LATENT CAUSAL RECOVERY FROM LONG-LAG DEPENDENCIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Inferring causal structure from observational time series is challenging in the best of conditions, and even more so when relevant variables are unobserved. Marginalizing latent variables typically induces complex long-lag dependencies among observed variables, where a lag denotes the number of time steps separating cause and effect, making the inverse problem of recovering latent structure ill-posed. To recover latent structure under such confounding, we introduce Causal Hidden Recovery of N-lagged Observed Structures (CHRONOS), a constraint-optimization method built on answer set programming. Given a lagged observed graph, CHRONOS searches for causal graphs with minimal latent complexity that are consistent with the observations. The method explicitly allows for multiple optimal solutions, yielding equivalence classes of latent structures rather than a single reconstruction. Through extensive simulations on synthetic causal graphs with varying graph density and numbers of latent variables, we evaluate when latent structure recovery is completed within the time budget, how many latent variables are required, and the limits of identifiability in the resulting reconstructions. We additionally apply CHRONOS to real traffic-sensor data (METR-LA). Our results characterize the regimes in which latent causal recovery from apparent long-range dependencies is tractable, and the identifiability limits that latent confounding imposes regardless of method.

1 INTRODUCTION

Causal structure learning from observational time series is a central problem in many scientific domains, including neuroscience, economics, and climate science (Runge, 2018). In practice, many relevant variables are unobserved, either because they are unmeasured or fundamentally inaccessible. Marginalizing such latent variables induces complex lagged dependencies among observed variables (we formalize this marginalization process in Section 3), complicating both causal interpretation and downstream analysis. Causal inference with latent variables and causal discovery in time series have each been studied in isolation (Spirtes et al., 2000; Runge, 2018). However, the problem of recovering latent causal structure from lagged causal representations induced by marginalizing latent variables remains underexplored (Plis et al., 2015; Hyttinen et al., 2016).

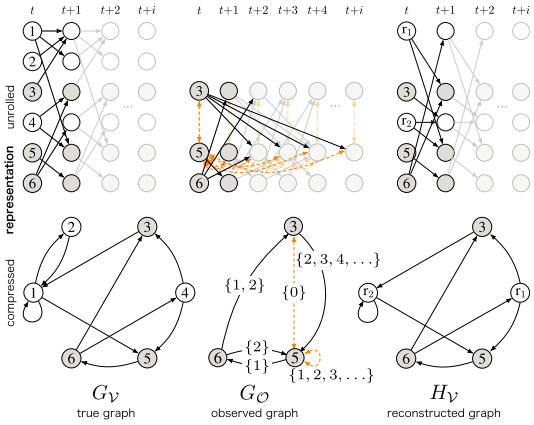


Figure 1: Underdetermination in latent causal reconstruction shown by distinct latent causal graphs ($G_{\mathcal{V}}$ and $H_{\mathcal{V}}$) with identical observed dependencies ($G_{\mathcal{O}}$) (in unrolled (top) and compressed (bottom) representations).

The first two columns in Figure 1 illustrate how marginalizing latent variables can transform a simple unit-lag causal system into a lagged observed graph with heterogeneous temporal dependencies. The ground-truth graph $G_{\mathcal{V}}$ contains six variables connected exclusively by directed unit-lag edges; that is, each edge represents a temporal dependency of the form $X_t \to Y_{t+1}$ (perhaps $X = Y$). After marginalizing three latent variables, the induced observed graph $G_{\mathcal{O}}$ exhibits directed lagged edges

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

annotated with singleton, finite, and infinite lag sets, as well as bidirected edges (shown in orange) arising from latent confounding. Each directed lagged edge $X \to Y$ in $G_{\mathcal{O}}$ represents all lags $\ell \in \mathbb{N}$ such that there exists a directed path of length ℓ from X to Y in $G_{\mathcal{V}}$ whose intermediate nodes are latent. The top row shows time-unrolled representations, while the bottom row shows them as compressed graphs.[1]

The third column demonstrates the difficulty of identifying latent structure from $G_{\mathcal{O}}$. The latent graph $H_{\mathcal{V}}$ also implies $G_{\mathcal{O}}$ after marginalization, but note that it contains only five variables. Multiple latent graphs, potentially with different latent counts, can be consistent with identical observed long-lag causal relations.

