 8, both stages are essential to the success of SDCD.

5.5. Experiment against the best baseline

In Supplementary Table 3 we report the results of SDCD on the simulated data presented in (Brouillard et al., 2020) alongside their original DCDI results. SDCD outperforms DCDI on all its sparse datasets ($s = 1$). Only for datasets where $d = 10, s = 4$, does SDCD perform worse than DCDI. However, we find the edge density ($\delta = 88.9\%$) of these graphs to be unrepresentative of realistic scenarios.

6. Conclusion

With SDCD, we addressed the limitations of existing DCD methods by applying an acyclicity constraint and a two-stage procedure that each promotes stability. We show it improves in all regimes and can scale to thousands of variables, enabling new applications for DCD in data-rich settings.

Future work could aim to provide a deeper theoretical understanding of the impact of the acyclicity constraint's stability on gradient-based optimization, particularly how the constraint's non-convexity affects training.

Acknowledgments

A.N. was supported by funding from the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard, and the Africk Family Fund. J.H. was supported by grant number 2022-253560 from the Chan Zuckerberg Initiative DAF, an advised fund of the Silicon Valley Community Foundation, and the Irving Institute for Cancer Dynamics. E.A. was supported by the National Institute of Health (NIH) NCI grant R00CA230195 and NHGRI grant R01HG012875. D.B. was funded by NSF 2127869, NSF 2311108, ONR N00014-17-1-2131, ONR N00014-15-1-2209, the Simons Foundation, and Open Philanthropy.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Bapat, R. B. *Graphs and matrices*, volume 27. Springer, 2010.

Bello, K., Aragam, B., and Ravikumar, P. DAGMA: Learning DAGs via M-matrices and a Log-Determinant Acyclicity Characterization. In *Neural Information Processing Systems*, 2022.

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. Differentiable causal discovery from interventional data. In *Neural Information Processing Systems*, 2020.

Bühlmann, P., Peters, J., and Ernest, J. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014.

Chickering, D. M. Learning Bayesian networks is NP-complete. *Learning from Data: Artificial Intelligence and Statistics V*, pp. 121–130, 1996.

Chickering, D. M. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3 (Nov):507–554, 2002.

Cvetković, D. M., Doob, M., and Sachs, H. *Spectra of Graphs: Theory and Application*. Academic Press, 1980.

Deng, C., Bello, K., Aragam, B., and Ravikumar, P. K. Optimizing notears objectives via topological swaps. In *International Conference on Machine Learning*, pp. 7563–7595. PMLR, 2023.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., et al. Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. *Cell*, 167(7):1853–1866, 2016.

Eberhardt, F. and Scheines, R. Interventions and causal inference. *Philosophy of Science*, 74(5):981–995, 2007.

Glymour, C., Zhang, K., and Spirtes, P. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

Hauser, A. and Bühlmann, P. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13(1):2409–2464, 2012.

Hoover, K. D. *Causality in Economics and Econometrics*, pp. 1–13. Palgrave Macmillan UK, 2006.

Horn, R. A. and Johnson, C. R. *Matrix analysis*. Cambridge university press, 2012.

Lachapelle, S., Brouillard, P., Deleu, T., and Lacoste-Julien, S. Gradient-based neural DAG learning. In *International Conference on Learning Representations*, 2019.

Lambert, S. A., Jolma, A., Campitelli, L. F., Das, P. K., Yin, Y., Albu, M., Chen, X., Taipale, J., Hughes, T. R., and Weirauch, M. T. The human transcription factors. *Cell*, 172(4):650–665, 2018.

Lee, H.-C., Danieletto, M., Miotto, R., Cherng, S. T., and Dudley, J. T. Scaling structural learning with NO-BEARS to infer causal transcriptome networks. In *Pacific Symposium on Biocomputing*, 2019.

Lippe, P., Cohen, T., and Gavves, E. Efficient neural causal discovery without acyclicity constraints. *arXiv preprint arXiv:2107.10483*, 2021.

Loh, P.-L. and Bühlmann, P. High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research*, 15(1):3065–3105, 2014.

Lopez, R., Hütter, J.-C., Pritchard, J., and Regev, A. Large-scale differentiable causal discovery of factor graphs. In *Neural Information Processing Systems*, 2022.

Lorch, L., Sussex, S., Rothfuss, J., Krause, A., and Schölkopf, B. Amortized inference for causal structure learning. *Advances in Neural Information Processing Systems*, 35:13104–13118, 2022.

Magnus, J. R. On differentiating eigenvalues and eigenvectors. *Econometric theory*, 1(2):179–191, 1985.

Neapolitan, R. E. et al. *Learning Bayesian Networks*. Prentice Hall, 2004.

Ng, I., Ghassami, A., and Zhang, K. On the role of sparsity and DAG constraints for learning linear DAGs. In *Neural Information Processing Systems*, 2020.

Ng, I., Lachapelle, S., Ke, N. R., Lacoste-Julien, S., and Zhang, K. On the convergence of continuous constrained optimization for structure learning. In *International Conference on Artificial Intelligence and Statistics*, 2022.

Ng, I., Huang, B., and Zhang, K. Structure learning with continuous optimization: A sober look and beyond. In *Causal Learning and Reasoning*, pp. 71–105. PMLR, 2024.

Peters, J. and Bühlmann, P. Identifiability of gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2014.

Peters, J., Mooij, J. M., Janzing, D., and Schölkopf, B. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 15:2009–2053, 2014.

Reisach, A. G., Seiler, C., and Weichwald, S. Beware of the simulated DAG! causal discovery benchmarks may be easy to game. In *Neural Information Processing Systems*, 2021.

Replogle, J. M., Saunders, R. A., Pogson, A. N., Hussmann, J. A., Lenail, A., Guna, A., Mascibroda, L., Wagner, E. J., Adelman, K., Lithwick-Yanai, G., et al. Mapping information-rich genotype-phenotype landscapes with genome-scale perturb-seq. *Cell*, 185(14):2559–2575, 2022.

Rolland, P., Cevher, V., Kleindessner, M., Russell, C., Janzing, D., Schölkopf, B., and Locatello, F. Score matching enables causal discovery of nonlinear additive noise models. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18741–18753. PMLR, 2022.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, pp. 523–529, 2005.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. Toward causal representation learning. *Proceedings of the IEEE*, pp. 612–634, 2021.

Spirtes, P., Glymour, C. N., and Scheines, R. *Causation, prediction, and search*. MIT press, 2000.

Triantafillou, S. and Tsamardinos, I. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *Journal of Machine Learning Research*, 16(1):2147–2205, 2015.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65:31–78, 2006.

Wang, Z. and Chan, L. Learning bayesian networks from markov random fields: an efficient algorithm for linear models. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(3):1–31, 2012.

Wei, D., Gao, T., and Yu, Y. Dags with no fears: A closer look at continuous optimization for learning bayesian networks. *Advances in Neural Information Processing Systems*, 33:3895–3906, 2020.

Withers, C. S. and Nadarajah, S. Log Det A= Tr Log A. *International Journal of Mathematical Education in Science and Technology*, 2010.

Yang, K., Katcoff, A., and Uhler, C. Characterizing and learning equivalence classes of causal dags under interventions. In *International Conference on Machine Learning*, 2018.

Yu, Y., Chen, J., Gao, T., and Yu, M. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, 2019.

Yu, Y., Gao, T., Yin, N., and Ji, Q. Dags with no curl: An efficient dag structure learning approach. In *International Conference on Machine Learning*, pp. 12156–12166. Pmlr, 2021.

Zhang, D. D., Lee, H. F., Wang, C., Li, B., Pei, Q., Zhang, J., and An, Y. The causality analysis of climate change and large-scale human crisis. In *Proceedings of the National Academy of Sciences*, 2011.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. DAGs with NO TEARS: Continuous optimization for structure learning. In *Neural Information Processing Systems*, 2018.

Zheng, X., Dan, C., Aragam, B., Ravikumar, P., and Xing, E. Learning sparse nonparametric dags. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Supplementary Materials: Stable Differentiable Causal Discovery

A. Theoretical Results

A.1. Proof of Theorem 3.2

Before proving Theorem 3.2, we precisely define an acyclic matrix and prove a few lemmas.

Definition A.1 (Cyclic and acyclic matrices)**.** Take a matrix $A \in \mathbb{R}_{\geq 0}^{d \times d}$.

We say that A has a *cycle of length* k if and only if:

$$\exists (i_0, ..., i_k) \in [\![1, k]\!]^{k+1}, \text{ such that, } \begin{cases} i_0 = i_k \\ \forall \ell \in [\![1, k]\!], A_{i_{\ell-1}, i_\ell} > 0 \end{cases} \tag{10}$$

We say that A is *cyclic* if it contains at least one cycle. We note that if A contains a cycle of length k for $k \in \mathbb{N}^*$ (the set of strictly positive integers), then A also contains a cycle of length k' for $k' \in [\![1, d]\!]$ (this follows from the pigeon hole principle).

We say that A is *acyclic* if it does not contain any cycle (or equivalently if it does not contain any cycle of length $k \leq d$).

Lemma A.2. *For any matrix* $A \in \mathbb{R}_{\geq 0}^{d \times d}$*,*

- $\text{Tr}\left[A^k\right] \geq 0$ *for any* k
- A *has a cycle of length* k *if and only if* $\text{Tr}\left[A^k\right] > 0$
- $A^d = 0$ *if and only if* A *is acyclic.*

Proof. Fix a matrix $A \in \mathbb{R}_{\geq 0}^{d \times d}$. We have,

$$\text{Tr}\left[A^k\right] = \sum_{\substack{(i_0, ..., i_k) \in [\![1, d]\!]^{k+1} \\ i_0 = i_k = i}} \prod_{\ell=1}^{k} A_{\ell-1, \ell}. \tag{11}$$

Each addend is non-negative so $\text{Tr}\left[A^k\right] \geq 0$. Furthermore, the total sum is strictly positive if and only if at least one addend is strictly positive. This happens if and only if A has a cycle of length k by definition.

Similarly, we have

$$(A^d)_{i,j} = \sum_{\substack{(i_0, ..., i_d) \in [\![1, d]\!]^{d+1} \\ i_0 = i_d = j}} \prod_{\ell=1}^{k} A_{\ell-1, \ell}. \tag{12}$$

If $A^d_{i,j} > 0$, then one addend is strictly positive and so there exists $(i_0, ..., i_d) \in [\![1, d]\!]^{d+1}$ such that $i_0 = i_d = j$ and $\prod_{\ell=1}^{k} A_{\ell-1, \ell} > 0$. By the pigeon-hole principle, two i_ℓ are identical, which provides a cycle. Reciprocally, if $(i_0, ..., i_k)$ is a cycle of length k, then by repeating $(i_0, ..., i_k, i_1, i_2 ... i_{d \mod k})$ until having a path of length $d + 1$, we have that $(A^d)_{i_0, i_{d \mod k}} > 0$. Hence, $A^d = 0$ if and only if A is acyclic. □

We recall Theorem 3.2.