In this work, we develop and test both (i) a formalization of the forward marginalization process; and also (ii) the Causal Hidden Recovery of N-lagged Observed Structures (CHRONOS) algorithm, an inverse causal inference procedure that searches for latent causal graphs consistent with a given observed lagged graph. Rather than aiming for a unique reconstruction, the latter method formulates latent recovery as an optimization problem that minimizes latent complexity subject to constraints on observational consistency. This formulation allows for multiple optimal solutions, yielding equivalence classes of latent causal structures. By enumerating all optimal solutions, we show when multiple latent graphs are equally consistent with the observed data.

2 Related Work

There are significant lines of work on causal inference from static or i.i.d. data in the presence of latent variables, and also from time series data without latent variables. In the former line, Spirtes et al. (1995) characterize equivalence classes of causal DAGs consistent with observed conditional independences in the presence of latent variables and selection bias, using partially oriented inducing path graphs. The Fast Causal Inference (FCI) algorithm (Spirtes et al., 2000) uses these formal results to recover latent causal structures from i.i.d. data. The FCI algorithm has been generalized and extended in many ways in subsequent years. Examples include an augmented version of the FCI algorithm with complete orientation rules (Zhang, 2008), RFCI (Colombo et al., 2012), and GFCI (Ogarrio et al., 2016).

A different approach to discovering latent structure from static data has focused on finding low-rank structure in the association matrix over the observed variables. The core idea—latent variables imply identifiable constraints on association structures—dates back to factor analysis and tetrad constraints (Spearman, 1904). In the causal context, these rank constraint approaches have been used to find latent causal structure within the broad class of measurement models such as tetrad-based methods (Silva et al., 2003; 2012), FTFC (Kummerfeld et al., 2014), FOFC (Kummerfeld & Ramsey, 2016). More generally, trek-separation results characterize graphical conditions under which covariance submatrices have low rank in Gaussian graphical models (Sullivant et al., 2010).

The space of causal learning methods from time series data without latent confounders is far too large to describe here. Notable methods include Granger causality (Granger, 1969), PCMCI (Runge et al., 2019), TiMINo (Peters et al., 2013), DYNOTEARS (Pamfil et al., 2020). While these methods have exhibited impressive performance, the assumption of no (causally relevant) latent variables is a significant one in many real-world use cases.

The intersection of these lines—causal learning from time series with latents—has been less-studied. Blondel et al. (2017) extend the do-calculus to dynamic causal networks and study how hidden confounding across time affects identification and transportability, though they assume a known causal graph. Entner & Hoyer (2010) develop a time series version of the FCI algorithm, though it does not fully exploit complex structure within the induced lags. More recently, LPCMCI extends PCMCI to handle autocorrelated time series with latent confounders while maintaining high recall (Gerhardus & Runge, 2020). Complementary to structure learning, Jahn et al. (2025) show that a finite time window suffices, under suitable bounds on the maximum lag, for identifiability of causal effects in time series graphs with latent confounders.

[1] Assuming first-order Markov dynamics and absence of isochronal (zero-lag) directed edges, each time-unrolled process admits an equivalent compressed representation in which nodes correspond to variables and each edge is annotated with the set of achievable temporal lags induced by directed paths.

Plis et al. (2015) provide a causal discovery method for situations in which the measurement rate is slower than the system dynamics, thereby generating "mediating" latents between measured timesteps (see also (Hyttinen et al., 2016; Abavisani et al., 2023; 2025)). Gong et al. (2015) study the same problem under parametric vector autoregressive assumptions, in contrast to our nonparametric formulation. Runge (2018) surveys how time lags, latent drivers, and sampling effects complicate reconstruction from multivariate time series.

Beyond these constraint-based and graphical approaches, deep learning methods have been used for causal discovery and latent representation learning from time series, often via invariance-based objectives and sequence models (Li et al., 2025; Shen et al., 2025). These methods use a parametric encoder or predictor, though, so cannot provide guarantees of finding all possible latent structures.

3 METHODS

We write the full system, observed and latent, in a unit-lag normal form. Each direct effect takes one time step, and longer observed lags come from paths through latent nodes. This loses no generality. A lag-k effect is a chain of k unit steps through $k-1$ latent states, just as a VAR(p) is a VAR(1) on a larger state (Corollary 1). So the observed process can have any order, even though the full system is first order. Many unit-lag graphs can reproduce the same lag-set graph, so we seek those with the fewest latent nodes, and these form an equivalence class.