Theorem 3.2 (PST constraint)**.** *For any sequence $(a_k)_{k\in\mathbb{N}^*} \in \mathbb{R}_{\geq 0}^{\mathbb{N}^*}$, if we have $a_k > 0$ for all $k \in [\![1, d]\!]$, then, for any matrix $A \in \mathbb{R}_{\geq 0}^{d\times d}$, we have*

$$\begin{cases} h_a(A) = 0 \Leftrightarrow A \text{ is acyclic}, \\ h_a(A) \geq 0, \\ \nabla h_a(A) = h_{a'}(A^\top) \text{ with } a'_k = (k+1)a_{k+1}. \end{cases}$$

We say that h_a is a PST constraint.

Proof. Fix a matrix $A \in \mathbb{R}_{\geq 0}^{d\times d}$ and a sequence $(a_k)_{k\in\mathbb{N}^*} \in \mathbb{R}_{\geq 0}^{\mathbb{N}^*}$ such that $a_k > 0$ for any $k \in [\![1, d]\!]$.

By definition, we have,

$$h_a(A) = \text{Tr}\left[\sum_{k=1}^{+\infty} a_k A^k\right] \tag{13}$$

$$= \sum_{k=1}^{+\infty} a_k \text{Tr}\left[A^k\right] \tag{14}$$

$$\tag{15}$$

1. By Lemma A.2, $\text{Tr}\left[A^k\right] \geq 0$ and so $h_a(A) \geq 0$. This proves the second property.
2. Then, $h_a(A) = 0$ if and only if $\text{Tr}\left[A^k\right] = 0$ for all k for which $a_k > 0$. Since $a_k > 0$ for any $k \in [\![1, d]\!]$ we conclude that if $h_a(A) = 0$, then A does not contain cycles of length $k \leq d$, so A is acyclic by Definition A.1. Reciprocally, if A is acyclic, it does not contain cycles of any length, so $h_a(A) = 0$.
3. Finally, if we write r_a the radius of convergence of f_a, then h_a converge absolutely over the set of matrices with $h_\rho(A) < r_a$ so it is differentiable with gradient given by: $\nabla h_a(A) = \sum_{k=1}^{+\infty} a_k k (A^\top)^{k-1}$.

This concludes the proof.

□

A.2. Proof of Theorem 2

We recall Theorem 3.4.

Theorem 3.4 (PST instability)**.** *For $d \geq 2$, any PST constraint h is both E-unstable and V-unstable. More precisely,* ***- E-unstable.*** $\exists A \in \mathbb{R}_{\geq 0}^{d\times d}, h(sA) = \Omega_{s\to\infty}(s^d)$
- V-unstable. $\exists A \in \mathbb{R}_{\geq 0}^{d\times d}, h(\varepsilon A) = O_{\varepsilon\to 0^+}(\varepsilon^d)$

Also, any PST constraint for which f_a has a finite radius of convergence is ***D-unstable*** *(e.g., $h_{\log}, h_{\text{inv}}$).*

Proof. Take a PST constraint h_a for some $(a_k)_k \in \mathbb{R}_{\geq 0}^{d\times d}$ with $a_k > 0$ for $k \in [\![1, d]\!]$.

We will show the E-unstable and V-unstable results using a particular adjacency matrix C.

Define C as the adjacency matrix of the cycle $1 \to 2 \to ... \to d \to 1$ with edges weights of 1. That is:

$$C = \begin{bmatrix} 0 & 1 & 0 & \dots & \dots & 0 \\ & 0 & 1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ & \vdots & & \ddots & \ddots & 0 \\ & & & & \ddots & 1 \\ 1 & 0 & & \dots & & 0 \end{bmatrix} \tag{16}$$

We have $C^d = I_d$ and $\text{Tr}\left[C^k\right] = \begin{cases} d & \text{if } k = 0 \mod d \\ 0 & \text{if } k \neq 0 \mod d \end{cases}$.

We obtain for any $w \in \mathbb{R}_{\geq 0}$,

$$h_a(wC) = d \sum_{\ell=1}^{+\infty} a_{\ell d} w^{\ell d}. \tag{17}$$

- In particular, we have for any $s \geq 0$, $h_a(wC) = d a_d s^d = \Omega_{s \to +\infty} s^d$ (since $a_d > 0$). This proves the E-instability.
- Define $u = \min(1, r_a/2)$ where r_a is the radius of convergence of f_a.
  Then, for any $\varepsilon \in [0, u^2]$,

$$\begin{aligned}
h_a(\varepsilon C) &= d \sum_{\ell=1}^{+\infty} a_{\ell d} \varepsilon^{\ell d} && (18)\\
&= \varepsilon^d d \left(\sum_{\ell=1}^{+\infty} a_{\ell d} \varepsilon^{(\ell-1)d} \right) && (19)\\
&\leq \varepsilon^d d \left(\sum_{\ell=1}^{+\infty} a_{\ell d} u^{2(\ell-1)d} \right) && (20)\\
&\leq \varepsilon^d d \left(\sum_{\ell=1}^{+\infty} a_{\ell d} u^{\ell d} + a_d \right) && (21)\\
&\leq \varepsilon^d d \left(f_a(uC) + a_d \right). && (22)\\
&= O_{\varepsilon \to 0^+}(\varepsilon^d) && (23)
\end{aligned}$$

  Where we obtain Equation (21) by noting that $2(\ell - 1) \geq \ell$ and $u \leq 1$. Finally, since $u < r_a$, $f_a(uC)$ is finite. Hence the result.

The D-instability result follows from the definition of the radius of convergence. □

A.3. Proof of Theorem 3

We recall Theorem 3.6.

Theorem 3.6 (Cvetković et al. (1980); Lee et al. (2019))**.** *The spectral radius is an acyclicity constraint.*

$$h_\rho(A) = 0 \Leftrightarrow A \text{ is a DAG.}$$

We refer to it as the spectral acyclicity constraint. It is differentiable almost everywhere, with gradient

$$\nabla h_\rho(A) = v_d u_d^\top / v_d^\top u_d,$$

where u_d, v_d are respectively the right and left eigenvectors associated with $\lambda_d(A)$ (Magnus, 1985).

The two properties stated in Theorem 3.6 are standard results.

Proof.

- We provide proof for the statement $h_\rho(A) = 0 \Leftrightarrow A$ is a DAG for the sake of completeness.
  - $\Rightarrow$ If $h_\rho(A) = 0$ then all eigenvalues $\lambda_j(A)$ are zeros. But since $\text{Tr}\left[A^k\right] = \sum_{j=1}^d \lambda_j(A)^k$, we have $\text{Tr}\left[A^k\right] = 0$ for any $k \geq 1$ and by Lemma A.2, A is acyclic.
  - $\Leftarrow$ Assume A is acyclic, then $A^d = 0$ by Lemma A.2. But then all eigenvalues are 0 (as for eigenvalue $\lambda_j(A)$ and associated eigenvector $v_j(A)$, we have $A^d v_j(A) = \lambda_j(A)^d v_j(A) = 0$.

Hence, h_ρ is a valid acyclicity constraint.

- Magnus (1985) shows that h_ρ is differentiable at every A that has mutually distinct eigenvalues, with the formula provided in Theorem 3.7. The set of matrices with all distinct eigenvalues is dense in the set of matrices (Horn & Johnson, 2012)[Theorem 2.4.7.1], which proves the result

□

A.4. Proof of Theorem 4

We recall Theorem 3.7.

Theorem 3.7. *h_ρ is stable.*

Proof. We prove each stability criterion.

- **E-stable**: For any $s > 0$ and matrix A, $h_\rho(sA) = |s|h_\rho(A) = O_{s\to+\infty}(s)$.
- **V-stable**:For any $\varepsilon > 0$ and matrix A such that $h_\rho(A) > 0$, $h_\rho(\varepsilon A) = |\varepsilon|h_\rho(A) = \Omega_{\varepsilon\to 0^+}(\varepsilon)$.
- **D-stable**: Every matrix has eigenvalues ($\mathbb{C}$ is algebraically closed), so h_ρ is well defined everywhere. In addition, Theorem 3.6 proved that h_ρ was differentiable almost everywhere.

Hence, h_ρ is a stable constraint. □

A.5. Proof for Stage 1

In this section, we guarantee that if the optimization problem solved in stage 1 is solved exactly, without relaxation and with infinite data, then stage 1 does not remove any true causal parent.

The optimization problem solved in stage 1 is given in Equation (7) as

$$\hat{\theta}_1 = \underset{\substack{\theta \\ \forall j, A_{\theta,jj}=0}}{\arg\max}\ S_{\alpha_1,\beta_1}(\theta) = \underset{\substack{\theta \\ \forall j, A_{\theta,jj}=0}}{\arg\max}\ \frac{1}{n}\sum_{i=1}^{n} \log p(x^i;\theta,t^i) - \alpha_1\|A_\theta\|_1 - \beta_2\|\theta\|_2^2.$$

With infinite data $x^i|t^i \sim p^*(x^i;t^i)$, the optimization problem writes,

$$\tilde{\theta}_1 = \underset{\substack{\theta \\ \forall j, A_{\theta,jj}=0}}{\arg\max} \sum_{k=0}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} [\log p(x;\theta,k)] - \alpha_1\|A_\theta\|_1 - \beta_2\|\theta\|_2^2. \tag{24}$$

where π_k is the proportion of data coming from intervention k.

Furthermore, in its non-relaxed form, Equation (24) above writes

$$\tilde{\theta}_1 = \underset{\substack{\theta \\ \forall j, A_{\theta,jj}=0}}{\arg\max} \sum_{k=0}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} [\log p(x;\theta,k)] - \lambda|A_\theta|, \tag{25}$$

where the L1 and L2 regularization are reverted back into the number of edges $|A_\theta|$ regularization (for some $\lambda > 0$).