Let $\mathcal{V} = \mathcal{O} \cup \mathcal{L}$ be the full set of variables, with observed variables $\mathcal{O}$ and latent variables $\mathcal{L}$. We assume the standard Markov and faithfulness conditions for dynamic causal graphical models, so the conditional independences are exactly those the graph implies. We represent the full system as a directed, perhaps cyclic, graph $G_{\mathcal{V}}$, where $X \to Y$ means X_t directly influences Y_{t+1}. Self-loops represent autoregressive dependence. We exclude contemporaneous directed edges, so the model fits data sampled at or faster than the causal time scale; slower sampling marginalizes time steps rather than variables, a separate problem (Plis et al., 2015; Hyttinen et al., 2016). A path is a sequence of directed edges, and its length, or lag, is the number of edges: a path from X_t to $Y_{t+\ell}$ has lag ℓ.

Latent variables produce longer observed lags through multi-step paths, so we record lags as sets. Let $\mathcal{P}(\mathbb{N})$ be the power set of the natural numbers, including the empty set. A compressed graph over a node set $\mathcal{S}$ has two lag-set functions, $d_{\mathcal{S}}, b_{\mathcal{S}} : \mathcal{S} \times \mathcal{S} \to \mathcal{P}(\mathbb{N})$, which give the lags of directed and bidirected influences. For example, $d_{\mathcal{S}}(X, Y) = \{1, 2, 4\}$ means X_t influences Y_{t+1}, Y_{t+2}, and Y_{t+4}. Without latent variables, each $d_{\mathcal{S}}(X, Y)$ is $\emptyset$ or $\{1\}$, and no bidirected edges exist.

3.1 FORWARD INFERENCE

The forward marginalization operator $\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{L}) = G_{\mathcal{O}}$ populates these lag-set functions for the observed variables. For $X, Y \in \mathcal{O}$, the directed lag set $d_{\mathcal{O}}(X, Y)$ is the set of all $\ell \in \mathbb{N}$ such that there exists a path $p : X \to \cdots \to Y$ in $G_{\mathcal{V}}$ of length ℓ where all intermediate nodes are in $\mathcal{L}$.

The bidirected lag-difference set $b_{\mathcal{O}}(X, Y)$ is the set of all $\ell \in \mathbb{N}$ such that there exists a *(latent) fork*: $Z \in \mathcal{L}$ and paths $p_X : Z \to \cdots \to X$ and $p_Y : Z \to \cdots \to Y$, with all intermediate nodes in $\mathcal{L}$, such that $\ell = \mathrm{len}(p_Y) - \mathrm{len}(p_X) > 0$. Thus, nonzero bidirected lags are stored in the direction from the observed endpoint reached earlier by the latent fork to the endpoint reached later. For $\ell = 0$, the endpoints are reached at the same lag, so we record the value only once by requiring $X < Y$.

In general, these lag sets may be infinite: cycles involving latent variables can induce arbitrarily long paths and therefore arbitrarily large lags. The marginalization operator is well-defined with respect to $\mathcal{L}$: the resulting lag sets do not depend on the order in which latent variables are removed.

Theorem 1. *For all* $\mathcal{I} \subset \mathcal{V}, x \in \mathcal{V} \setminus \mathcal{I}$: $\mathrm{Mar}(\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I}), \{x\}) = \mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I} \cup \{x\})$

Proof sketch. First, note that marginalizing an additional variable monotonically increases each lag set; marginalization cannot remove a previously-existing long-lag dependence. The proof of this theorem thus consists in showing that the marginal lag additions to the graph resulting from marginalizing $\mathcal{I}$ (on the left-hand side of the equality) result in exactly the lag sets that are produced by marginalizing $\mathcal{I} \cup \{x\}$ simultaneously. A full proof is given in Appendix A.6. □

We implement the marginalization operator using an ASP encoding (Listing 1) that computes the directed paths and forks that populate the lag sets. The predicates `observed_directed(X,Y,L)` and `observed_bidirected(X,Y,L)` correspond to the lag-set functions $d_{\mathcal{O}}(X, Y)$ and

$b_{\mathcal{O}}(X, Y)$, respectively. Since the program cannot actually construct infinite sets, we use a finite horizon M (specified by `maxlag`), where $d_{\mathcal{O}}^{M}(X, Y) = d_{\mathcal{O}}(X, Y) \cap \{1, 2, \ldots, M\}$, and $b_{\mathcal{O}}^{M}(X, Y) = b_{\mathcal{O}}(X, Y) \cap \{0, 1, \ldots, M\}$. In the simulations, we use `maxlag` $= 17$; in general, `maxlag` should be chosen to be at least as large as the largest delay of interest in the observed data. When the latent subgraph is acyclic, any `maxlag` exceeding the longest latent-only directed path length suffices to recover the complete finite lag structure.

Listing 1: ASP specification of forward marginalization

```
#const maxlag = 17.
lag(0..maxlag).
path(X,Y,1)  :- edge(X,Y,1).
path(X,Y,L2) :- path(X,Z,L1), edge(Z,Y,1), hidden(Z), L2 = L1 + 1, lag(L2
    ).
fork(X,Y,L) :- path(Z,Y,L1), path(Z,X,L2), hidden(Z), observed(X),
    observed(Y), L = L1 - L2, L2 < L1, lag(L1), lag(L2).
fork(X,Y,0) :- path(Z,Y,L1), path(Z,X,L2), hidden(Z), observed(X),
    observed(Y), L1 == L2, X < Y, lag(L1), lag(L2).
observed_directed(X,Y,L)  :- path(X,Y,L), observed(X), observed(Y).
observed_bidirected(X,Y,L) :- fork(X,Y,L).
#show observed_directed/3.
#show observed_bidirected/3.
```

The encoding is sound and complete for $d_{\mathcal{O}}$ and $b_{\mathcal{O}}$: it derives exactly the paths and latent forks defined above.

3.2 Inverse inference

The unit-lag form can express any observed lag representation, given enough (possibly infinitely many) latent nodes.

Lemma 1. *For any observed lag representation $G_{\mathcal{O}}$, there exists a (possibly infinite) latent graph $G_{\mathcal{V}}$ over $\mathcal{V} = \mathcal{O} \cup \mathcal{L}$ such that* $\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{L}) = G_{\mathcal{O}}$.

Proof. We construct $G_{\mathcal{V}}$ by realizing each observed lag independently. For each $(X, Y) \in \mathcal{O} \times \mathcal{O}$ and each $\ell \in d_{\mathcal{O}}(X, Y)$, add a latent chain of length ℓ from X to Y, introducing fresh intermediate nodes. For each X, Y with $X < Y$ and each $\ell \in b_{\mathcal{O}}(X, Y)$, introduce a fresh latent node with two directed latent chains to X and Y whose lengths differ by ℓ. All latent nodes are chosen to be distinct across constructions, so no unintended paths or forks are introduced. By definition of Mar, this yields exactly the prescribed lag sets on $\mathcal{O}$, hence $\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{L}) = G_{\mathcal{O}}$. □

Corollary 1 (No loss of generality)**.** *Let G' be any discrete-time causal graph over $\mathcal{O} \cup \mathcal{L}'$ whose direct effects have lags in $\mathbb{N}$, not necessarily unit lags. Then there is a unit-lag graph $G_{\mathcal{V}}$ over $\mathcal{O} \cup \mathcal{L}$ with* $\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{L})$ *equal to the observed lag-set graph of G'.*

Proof sketch. Replace each direct edge of lag $k > 1$ in G' by a chain of k unit-lag edges through $k - 1$ fresh hidden nodes $\mathcal{D}$. Each fresh node has one parent and one child, so it creates no forks, and marginalizing $\mathcal{D}$ returns G'. By Theorem 1, marginalizing $\mathcal{D} \cup \mathcal{L}'$ at once gives the same result as marginalizing $\mathcal{D}$ and then $\mathcal{L}'$. □

The latent realizations of a given $G_{\mathcal{O}}$ generally are not unique; that is, the inverse inference problem—given $G_{\mathcal{O}}$, construct a latent graph $H_{\mathcal{V}}$ such that $\mathrm{Mar}(H_{\mathcal{V}}, \mathcal{L}) = G_{\mathcal{O}}$—is generally non-identifiable. Define $H_{\mathcal{V}}$ as a *valid expanded graph* of $G_{\mathcal{O}}$ if and only if: (1) $\mathrm{Mar}(H_{\mathcal{V}}, \mathcal{L}) = G_{\mathcal{O}}$; (2) All directed edges in $H_{\mathcal{V}}$ are unit-lag; and (3) $H_{\mathcal{V}}$ contains no bidirected edges. Because multiple expanded graphs may yield the same observed compressed graph, we search for *minimal* expanded graphs, defined lexicographically: first minimize the number of latent nodes $|\mathcal{L}|$, and subsequently minimize the number of edges incident to those nodes. The result is an equivalence class of minimal structures consistent with the observed lag representation $G_{\mathcal{O}}$.