Since we are interested in the graph induced by $\tilde{\theta}_1$, that we write $\tilde{G}_1 = A_{\tilde{\theta}_1}$, we can rewrite Equation (25) as

$$\tilde{G}_1 = \underset{\substack{G \\ \text{without self-loops}}}{\arg\max}\ \sup_{G=A_\theta} \sum_{k=0}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} [\log p(x;\theta,k)] - \lambda|G|, \tag{26}$$

Finally, since there are no constraint over G other than no self-loops, Equation (26) can be solved as d independent optimization problems, each one determining the parents of j in the graph $\tilde{G}_1$,

$$\mathrm{pa}_j^{\tilde{G}_1} = \underset{S \subset [\![1,d]\!] \setminus \{j\}}{\arg\max} \sup_{\substack{\theta \\ S = \mathrm{pa}_j^{A_\theta}}} \sum_{k=0}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} \left[\log p_j(x_j | x_{-j}; \theta, k)\right] - \lambda |S|, \tag{27}$$

Furthermore, whenever $j \in I_k$, our model class has $p_j(x_j|x_{-j}, \theta, k) = p_j(x_j|\theta_{(j,k)}, k)$ — we know we have perfect interventions and the interventions are known. So the $\theta_{(j,k)}$ is not related to the coordinates of θ that define A_θ. That is to say, Equation (27) is equivalent to

$$\mathrm{pa}_j^{\tilde{G}_1} = \underset{S \subset [\![1,d]\!] \setminus \{j\}}{\arg\max} \sup_{\substack{\theta \\ S = \mathrm{pa}_j^{A_\theta}}} \sum_{\substack{k=0 \\ j \notin I_k}}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} \left[\log p_j(x_j | x_{-j}; \theta, k)\right] - \lambda |S|, \tag{28}$$

We recall Theorem 4.2.

Theorem 4.2. *Under regularity assumptions detailed in Appendix A.5, the candidate parents* $\mathrm{pa}_j^{\tilde{G}_1}$ *of* j *selected by stage 1 are precisely the Markov boundary of* j *in the true graph* G^**, That is,* $\mathrm{pa}_j^{\tilde{G}_1} = \mathrm{pa}_j^{G^*} \cup \mathrm{ch}_j^{G^*} \cup \mathrm{pa}_{\mathrm{ch}_j^{G^*}}^{G^*}$.

The assumptions are similar to the ones detailed in Brouillard et al. (2020) to guarantee that differentiable causal discovery can identify causal graphs.

The assumptions are:

- $\pi_0 > 0$ – we observe some observational data,
- $\exists \theta,$ s.t. $\forall k, p^*(\cdot\,; k) = p(\cdot\,; \theta, k)$ – the model class can express the true model p^*,
- The observational distribution $p^*(x; 0)$ is *faithful* to the graph G^* (that is any edge in G^* indeed result in a nonzero cause-and-effect relation in the distribution $p^*(x; 0)$. See Neapolitan et al. (2004) for more details.
- The true distributions $p^*(x; k)$ and any distribution of the model class $p(x; \theta, k)$ have strictly positive density $p^*(x; k) > 0$, $p(x; \theta, k) > 0$. This avoids technical difficulty when forming conditional distributions (e.g., $p^*(x_j|x_T; k)$).
- The expectations $\mathbb{E}_{p^*(x;k)}[\log p^*(x; k)]$ are well defined (they are finite). This enables us to consider the likelihood expectations in the first place.
- The regularization strength λ is strictly positive and small enough (see the proof for how small).

Proof. Fix $j \in [\![1, j]\!]$.

For clarity of notations, we rewrite Equation (28) as

$$\mathrm{pa}_j^{\tilde{G}_1} = \underset{S \subset [\![1,d]\!] \setminus \{j\}}{\arg\max} \sup_{\theta} \sum_{\substack{k=0 \\ j \notin I_k}}^{K} \pi_k \mathop{\mathbb{E}}_{p^*(x;k)} \left[\log p_j(x_j | x_S; \theta, k)\right] - \lambda |S|, \tag{29}$$

where the condition $S = \mathrm{pa}_j^{A_\theta}$ is fully captured by the notation $p_j(x_j|x_S; \theta, k)$.

Then, define

$$\psi(T) = \sup_{\theta} \sum_{\substack{k \\ j \notin I_k}} \mathbb{E}_{p^*(x;k)} \left[\pi_k \log p_j(x_j | x_S; \theta, k)\right] - \lambda |S|. \tag{30}$$

Further, define $B = \mathrm{bo}_j^{G^*}$ to be the Markov boundary of node j in the true causal graph G^*.

We will show that $\psi(B) > \psi(T)$ for any other $T \subset [\![1, d]\!] \setminus \{j\}$.

We compute,

$$\psi(B) - \psi(T) = \sup_\theta \sum_{\substack{k \\ j \notin I_k}} \pi_k \mathbb{E}_{p^*(x;k)} \left[\log p_j(x_j|x_B;\theta,k)\right] - \sup_\theta \sum_{\substack{k \\ j \notin I_k}} \pi_k \mathbb{E}_{p^*(x;k)} \left[\log p_j(x_j;x_T;\theta,k)\right] \tag{31}$$

$$- \lambda|B| + \lambda|T|$$

$$= -\inf_\theta \sum_{\substack{k \\ j \notin I_k}} \pi_k \mathbb{E}_{p^*(x_{-j};k)} \left[D_{KL}\left(p_j^*(x_j|x_{-j};k) \parallel p_j(x_j|x_B;\theta,k)\right)\right] \tag{32}$$

$$+ \inf_\theta \sum_{\substack{k \\ j \notin I_k}} \pi_k \mathbb{E}_{p^*(x_{-j};k)} \left[D_{KL}\left(p_j^*(x_j|x_{-j};k) \parallel p_j(x_j;x_T;\theta,k)\right)\right]$$

$$- \lambda(|B| - |T|)$$

$$= \inf_\theta \sum_{\substack{k \\ j \notin I_k}} \pi_k \mathbb{E}_{p^*(x_{-j};k)} \left[D_{KL}\left(p_j^*(x_j|x_{-j};k) \parallel p_j(x_j;x_T;\theta,k)\right)\right] \tag{33}$$

$$+ \lambda(|T| - |B|).$$

The line 32 comes from $\mathbb{E}_{p^*(x;k)}\left[\log p_j(x_j|x_B;\theta,k)\right] = -\mathbb{E}_{p^*(x_{-j};k)}\left[D_{KL}\left(p_j^*(x_j|x_{-j};k) \parallel p_j(x_j|x_B;\theta,k)\right)\right] + \mathbb{E}_{p^*(x;k)}\left[\log p_j^*(x_j|x_{-j};k)\right]$ where we added and substracted the $\log p^{(k)}$ term (the $\mathbb{E}_{p^*(x;k)}$ is decomposed into $\mathbb{E}_{p^*(x_{-j};k)}\mathbb{E}_{p^*(x_j;k)}$, where the second expectation is in the KL divergence). We use the assumption of strictly positive density here to define the conditional $p_j^*(x_j|x_{-j};k)$ without technical difficulties.

The line 33 comes from the assumption of sufficient model class capacity and the definition of the Markov boundary. Indeed, we first have $p_j^*(x_j|x_{-j};k) = p_j^*(x_j|x_B;k)$ by definition of the Markov boundary B, and since the model class is expressive enough, there exists θ such that $D_{KL}\left(p_j^*(x_j|x_{-j};k) \parallel p_j(x_j|x_B;\theta,k)\right) = 0$.

We further have:

$$\psi(B) - \psi(T) \geq \pi_0 \inf_\theta \mathbb{E}_{p^*(x_{-j};0)}\left[D_{KL}\left(p_j^*(x_j|x_B;0) \parallel p_j(x_j|x_T;\theta,0)\right)\right] + \lambda(|T| - |B|) \tag{34}$$

$$= \pi_0 \mathbb{E}_{p^*(x_{-j};0)}\left[D_{KL}\left(p_j^*(x_j|x_B;0) \parallel p_j^*(x_j|x_T;0)\right)\right] + \pi_0 \inf_\theta \mathbb{E}_{p^*(x;0)}\left[\log \frac{p_j^*(x_j|x_T;0)}{p_j(x_j|x_T;\theta,0)}\right] \tag{35}$$

$$+ \lambda(|T| - |B|)$$

$$\geq \underbrace{\pi_0 \mathbb{E}_{p^*(x_{-j};0)}\left[D_{KL}\left(p_j^*(x_j|x_B;0) \parallel p_j^*(x_j|x_T;0)\right)\right]}_{\eta(T)} + \lambda(|T| - |B|). \tag{36}$$

where line 36 follows from $\mathbb{E}_{p^*(x;0)}\left[\log \frac{p_j^*(x_j|x_T;0)}{p_j(x_j|x_T;\theta,0)}\right] = \mathbb{E}_{p^*(x_T)}\left[D_{KL}\left(p_j^*(x_j|x_T) \| p_j(x_j|x_T;\theta,0)\right)\right] \geq 0$.

Let's finally define $u = \min\left(\left\{\frac{\eta(T)}{|B|-|T|} \mid T \subset [\![1,d]\!]\backslash\{j\} \text{ and } \eta(T) > 0 \text{ and } |B| > |T|\right\} \cup \{1\}\right)$ and fix any $\lambda \in]0, u[$.

Let's assume now that $\psi(T) \geq \psi(B)$ for some $T \subset [\![1,d]\!]\backslash\{j\}$, and show that we obtain contradictions.

First, we would have $\lambda(|B| - |T|) \geq \eta(T)$. In particular we deduce that $|B| \geq |T|$ (since $\eta(T) \geq 0$).

Now, two possibilities:

1. If $\eta(T) > 0$, then $|B| > |T|$ and by definition of λ, $\lambda > \lambda$ which is absurd.
2. If $\eta(T) = 0$, then $\pi_0 \mathbb{E}_{p^*(x_{-j};0)}\left[D_{KL}\left(p_j^*(x_j|x_B;0) \parallel p_j^*(x_j|x_T;0)\right)\right] = 0$. This implies that $D_{KL}\left(p_j^*(x_j|x_B;0) \parallel p_j^*(x_j|x_T;0)\right) = 0$ for all (x_{-j}); since $p^*(x_{-j};0)$ has positive density and $\pi_0 > 0$. Hence, the conditional $p_j^*(x_j|x_B;0)$ and $p_j^*(x_j|x_T;0)$ are identical. Since B was the Markov boundary of x_j, that makes T also a Markov blanket of x_j. But then, by minimality of the Markov boundary in a faithful graph, we have $B \subset T$. Remember that we had deduced $|B| \geq |T|$. So $B = T$.

This ends the proof, where $\lambda \in]0, u[$. □

A.6. Proof for Stage 2

Since stage 1 does not remove any true causal parents, theorem 1 of Brouillard et al. (2020) remains valid.

A.7. Lemma: Asymptotic Bound on number of edges returned in Stage 1

We denote the Markov boundary of j in G^* by $\text{bo}_j^{G^*}$, and recall that $\text{bo}_j^{G^*} = \text{pa}_j^{G^*} \cup \text{ch}_j^{G^*} \cup \text{pa}_{\text{ch}_j^{G^*}}^{G^*} \setminus \{j\}$.

The following lemma upper-bounds the theoretical number of edges returned by stage 1 when each node has at most k parents.