The CHRONOS algorithm formulates the inverse search as a declarative constraint-satisfaction problem. The input lag representation of $G_{\mathcal{O}}$ is provided as facts, and the ASP solver enumerates minimal latent structures consistent with them. As with the forward inference problem, the ASP formulation below operates on lag sets truncated to a fixed horizon `maxlag`.

CHRONOS searches over unit-lag graphs that extend $\mathcal{O}$ with latent nodes. We fix an upper bound H on the number of latent nodes, based on domain or problem knowledge. A candidate graph $H_{\mathcal{V}}$ is accepted if and only if its marginal representation (after marginalizing out non-$\mathcal{O}$ variables

using the rules in Listing 1) matches $G_{\mathcal{O}}$ exactly. Specifically, we provide predicates for observed directed and bidirected relations, and enforce exact agreement by closure constraints for directed and bidirected edges: every input lag must be produced, and no extra lags may be produced. To reduce grounding and avoid redundant solutions, we restrict edge generation to active latent nodes using an `enabled(H)` predicate and enforce contiguity of their identifiers.

Because solutions that differ only by relabeling latent nodes are equivalent, we add symmetry-breaking constraints that select a canonical labeling (ordering latents by degree signature with a lexicographic tie-breaker; Appendix A.7).

Listing 2 shows the key components of the inverse encoding (full ASP program in Appendix A.7, Listing 3). The first two lines pick which latents to use and number them without gaps. The next two guess edges among observed nodes and from latents to observed nodes; the full program also allows edges into and between latents, so they can form chains and cycles. We minimize latent count first, then edge count.

Listing 2: Key components of inverse ASP encoding

```
{ enabled(H) } :- hidden(H). % choose latent nodes
:- enabled(H), H > 1001, not enabled(H-1). % enforce contiguity
{ edge(X,Y,1) } :- observed(X), observed(Y). % observed-observed edges
{ edge(H,Y,1) } :- enabled(H), observed(Y).  % latent-to-observed edges
:- enabled(H), not edge(H,_,1), not edge(_,H,1). % no isolated latents
#minimize { 1@3,H : enabled(H) }.    % minimize number of latents
#minimize { 1@2,X,Y : edge(X,Y,1) }. % minimize edges
```

3.3 Constraint-relaxed inverse inference

The inverse inference formulation in Listing 2 enforces exact agreement between the observed lag representation $G_{\mathcal{O}}$ and the graph induced by marginalizing the latent nodes of a candidate latent graph $H_{\mathcal{V}}$. That is, we require that $\mathrm{Mar}(H_{\mathcal{V}}, \mathcal{L}) = G_{\mathcal{O}}$ for any solution $H_{\mathcal{V}}$. However, in practice, $G_{\mathcal{O}}$ is typically estimated from finite data and may contain either spurious or missing lags. Requiring exact agreement can therefore rule out latent graphs that are consistent with the underlying system, but differ slightly from the estimated observations. We therefore introduce a *constraint-relaxed inverse inference* formulation that allows controlled deviations from $G_{\mathcal{O}}$, while still preferring solutions that remain close to the observed lag structure.

Let $\widetilde{H}_{\mathcal{V}}$ denote a candidate latent graph, where $\mathcal{L}(\widetilde{H}_{\mathcal{V}}) = \mathcal{V}(\widetilde{H}_{\mathcal{V}}) \setminus \mathcal{O}$ denotes the latent-node set of $\widetilde{H}_{\mathcal{V}}$. Define the induced graph over observed variables: $\mathrm{Mar}(\widetilde{H}_{\mathcal{V}}, \mathcal{L}(\widetilde{H}_{\mathcal{V}})) = \widetilde{G}_{\mathcal{O}}$, where $\widetilde{d}(X, Y)$ and $\widetilde{b}(X, Y)$ respectively denote the directed and bidirected lag sets between X and Y in $\widetilde{G}_{\mathcal{O}}$. We assume that all observed lag instances are potentially uncertain, though mismatches at different lags may be weighted differently. In particular, we will often assign higher weight to shorter lags since they will typically be on stronger statistical footing, though we could also base weights on, e.g., p-values. The directed lag discrepancy D_d between $\widetilde{G}_{\mathcal{O}}$ and $G_{\mathcal{O}}$ is measured through differences in the sets of (weighted) directed lags:

$$D_d(G_{\mathcal{O}}, \widetilde{G}_{\mathcal{O}}) = \sum_{(X,Y)\in\mathcal{O}\times\mathcal{O}} \sum_{\ell \in d(X,Y) \,\triangle\, \widetilde{d}(X,Y)} w_d(\ell), \tag{1}$$

where $\triangle$ denotes symmetric difference of sets; and $w_d(\ell)$ denotes the penalty for a mismatch at directed lag ℓ. The bidirected lag discrepancy D_b is defined identically, though the second sum is over $b(X, Y)$ and $\widetilde{b}(X, Y)$, and $w_b(\ell)$ denotes the penalty at bidirected lag ℓ. The total discrepancy is then $D_{\mathcal{O}}(G_{\mathcal{O}}, \widetilde{G}_{\mathcal{O}}) = \alpha_d\, D_d(G_{\mathcal{O}}, \widetilde{G}_{\mathcal{O}}) + \alpha_b\, D_b(G_{\mathcal{O}}, \widetilde{G}_{\mathcal{O}})$, where $\alpha_d, \alpha_b \geq 0$ are weights on directed and bidirected discrepancies, respectively.

Discrepancies can always be minimized by including additional latent variables; hence, we need to balance discrepancies vs. simplicity. We introduce a penalty for exceeding a latent-node budget. Let K denote the maximal unpenalized number of latent nodes, and define the overflow penalty as a convex function that assigns no penalty when $|\mathcal{L}| \leq K$ but grows superlinearly beyond this budget. In the evaluations, we use: $\mathcal{P}(|\mathcal{L}|) = \sum_{j=1}^{|\mathcal{L}|} \max(0,\, j - K)^2$. The solver then minimizes the objective

$$\mathcal{H}^\star = \arg\min_{\widetilde{H}_{\mathcal{V}}} \left[D_{\mathcal{O}}(G_{\mathcal{O}}, \widetilde{G}_{\mathcal{O}}) + \mathcal{P}(|\mathcal{L}(\widetilde{H}_{\mathcal{V}})|) \right], \tag{2}$$

$$\widetilde{H}_{\mathcal{V}} \in \arg\min_{\widetilde{H}_{\mathcal{V}} \in \mathcal{H}^\star} |\mathcal{L}(\widetilde{H}_{\mathcal{V}})|. \tag{3}$$

Accordingly, the solver first minimizes the total cost of observed-graph discrepancy and latent-node overflow. Among ties at this stage, it minimizes the number of latent nodes. Exact inverse inference is recovered as the special case in which the optimal solution satisfies $D_{\mathcal{O}}(G_{\mathcal{O}}, \widetilde{G}_{\mathcal{O}}) = 0$.

4 Results

We evaluate CHRONOS on synthetic causal graphs under a controlled parameter sweep, analyzing tractability, reconstruction accuracy, and solution ambiguity, and then apply it to real traffic data.

4.1 Experimental Setup

We generate synthetic latent causal graphs with unit-lag directed edges (self-loops allowed) over $\mathcal{V} = \mathcal{O} \cup \mathcal{L}$. We vary the total number of variables $|\mathcal{V}| = N \in \{12, 24, 36\}$, the latent fraction $h \in \{0.1, 0.2, 0.3, 0.5\}$ (with $|\mathcal{L}| = \lfloor hN \rfloor$), and the edge density $e \in \{0.1, 0.2, 0.3, 0.5\}$.

For each configuration, we generate 100 instances. Each latent graph is forward-marginalized to produce $G_{\mathcal{O}}$, which serves as input to inverse inference. To allow slight model mismatch, we set the latent budget to $\texttt{hhmax} = |\mathcal{L}| + 2$.

A run *completes* if the solver proves optimality and enumerates the full optimal equivalence class within the time budget. Runs that exceed the budget are of two kinds: (A) an optimal solution was found but enumeration of its equivalence class was not finished; (B) optimality had not yet been proven.