Lemma A.3. *Assume G^* is sparse such that each node has at most k parents. Then, the total size of all the Markov boundaries is upper-bounded by $dk(k+2) = O(dk^2)$.*

Proof. First, note that if each node has at most k parents, then $|E| \leq dk$. Finally,

$$\sum_{j \in V} |\text{bo}_j^{G^*}| = \sum_{j \in V} |\text{bo}_j^{G^*}| \tag{37}$$

$$\leq \sum_{j \in V} |\text{pa}_j^{G^*}| + |\text{ch}_j^{G^*}| + |\text{pa}_{\text{ch}_j^{G^*}}^{G^*}| \tag{38}$$

$$\leq |E| + |E| + \sum_{j \in V} \sum_{k \in \text{ch}_j^{G^*}} |\text{pa}_k^{G^*}| \tag{39}$$

$$\leq 2kd + \sum_{k \in V} \sum_{j \in \text{pa}_j^{G^*}} |\text{pa}_k^{G^*}| \tag{40}$$

$$\leq 2kd + dk^2 \tag{41}$$

□

B. Methods

B.1. Model Details

In SDCD, the conditional distributions, $p_j(x_j|x_{-j};\theta,k)$, are modeled as Gaussian distributions where the mean and variance are learned by a neural network that takes in all of the other x_{-j} as input. The initial layer of the network applies d independent linear transformations followed by a sigmoid nonlinearity to the input and outputs d hidden states of size 10. Each of the d hidden states corresponds to the features then used to predict each variable. Each hidden state is fed into two linear layers: one to predict the mean parameter of the conditional and one to predict the variance parameter of the conditional. For the variance, a softplus operation is applied to the output of the linear layer to constrain the variance to be strictly positive.

B.2. Algorithm Details

B.2.1. Spectral Acyclicity Constraint Estimation

As described in Theorem 3.6, the gradient of the spectral acyclicity constraint can be computed as $h_\rho(A) = v_d u_d^\top / v_d^\top u_d$, where u_d, v_d are the right and left eigenvectors of A respectively. Using the power iteration method, which involves a fixed number of matrix-vector multiplications, u_d, v_d can be estimated in $O(d^2)$. Specifically, the updates are as follows:

$$u_d^{(i+1)} := \frac{A^\top u_d^{(i)}}{\|u_d\|_2},\ v_d^{(i+1)} := \frac{A v_d^{(i)}}{\|v_d\|_2}$$

where u_d, v_d are initialized as $u_d^{(1)}, v_d^{(1)} := [\frac{1}{\sqrt{d}}, \ldots, \frac{1}{\sqrt{d}}]$ at the very first epoch of SDCD. In our implementation, we use 15 iterations to estimate the spectral acyclicity constraint value.

Importantly, we re-use the estimates of u_d and v_d from one epoch to another, as we don't expect A (and its eigenvectors) to change drastically.

Hence, at each epoch, we initialize u_d, v_d using their last epoch's value and perform 15 power iterations.

B.2.2. SDCD Algorithm

The SDCD algorithm follows a two-stage procedure. In the first stage, the coefficient of the spectral acyclicity constraint, γ, is fixed at zero. We use an Adam optimizer with a learning rate, η_1, specific to stage 1 to perform minibatch gradient-based optimization. The coefficients corresponding to the L1 and L2 penalties, α_1 and β_1, respectively, are fixed throughout training. The stage 1 training loss is written as:

$$\begin{aligned}\mathcal{L}_1(X,\theta,\alpha_1,\beta_1) &= S_{\alpha_1,\beta_1}(\theta) \\ &= \frac{1}{n}\sum_{i=1}^{n} \log p(x^i;\theta,t^i) - \alpha_1\|A_\theta\|_1 - \beta_1\|\theta\|_2^2.\end{aligned}$$

To prevent the model from learning implicit self-loops, the weights corresponding to the predicted variable are masked out for every hidden state output by the initial neural network layer. Thus, the prediction of each variable is prevented from being a function of the same variable.

In interventional regimes, the log-likelihood terms corresponding to the prediction of intervened variables are zeroed out. The intervened variables do not have to be modeled as we assume perfect interventions.

Stage 1 is run for a fixed number of epochs. By default, stage 1 also has an early stopping mechanism that uses the reconstruction loss of a held-out validation set of data (sampled uniformly at random from the training set) as the early stopping metric. If the validation reconstruction loss does not achieve a new minimum after a given number of epochs, the stage 1 training loop is exited.

At the end of stage 1, the learned input layer weights are used to compute a set of removed edges, $\hat{R}$, for stage 2. Let $W \in \mathbb{R}^{d\times d\times 10}$ represent the input layer weights. Then, each value of the implicitly defined weighted adjacency matrix is computed as the L2 vector norm for the corresponding set of weights (i.e., $A_{\theta,i,j} := \|W_{i,j,:}\|_2$). This weighted adjacency matrix is discretized with a fixed threshold, τ_1, such that each edge, (i,j), is removed if it falls below the threshold (i.e., $A_{\theta,i,j} < \tau_1$).

In stage 2, the spectral acyclicity constraint is introduced. Like stage 1, we use an Adam optimizer with learning rate, η_2, and perform minibatch gradient-based optimization. Once again, the L1 and L2 coefficients, α_2, β_2, are fixed throughout training. Rather than a fixed γ, SDCD takes an increment value, $\gamma^+ \in \mathbb{R}^+$, determining the rate at which γ increases every epoch. The training loss for stage 2 is as follows:

$$\begin{aligned}\mathcal{L}_2(X, \theta, \hat{R}, \alpha_2, \beta_2, \gamma) &= S_{\alpha_2,\beta_2}(\theta) - \gamma h_\rho(A_\theta) \\ &= \frac{1}{n}\sum_{i=1}^{n} \log p(x^i; \theta, t^i) - \alpha_2 \|A_\theta\|_1 - \beta_2 \|\theta\|_2^2 - \gamma h_\rho(A_\theta).\end{aligned}$$

The same masking strategy as in stage 1 is used to prevent self-loops in A_θ. However, the input layer weights corresponding to edges $(i, j) \in \hat{R}$ are also masked.

Like before, the reconstruction loss terms corresponding to intervened variables are removed from the loss.

To reduce the sensitivity of stage 2 to the choice of γ^+ and to prevent the acyclicity constraint term from dominating the loss, the linear increment schedule is frozen when A_θ achieves a DAG at the final threshold, τ_2. In practice, the DAG check is performed every 20 epochs. If the adjacency matrix returns to being cyclic throughout training, the γ increment schedule restarts to increase from where it left off.

The early stopping metric is computed similarly to stage 1, but in stage 2, the early stopping can only kick in when γ has been frozen. If the γ schedule is resumed due to A_θ reintroducing a cycle, the early stopping is reset.

Lastly, once stage 2 is complete, A_θ is computed and thresholded according to a fixed threshold, τ_2. All values exceeding the threshold (i.e., $A_\theta, i, j \geq \tau_2$) are considered edges in the final graph prediction.

The thresholded adjacency matrix may contain cycles if stage 2 runs to completion without hitting early stopping. To ensure a DAG, we follow a greedy edge selection procedure detailed in Algorithm 2.

Pseudocode for a simplified SDCD algorithm (excludes γ freezing and early stopping) is provided in Algorithm 1.

B.2.3. Time and Space Complexity

The time complexity of each iteration of SDCD is $O(d^2)$. The forward pass in stage 1 can be computed in $O(d^2)$ time. On the other hand, each of the d prediction problems can be computed independently. This allows for parallelizing the d problems, each taking $O(d)$ time. Stage 2 also takes $O(d^2)$ time as the spectral acyclicity constraint and the forward pass both take $O(d^2)$ time to compute. Thus, the time complexity of each iteration in both stages is $O(d^2)$.

If the sparsity pattern of the underlying causal graph is known beforehand such that each variable has at most k parents, we can further tighten the time complexity of SDCD. By Appendix A.7, we know the size of the set of remaining edges after stage 1 is $O(dk^2)$. Using sparse matrix multiplication, the spectral acyclicity constraint can be done in $O(dk^2)$, which is effectively linear in d if $k \ll d$. However, this improvement only becomes significant when $d > 10,000$ (from experiments not reported in this paper).

The space complexity of the algorithm is $O(d^2)$, as the number of parameters in the input layer scale quadratically in the number of features.

Algorithm 1 SDCD

Require: $\alpha_1, \alpha_2 \in \mathbb{R}^+, \beta_1, \beta_2 \in \mathbb{R}^+, \gamma^+ \in \mathbb{R}^+,$
$\tau_1, \tau_2 \in \mathbb{R}^+, \eta_1, \eta_2 \in \mathbb{R}^+, E_1, E_2 \in \mathbb{Z}^+$
$A_\theta^{(0)} \leftarrow \vec{0}^{G \times G}$
$\theta_{-A_\theta}^{(0)} \leftarrow$ RandomGaussianInit()
$e \leftarrow 0$
while $e < E_1$ **do**
$\theta^{(e+1)} := \mathrm{AdamUpdate}(\theta^{(e)}, \nabla \mathcal{L}_1(X, \theta^{(e)}, \alpha_1, \beta_1), \eta_1)$
$e \leftarrow e + 1$
end while
$\hat{R} := \mathrm{Threshold}(A_\theta^{(E_1)}, \tau_2)$
$A_\theta^{(E_1)} \leftarrow \vec{0}^{G \times G}$
$\theta_{-A_\theta}^{(E_1)} \leftarrow$ RandomGaussianInit()
$\gamma \leftarrow 0$
while $e < E_1 + E_2$ **do**
$\theta^{(e+1)} := \mathrm{AdamUpdate}(\theta^{(e)}, \nabla \mathcal{L}_1(X, \theta^{(e)}, \hat{R}, \alpha_2, \beta_2, \gamma), \eta_2)$
$\gamma \leftarrow \gamma + \gamma^+$
$e \leftarrow e + 1$
end while
output $\mathrm{DAGTrim}(A_\theta^{(E_2)}, \tau_2)$

Algorithm 2 DAGTrim

Require: $A_\theta \in \mathbb{R}^{D \times D}, \tau \in \mathbb{R}^+$
$E \leftarrow \emptyset$ {Initialize the set of final edges.}
$C \leftarrow [(i,j) \in [\![1, d]\!]^2 \mid (A_{\theta,i,j} > \tau]$ {Candidate edges above threshold τ.}
Sort C by decreasing $A_{\theta,i,j}$.
for each $(i,j) \in C$ **do**
if the graph with edges $E \cup \{(i,j)\}$ is still acyclic **then**
$E \leftarrow E \cup \{(i,j)\}$ {We add the edge if it does not create a cycle.}
end if
end for

C. Empirical Studies Details

C.1. Simulation Details

To judge the performance of SDCD against existing methods over both interventional and observational data, we generated simulated data according to the following procedure:

- Draw a random undirected graph from the Erdős-Rényi distribution.
- Convert the undirected graph into a DAG G^* by setting the direction of each edge $i \to j$ if $\pi(i) < \pi(j)$, where π is a random permutation of the nodes.
- Form d possible sets of interventions that target one variable at a time: $I_j = \{j\}$ and $I_0 = \emptyset$.
- Draw a set of random fully connected neural networks $\mathrm{MLP}^{(j)} : \mathbb{R}^{|\,\mathrm{pa}_j^{G^*}\,|} \to \mathbb{R}^{100} \to 1$, each one with one 100-dimensional hidden layer. Each neural network parametrizes the mean of the observational conditional distributions:

$$p_j^*(x_j \mid x_{\mathrm{pa}_j^{G^*}}; 0) \sim \mathcal{N}\left(\mu = \mathrm{MLP}^{(j)}(x_{\mathrm{pa}_j^{G^*}}), \sigma = 0.5\right).$$

- For intervention distribution $k \geq 1$, perform a hard intervention on variable k and set

$$p_j^*(x_k; k) \sim \mathcal{N}(0, 0.1).$$

- Draw the data according to the model, with 10,000 observational samples and 500 extra interventional samples per target variable.
- Standardize the data.

We consider several values of d to simulate different scenarios.

C.2. Choice of Hyperparameters

We fixed the hyperparameters as follows: $\alpha_1 := 1\mathrm{e}{-2}, \beta_1 := 2\mathrm{e}{-4}, \eta_1 := 2\mathrm{e}{-3}, \tau_1 := 0.2, \alpha_2 := 5\mathrm{e}{-4}, \beta_2 := 5\mathrm{e}{-3}, \eta_2 := 1\mathrm{e}{-3}, \gamma^+ := 0.005, \tau_2 := 0.1$. We found that these selections worked well empirically across multiple simulated datasets and were used in all experiments without simulation-specific fine-tuning.

Each stage was run for 2000 epochs with a batch size of 256, and the validation loss was computed over a held-out fraction of the training dataset (20% of the data) every 20 epochs for early stopping. In stage 2, the DAG check of the implicit adjacency matrix was performed every 20 epochs before the validation loss computation.

C.3. Baseline Methods

Here, we provide details on the baseline methods and cite which implementations were used for the experiments. For DCDI and DCDFG, we used the implementations from `https://github.com/Genentech/dcdfg`, using the default parameters for optimization. For DCDFG, we used 10 modules in our benchmarks, as reported in the paper experiments. For GIES, we used the Python implementation from `https://github.com/juangamella/gies`, using the default parameters. For DAGMA, we used the original implementation from `https://github.com/kevinsbello/dagma` with the default parameters. For NOTEARS, we used the implementation from `https://github.com/xunzheng/notears`, and for NOBEARS, we used the implementation from `https://github.com/howchihlee/BNGPU`. For NOTEARS and NOBEARS, we found the default thresholds for determining the final adjacency matrix performed poorly or did not return a DAG, so for each of these baselines, we followed the same procedure described in Lopez et al. (2022): we find the threshold that returns the largest possible DAG via binary search. sortnregress (Reisach et al., 2021) is a trivial baseline meant to ensure that the causal graph cannot be easily inferred from the variance pattern across the variables. For this baseline, we used the implementation in `https://github.com/Scriddie/Varsortability`.

C.4. Robustness Checks

Below, we discuss three categories of issues that commonly arise when evaluating causal discovery methods and address each issue with a diagnostic metric.

Sparsity. Particularly when the true causal graph is sparse, SHD may favor sparser predictions since, in the extreme case, the empty graph achieves an SHD equal to the number of true edges. To show the relative performance of the benchmarked methods with respect to this trivial solution, we indicate the number of true edges for each simulated setting in Figure 3 and Figure 4. We find that most methods outperform this baseline. Additionally, we report the F1 score and the recall of the predictions (see Figures 7 and 10), two metrics that suffer when a method predicts many false negatives. We find that SDCD still outperforms other methods with these metrics.

Identifiability. In settings with incomplete or no interventional data, the true causal graph may be unidentifiable, meaning multiple $\mathcal{I}$-Markov equivalent graphs can maximize the score (Brouillard et al., 2020). Therefore, graphs in the same Markov equivalence class as the true causal graph may have positive SHD values despite being optimal with respect to the available data. As proposed in Peters et al. (2014), we also compute an adapted version of the SHD to compare the Markov equivalence class of the methods' results against the true Markov equivalence class instead of the graphs themselves. This metric, called SHD-CPDAG, is computed as the SHD between the completed partially directed acyclic graph (CPDAG) of the predicted graph and the CPDAG of the true graph. Unlike the regular SHD metric, this metric is zero if two graphs are in the same equivalence class. We report it alongside SHD for our experiments in Figure 7 to better represent the results in scenarios with an unidentifiable causal graph. We find very similar results.

Simulation issues. As discussed in Reisach et al. (2021), certain simulation processes used for causal discovery benchmarking exhibit an issue where the order of the variables, when sorted by sample variance, reflects the true causal ordering of the graph. As a result, methods that exploit this phenomenon to accurately infer the causal graph may be misrepresented. To ensure that our simulation process does not suffer from this issue and that the methods are being properly evaluated, we take two complementary steps recommended in Reisach et al. (2021): (i) we standardize the data before being input into any of the evaluated methods so that no artificial sample variance information can be exploited, and (2) we include the trivial baseline, sortnregress (Reisach et al., 2021), which is designed to exploit sample variance artifacts from a flawed simulation, and should be outperformed by an effective, scale-invariant algorithm. We find that sortnregress performs poorly, which confirms that our normalization scheme removes simulation artifacts, and we find that SDCD and its competing methods beat sortnregress by a wide margin.

D. Supplementary Figures and Tables

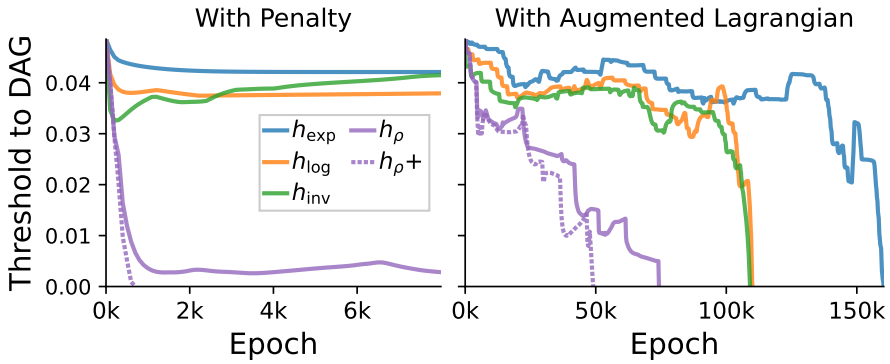

Figure 5: The effect of constraints on the learned graph throughout training. The training with penalty $h_\rho+$ (dashed purple, exactly h_ρ with a hard mask on the diagonal as to prevent self-loops, as implemented in SDCD) converges the fastest toward a DAG. (left) training with h as a regularization penalty. (right) training with h as an augmented Lagrangian constraint. *Threshold to DAG* is the smallest η at which all edges with weight $> \eta$ form a DAG.

s	d	δ	Method	SDCD	DCDI-G	DCDI-DSF
1	10	22.2%	L	**0.7**$\pm$**1.2**	1.3$\pm$1.9	0.9$\pm$1.3
			NL-Add	**0.6**$\pm$**0.7**	5.2$\pm$7.5	4.2$\pm$5.6
			NL-NN	**0.7**$\pm$**0.7**	2.3$\pm$3.6	7.0$\pm$10.7
	20	10.5%	L	**1.4**$\pm$**3.4**	5.4$\pm$4.5	3.6$\pm$2.7
			NL-Add	**4.1**$\pm$**3.0**	21.8$\pm$30.1	4.3$\pm$1.9
			NL-NN	**3.0**$\pm$**2.5**	13.9$\pm$20.3	8.3$\pm$4.1
4	10	88.9%	L	5.2$\pm$3.5	**3.3**$\pm$**2.1**	3.7$\pm$2.3
			NL-Add	4.8$\pm$2.1	**4.3**$\pm$**2.4**	5.5$\pm$2.4
			NL-NN	7.3$\pm$3.0	2.4$\pm$1.6	**1.6**$\pm$**1.6**
	20	42.1%	L	18.8$\pm$10.5	23.7$\pm$5.6	**16.6**$\pm$**6.4**
			NL-Add	**18.0**$\pm$**7.3**	35.2$\pm$13.2	26.7$\pm$16.9
			NL-NN	14.9$\pm$1.9	16.8$\pm$8.7	**11.8**$\pm$**2.1**

Table 3: Means and standard deviations of SHD scores over simulations from Brouillard et al. (2020). The "Method" column refers to the model used to simulate the causal relationships. "L" refers to linear model, "NL-Add" refers to nonlinear, additive model, and "NL-NN" refers to nonlinear, non-additive (neural network) model. We refer to Brouillard et al. (2020) for the simulation details. The results are reported alongside the values presented in the original paper. s refers to the expected number of edges per node, d denotes the number of nodes, and the edge density, δ, is computed as the fraction of $\frac{d(d-1)}{2}$, the maximum number of edges for a DAG. The lowest average SHD values are set in bold.