4.1.1 Lag perturbation method for evaluations

To systematically test the constraint-relaxed inverse inference method, we need a way to induce "errors" in the input. Let $\mathcal{U}$ be the set of all possible directed and bidirected lag instances over $\mathcal{O}$ up to $\texttt{maxlag}$. For a given $k \geq 0$, we construct a perturbed graph $\widehat{G}_{\mathcal{O}} = \mathrm{flip}_k(G_{\mathcal{O}})$ to use as the input graph. In the unweighted flips regime, we select k elements from $\mathcal{U}$ uniformly at random, and flip their presence: existing lag instances are removed, and absent ones are added. This process does not distinguish between false negatives and false positives, nor between directed and bidirected lags. We also consider a weighted flips regime that captures the intuition that inferences about short-lag relations should be more stable by selecting with weighted probabilities that increase with the lag.

4.2 Simulations

The initial set of simulations used the "true" $G_{\mathcal{O}}$ as input. Because CHRONOS is exact, every completed run returns the true optimal equivalence class, so in this setting the only limitation is runtime and these simulations provide an empirical complexity analysis of the problem. Figure 2 shows the median runtimes (markers) with interquartile ranges (whiskers) across the full parameter sweep, using an 18000-second time budget. For small graphs ($N = 12$), all instances are solved within the time budget, with runtimes remaining consistently low across densities and hidden fractions. As N increases, completed runs exhibit heavier-tailed runtimes and increasingly cluster near the time limit, particularly at higher edge densities and larger hidden fractions. The timeout-rate panels (top row) show that at $N = 36$ a growing share of instances exceeds the budget, while those that complete still do so in moderate time. Overall, Figure 2 highlights a clear tractability boundary in (N, h, e) space, where both runtime and timeout frequency increase with structural complexity.

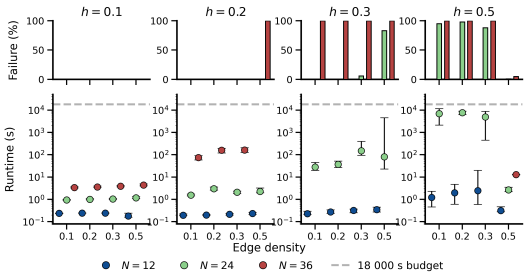


Figure 2: Runtime of completed inverse inference runs for a given edge density and hidden fraction. Top: Timeout rates.

As noted earlier, inverse inference is typically underdetermined, resulting in an equivalence class of output graphs. Unsurprisingly, equivalence class sizes grow with N, h, e, indicating increasing ambiguity in more complex settings (see Figure 8 in the Appendix). More interestingly, CHRONOS often found latent graphs that explained $G_{\mathcal{O}}$ in a simpler way than the "true" $G_{\mathcal{V}}$. For each completed

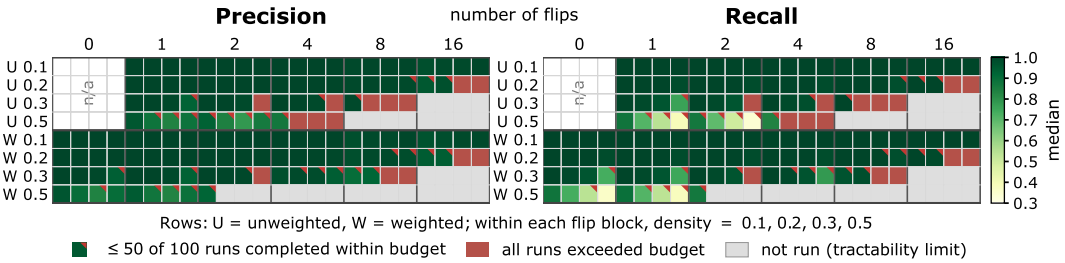


Figure 3: Reconstruction precision and recall for $N = 24$ graphs under uniform lag flips. Rows group the discrepancy penalty (U = unweighted, W = weighted) and hidden fraction h; columns group the number of flips and, within each group, densities $0.1, 0.2, 0.3, 0.5$. Color gives the median over completed runs. Corner-marked cells have at most 50 of 100 runs completed within the time budget, red cells indicate that all runs exceeded the budget, and gray cells were not run after the tractability limit was reached. The weighted setting additionally includes a zero-flip baseline.

run, we computed Δ_H, the latent count difference between $G_{\mathcal{V}}$ and graphs in the output equivalence class (which all have the same number of latents). Figure 9 (in Appendix) shows that Δ_H increases with graph size and edge density, suggesting that larger and denser underlying systems admit greater redundancy in latent-mediated pathways.