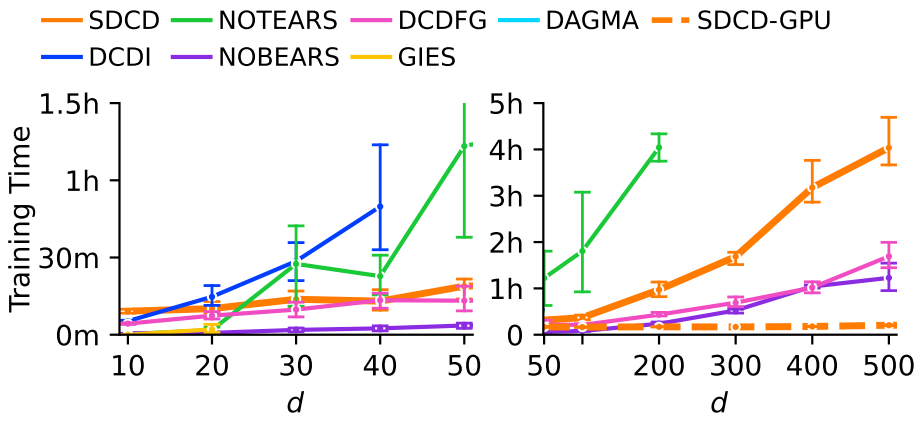

Figure 6: Training runtimes across simulations from Figure 3. SDCD on GPU (dashed) scales to 500 variables in under 20 minutes. Error bars indicate std on 5 random datasets for $d < 50$ and 3 random datasets for $d \geq 50$.

d	SDCD	SDCD-GPU	DCDI	DCDFG	GIES	DAGMA	NOTEARS	NOBEARS	SCORE	sortnregress	AVICI	NOCURL
10	14.7 ±5.5	NT	24.3 ±3.9	24.6 ±6.0	27.8 ±3.9	25.3 ±6.2	35.3 ±1.9	33.5 ±3.0	14.4 ±4.0	28.6 ±4.7	23.52	33.23
20	35.7 ±6.2	NT	31.7 ±6.5	108.0 ±14.3	123.4 ±12.3	62.0 ±12.0	75.4 ±4.0	74.2 ±3.0	118.6 ±8.5	81.6 ±6.3	60.93	82.37
30	53.8 ±11.9	NT	55.5 ±10.7	258.8 ±32.6	NA	89.4 ±10.9	113.1 ±4.8	113.7 ±3.3	275.9 ±21.0	134.6 ±8.4	97.13	134.60
40	64.0 ±13.7	NT	102.4 ±21.6	426.6 ±73.7	NA	115.3 ±13.4	147.9 ±6.0	151.1 ±3.4	454.3 ±52.8	172.9 ±12.2	135.83	179.93
50	69.9 ±12.3	68.3 ±13.3	NA	660.8 ±126.1	NA	NA	183.4 ±7.4	192.0 ±3.5	619.4 ±59.7	216.6 ±12.4	170.83	240.93
100	92.7 ±9.1	89.7 ±11.0	NA	1807.3 ±788.2	NA	NA	327.3 ±7.5	389.0 ±3.6	NA	421.3 ±12.0	366.50	513.07
200	225.3 ±13.7	228.0 ±18.3	NA	5657.3 ±2982.6	NA	NA	619.0 ±4.2	770.0 ±7.8	NA	824.0 ±19.0	NT	NT
300	350.0 ±12.5	360.0 ±*nan*	NA	7284.7 ±5072.3	NA	NA	NA	1149.0 ±14.0	NA	1190.7 ±26.3	NT	NT
400	466.3 ±62.4	471.7 ±68.0	NA	3779.7 ±507.3	NA	NA	NA	1534.7 ±3.1	NA	1585.0 ±59.3	NT	NT
500	621.7 ±10.7	621.0 ±10.5	NA	7252.7 ±3284.6	NA	NA	NA	1915.7 ±18.8	NA	1974.3 ±34.6	NT	NT

Table 4: Detailed results of SHD means and standard deviations from Figure 3. SDCD-GPU was only run for $d \geq 50$. All other NA values correspond to failed runs (possibly from timeout after 6h, e.g., GIES, or from training error, e.g., DCDI). NT corresponds to the method not having been tested on that particular example.

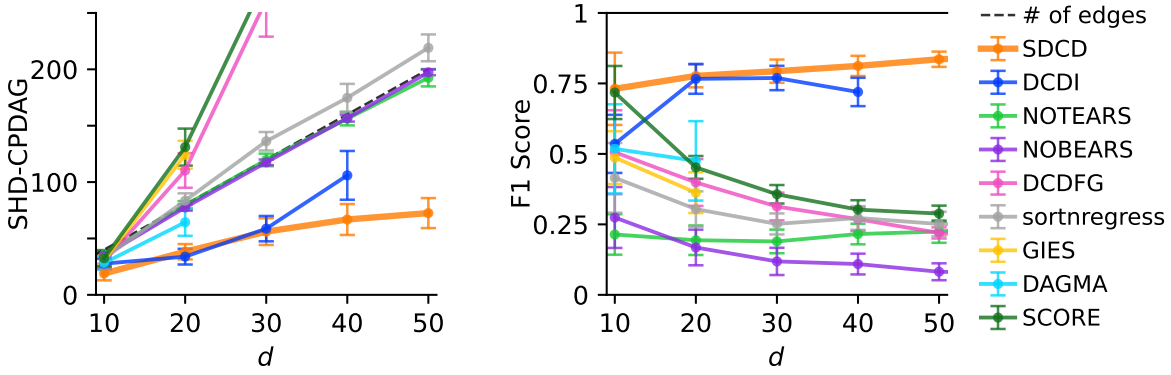

Figure 7: F1 and SHD-CPDAG metrics across simulations from Figure 3, observational data with increasing numbers of variables d. Missing data points imply the method failed to run. Error bars indicate std on 30 random datasets.

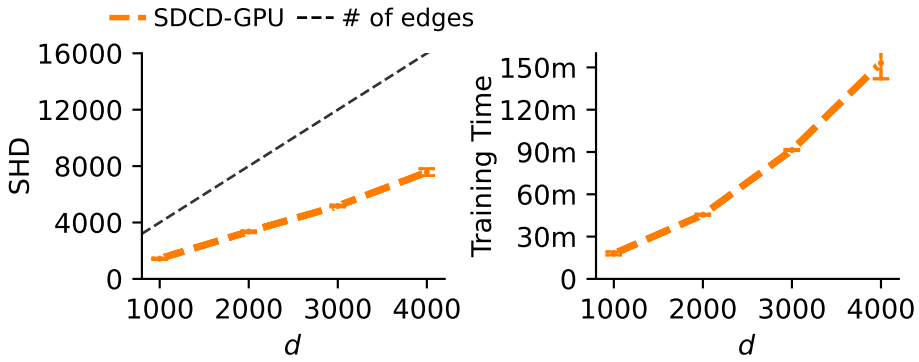

Figure 8: SDCD on GPU (dashed) scales to 4000 variables under 3 hours while maintaining competitive SHD. Error bars indicate std on 3 random datasets for $d = 1000, 2000$ and 2 random datasets for $d = 3000, 4000$.

d	SDCD-GPU
1000	1438.7 $\pm$59.2
2000	3356.7 $\pm$70.0
3000	5172.5 $\pm$89.8
4000	7567.0 $\pm$343.7

Table 5: Detailed results of SHD means and standard deviations from Figure 8.

In addition to SHD, we computed precision and recall metrics over the predicted edges with respect to the true edges for both observational and interventional scenarios. The precision is the fraction of true edges among all the predicted edges. The recall is the fraction of true edges that have been correctly predicted.

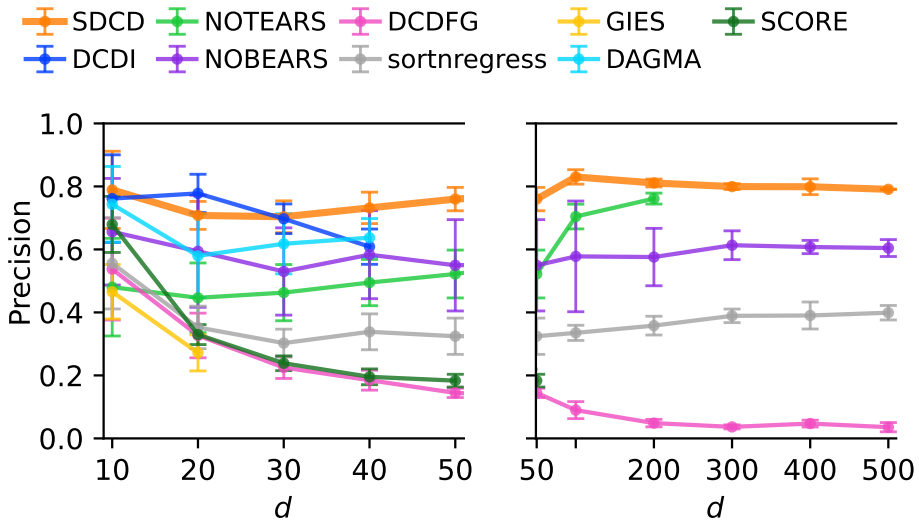

Figure 9: Precision across simulations from Figure 3, observational data with increasing numbers of variables d. The SDCD(-CPU) and SDCD-GPU lines overlap, indicating consistent results. Missing data points imply the method failed to run. Error bars indicate std on 30 random datasets for $d < 50$ and 5 random datasets for $d \geq 50$.

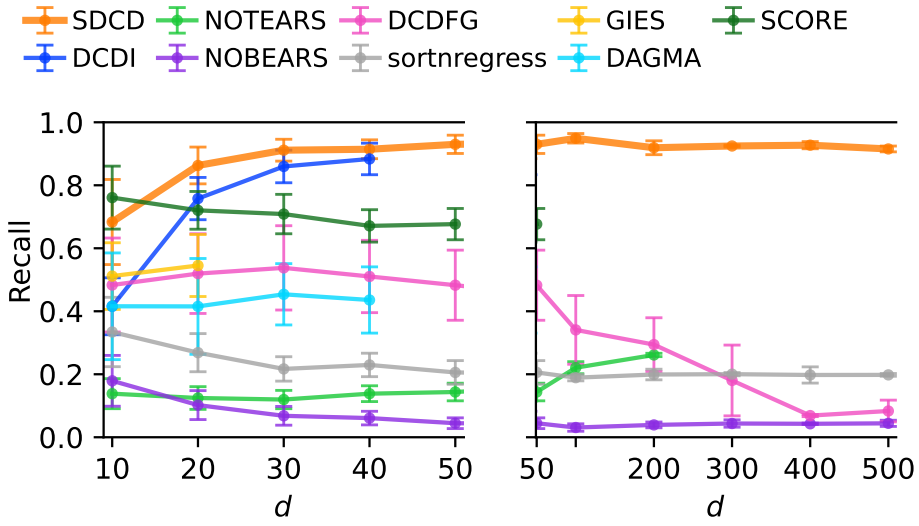

Figure 10: Recall across simulations from Figure 3, observational data with increasing numbers of variables d. The SDCD(-CPU) and SDCD-GPU lines overlap, indicating consistent results. Missing data points imply the method failed to run. Error bars indicate std on 30 random datasets for $d < 50$ and 5 random datasets for $d \geq 50$.

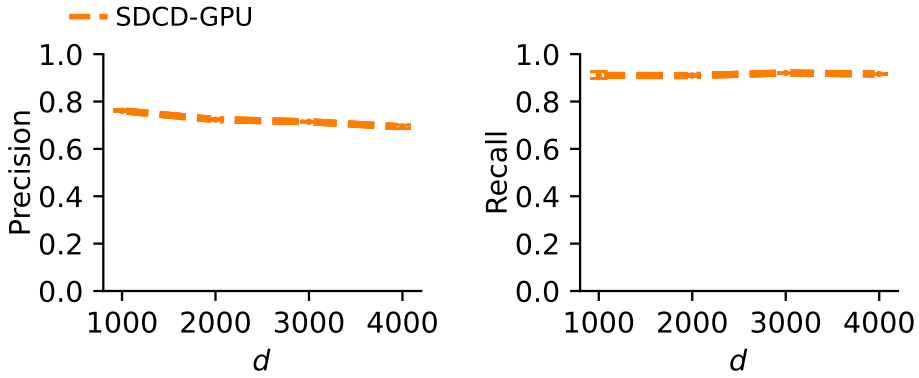

Figure 11: Precision and recall across simulations from Figure 8, observational data with increasing numbers of variables d. Error bars indicate std on 3 random datasets for $d = 1000, 2000$ and 2 random datasets for $d = 3000, 4000$.

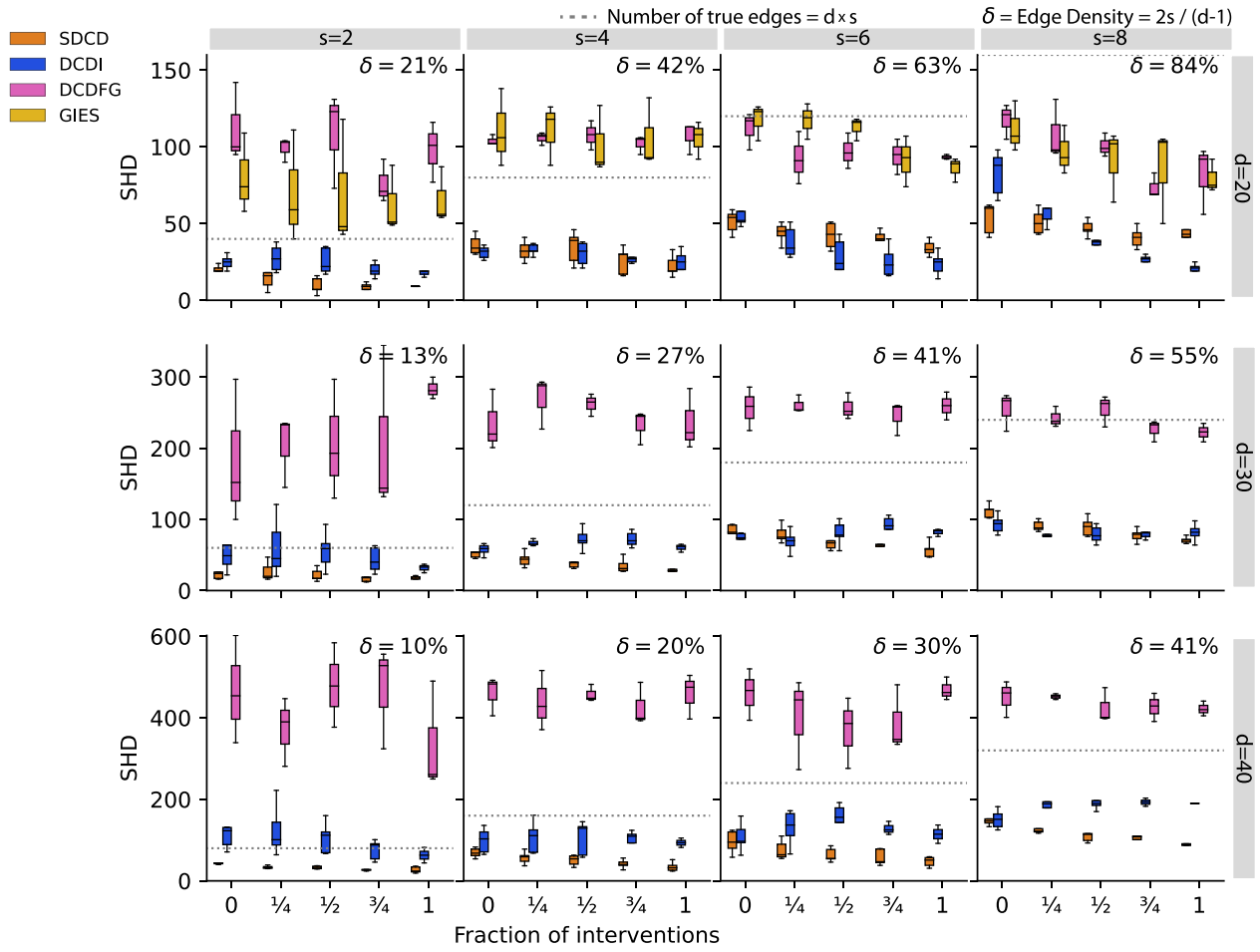

Figure 12: SHD across simulations with an increasing proportion of variables intervened on, varying the total number of variables d (columns) and average edges per variable s (rows). Extended version of Figure 4 with DCDFG and GIES and $s = 8$. Boxplots over 5 random datasets.

s	d	δ	Fraction of Variables Intervened on	SDCD	DCDI	DCDFG	GIES
2	20	21%	0.00	**18.0** ±6.5	24.8 ±4.6	112.3 ±25.8	80.3 ±26.1
			0.25	**13.4** ±5.7	27.4 ±8.6	99.0 ±7.8	70.0 ±36.8
			0.50	**9.4** ±5.4	25.4 ±8.5	109.0 ±31.4	69.7 ±41.9
			0.75	**9.0** ±2.1	19.8 ±4.8	76.0 ±14.2	62.7 ±22.0
			1.00	**9.0** ±2.8	18.8 ±3.3	98.0 ±19.7	65.7 ±18.5
	30	13%	0.00	**24.6** ±9.6	56.0 ±32.9	183.0 ±102.1	NA
			0.25	**26.8** ±13.1	60.4 ±40.9	204.3 ±51.4	NA
			0.50	**21.8** ±9.0	56.2 ±26.6	206.7 ±84.3	NA
			0.75	**16.2** ±3.4	43.2 ±17.8	207.0 ±119.7	NA
			1.00	**18.2** ±2.2	31.7 ±6.1	283.7 ±15.2	NA
	40	10%	0.00	**44.2** ±4.1	109.2 ±27.6	465.0 ±131.8	NA
			0.25	**33.4** ±3.6	123.8 ±62.1	372.7 ±84.3	NA
			0.50	**33.0** ±3.5	105.6 ±38.8	479.7 ±103.5	NA
			0.75	**27.6** ±3.4	76.0 ±24.4	469.3 ±126.6	NA
			1.00	**27.0** ±7.5	63.0 ±26.9	333.7 ±135.5	NA
4	20	42%	0.00	36.0 ±6.4	**31.2** ±4.1	104.0 ±3.5	110.7 ±25.3
			0.25	**32.2** ±6.6	33.0 ±3.6	105.7 ±4.2	110.7 ±20.0
			0.50	34.6 ±10.6	**30.4** ±7.6	107.7 ±9.5	101.3 ±22.3
			0.75	**25.8** ±8.8	29.6 ±8.2	102.0 ±6.1	105.7 ±22.8
			1.00	**22.4** ±7.1	25.8 ±6.4	107.0 ±10.4	105.3 ±12.2
	30	27%	0.00	**54.0** ±9.8	57.6 ±7.9	234.7 ±42.9	NA
			0.25	**43.8** ±10.3	67.0 ±4.0	269.3 ±36.7	NA
			0.50	**39.2** ±8.6	72.4 ±15.5	262.0 ±15.7	NA
			0.75	**35.0** ±9.9	72.2 ±10.7	232.7 ±24.0	NA
			1.00	**29.0** ±6.5	60.3 ±5.7	236.0 ±42.8	NA
	40	20%	0.00	**69.0** ±11.7	99.0 ±30.7	460.0 ±47.8	NA
			0.25	**56.8** ±15.4	107.0 ±39.2	438.3 ±73.1	NA
			0.50	**50.4** ±13.0	105.6 ±41.6	457.7 ±21.2	NA
			0.75	**41.4** ±10.7	97.8 ±33.9	426.3 ±52.6	NA
			1.00	**34.4** ±11.3	93.5 ±16.3	458.7 ±55.3	NA
6	20	63%	0.00	**51.2** ±7.5	56.6 ±10.4	112.0 ±12.3	117.7 ±11.9
			0.25	44.0 ±6.5	**37.8** ±10.2	92.3 ±17.0	117.3 ±11.6
			0.50	42.2 ±8.6	**29.0** ±10.8	97.0 ±11.5	112.7 ±7.6
			0.75	38.8 ±8.3	**25.2** ±10.0	94.0 ±11.5	91.3 ±16.6
			1.00	34.0 ±5.1	**23.8** ±7.7	93.3 ±1.5	86.0 ±7.9
	30	41%	0.00	85.4 ±6.5	**75.8** ±4.4	256.7 ±30.6	NA
			0.25	79.8 ±12.5	**69.2** ±15.4	260.7 ±12.4	NA
			0.50	**69.4** ±14.8	80.6 ±17.2	257.3 ±18.6	NA
			0.75	**67.0** ±12.2	86.2 ±23.3	245.3 ±23.7	NA
			1.00	**57.4** ±11.3	82.0 ±5.3	259.7 ±19.5	NA
	40	30%	0.00	**95.4** ±27.7	107.8 ±36.3	460.3 ±63.3	NA
			0.25	**75.6** ±23.6	130.2 ±43.3	401.0 ±112.8	NA
			0.50	**63.6** ±17.0	146.0 ±51.7	370.0 ±87.1	NA
			0.75	**57.4** ±19.6	128.3 ±16.3	387.7 ±81.1	NA
			1.00	**47.2** ±12.2	114.5 ±31.8	469.0 ±28.2	NA
8	20	84%	0.00	**53.6** ±10.2	82.8 ±14.5	117.7 ±11.4	111.7 ±16.5
			0.25	**51.0** ±8.1	58.2 ±12.6	108.3 ±19.7	96.7 ±15.8
			0.50	47.0 ±5.3	**41.4** ±13.8	100.7 ±7.6	91.0 ±23.5
			0.75	40.8 ±6.7	**26.0** ±3.8	73.7 ±8.1	86.0 ±31.2
			1.00	43.0 ±9.0	**19.8** ±4.9	81.7 ±22.4	79.7 ±10.8
	30	55%	0.00	111.8 ±9.8	**93.4** ±13.3	255.0 ±27.1	NA
			0.25	90.8 ±7.5	**75.6** ±8.2	242.7 ±14.6	NA
			0.50	89.6 ±13.2	**78.8** ±12.2	255.0 ±22.1	NA
			0.75	**77.6** ±9.3	81.2 ±9.4	226.3 ±15.1	NA
			1.00	**71.0** ±4.6	81.6 ±12.5	222.3 ±13.0	NA
	40	41%	0.00	**150.4** ±16.9	151.0 ±23.1	450.0 ±44.5	NA
			0.25	**127.0** ±12.4	188.4 ±27.4	452.0 ±7.0	NA
			0.50	**113.4** ±20.2	200.0 ±33.9	424.3 ±43.0	NA
			0.75	**104.4** ±21.3	193.0 ±14.1	426.7 ±34.6	NA
			1.00	**92.0** ±17.6	190.0 $\pm nan$	422.0 ±18.1	NA

Table 6: Detailed results of SHD means and standard deviations from Figure 12. GIES failed to run on $d \geq 30$.

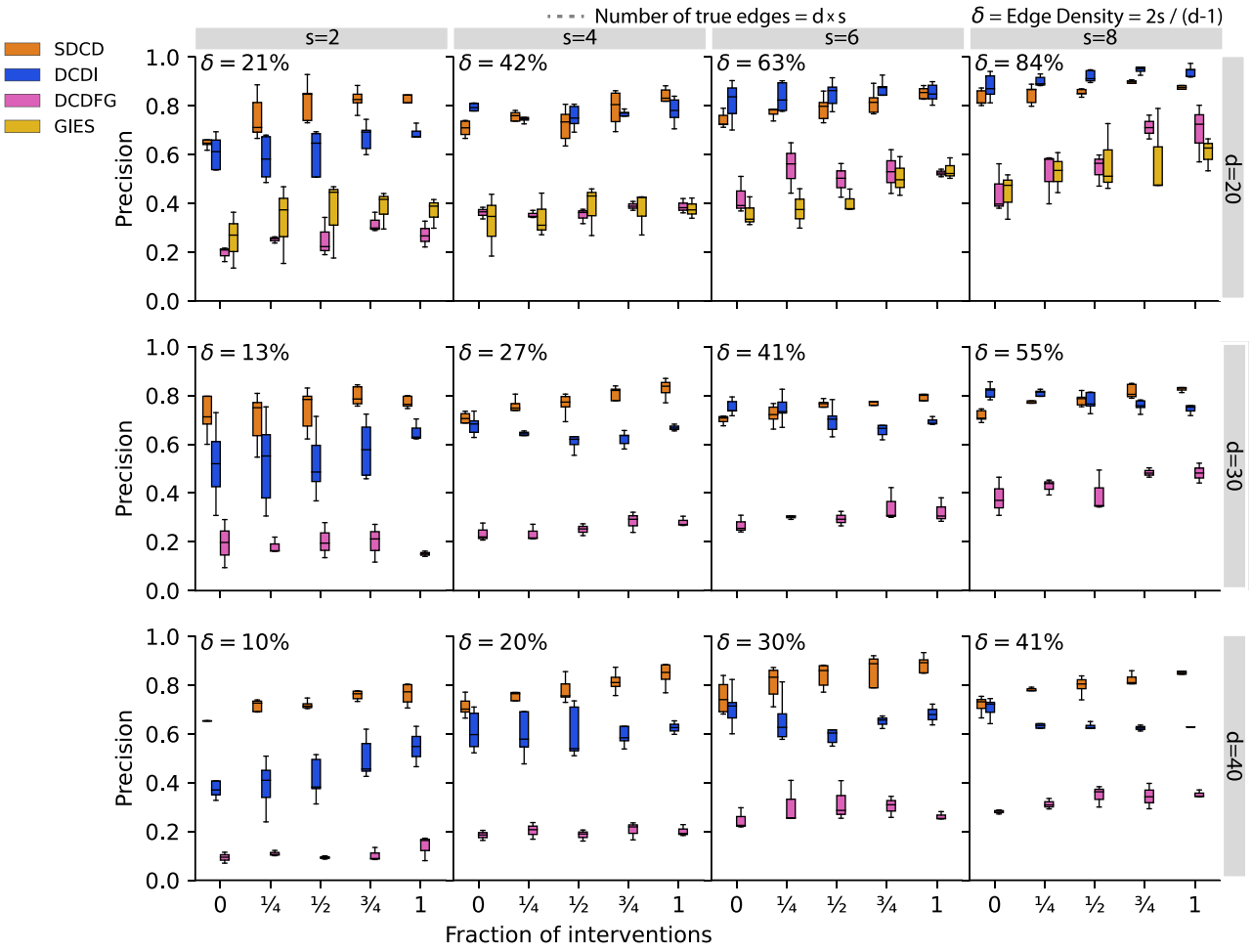

Figure 13: Precision across simulations from Figure 12 increasing proportion of variables intervened on, varying the total number of variables d (columns) and average edges per variable s (rows). Boxplots over 5 random datasets.

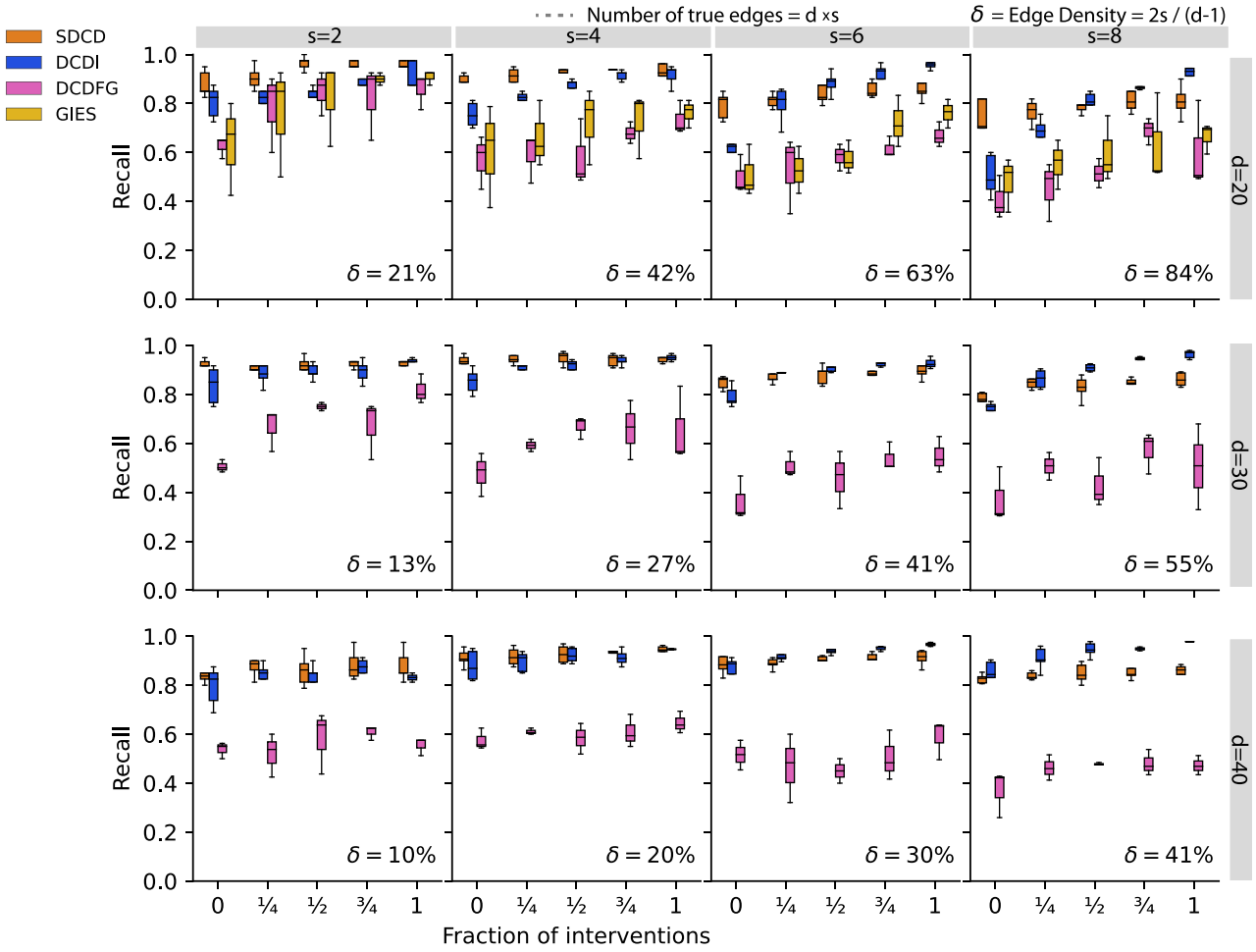

Figure 14: Recall across simulations from Figure 12 increasing proportion of variables intervened on, varying the total number of variables d (columns) and average edges per variable s (rows). Boxplots over 5 random datasets.

Name	d=10	d=20	d=30	d=40
SDCD	**14.7**	40.3	**54.3**	69.0
SDCD-warm	**14.7**	40.7	55.0	**68.7**
SDCD-warm-nomask	19.3	69.7	156.0	272.7
SDCD-no-s1	19.3	68.3	155.3	272.3
SDCD-no-s1-2	16.3	56.7	95.0	135.0
DCDI	24.0	**35.7**	56.7	87.0

Table 7: Ablation study for SDCD stage 1. We observe that the described version of SDCD performs the best out of all variations. SDCD-warm performs competitively but generally provides little benefit. SDCD-warm-nomask performs much worse than SDCD, demonstrating that enforcing the mask during stage 2 is important. We report mean SHD over three random seeds of observational data (no interventions) with a fixed number of edges per variable, $s = 4$, for a range of numbers of variables, d. **SDCD-warm** refers to starting stage 2 of SDCD, where the input layer is ported over from stage 1 instead of re-learned. **SDCD-warm-nomask** performs the same warmstart as SDCD-warm but does not enforce the mask in stage 2. **SDCD-no-s1** only performs stage 2. **SDCD-no-s1-2** only does stage 2, but sets (α_2, β_2) to the default values from stage 1 (α_1, β_1). We report these values alongside DCDI. The lowest SHD values are bolded for each value of d.

Name	$d = 10$	$d = 20$	$d = 30$	$d = 40$	$d = 50$
SDCD	13.33	**33.47**	**54.07**	**70.80**	**76.60**
SDCD-exp	11.60	44.33	69.07	85.07	89.93
SDCD-log	**11.20**	52.00	87.47	117.87	116.00

Table 8: Ablation study for SDCD stage 2 (choice of the constraint). We observe that SDCD performs the best out of the three variations. Additionally, the variations using the PST constraints do not crash for any of the runs, even for those with $d = 50$. We attribute this improved stability (as compared to DCDI and DAGMA) to stage 1 since there are fewer non-zero parameters contributing to the value of the constraint. We report mean SHD over five random seeds of observational data (no interventions) with a fixed number of edges per variable, $s = 4$, for a range of numbers of variables, d. **SDCD-exp** is SDCD except using the h_{exp} constraint in place of the h_ρ, and **SDCD-log** uses the h_{log} constraint.