4.2.1 Constraint-relaxed inverse inference

Tests of constraint-relaxed inverse inference are more complicated. First, latent node identities are not directly comparable between $G_{\mathcal{V}}$ (the true graph) and $\widetilde{H}_{\mathcal{V}}$ (the graph learned from the perturbed $G_{\mathcal{O}}$), so we align the latent nodes of $\widetilde{H}_{\mathcal{V}}$ to those of $G_{\mathcal{V}}$ using the permutation that minimizes edge disagreement. Precision and recall are computed for this optimal alignment.

Second, perturbation levels are evaluated up to 16 flipped lag instances. A fixed setting with at least 50 of the 100 instances exceeding the time budget is treated as having reached its computational tractability limit; higher perturbation levels are then omitted unless they were already run.

In the (unweighted discrepancy, uniform flips) setting, the discrepancy penalties depend only on edge type (directed or bidirected) and, for directed edges, whether the lag is unit-length or not. That is, there are only three penalty values: unit-lag directed, non-unit-lag directed, and bidirected. The uniform flips are chosen using the method described in Section 4.1.1.

The U rows of Figure 3 summarize reconstruction quality in this setting, with precision in the left pane and recall in the right. For small fractions of latent variables ($h = 0.1, 0.2$), most instances complete within the budget and inverse inference is highly robust to perturbations. Both precision and recall remain near one, across all densities and flip levels. In this regime, the solver reliably recovers latent structures consistent with the underlying system despite the introduction of errors in the observed lag sets.

As h increases to 0.3, the fraction of runs exceeding the time budget increases substantially, especially at higher e and higher flip counts, indicating that optimality becomes increasingly difficult to establish as structural complexity grows. At $h = 0.5$, reconstruction quality deteriorates substantially: recall drops sharply, in some settings falling below 0.5. Precision remains comparatively higher (under flip perturbations, precision is at least as high as recall in most cells), indicating that errors are dominated by missed relations rather than spurious ones. Budget exceedances also become common.

Crucially, the number of flipped lags has a secondary effect on reconstruction quality compared to structural parameters. Increasing perturbation levels mainly reduces the number of runs that complete within the budget, while the precision and recall of completed runs change comparatively little. The zero-flip baseline in the W rows makes this explicit: without any perturbation, recall at $h = 0.5$ already falls from 0.87 to 0.36 as density increases while precision stays at or above 0.83. Overall behavior is therefore primarily governed by h and e. We see a similar pattern in the other two settings that we examined.

In the (weighted discrepancy, uniform flips) setting, unit-lag and small-delay relations are often more directly tied to underlying causal mechanisms, and so should be prioritized in reconstruction. Specifically, for each lag ℓ, we define a weight $w(\ell) = \texttt{maxlag} + 1 - \ell$, so that smaller lags receive larger weights. The directed and bidirected mismatch penalties (from the unweighted-discrepancy setting) are scaled by this lag-dependent factor. As a result, unit-lag directed mismatches incur

the highest penalty, and penalties decrease linearly with increasing lag. Bidirected mismatches are weighted analogously. These weights mean that the solver prioritizes agreement on short-lag structure while allowing greater flexibility on longer delays.

The W rows of Figure 3 summarize the results for $N = 24$ in the (weighted discrepancy, uniform flips) setting. At $h = 0.1$ and $h = 0.2$, performance remains near-perfect across all perturbation levels, as in the unweighted case. Weighting has little effect in this regime, since both short- and long-lag relations are already recovered accurately.

At larger h, the previous qualitative behavior persists: reconstruction remains robust for small hidden fractions and degrades sharply as hidden fraction and edge density increase. Where both settings have completed runs, median precision and recall under weighted penalties are nearly identical to the unweighted ones. The main measurable effect of weighting is computational: fewer runs complete within the budget (for example, at $h = 0.2$ with 8 flips and density 0.3, 37 of 100 runs complete under weighted penalties versus 67 under unweighted penalties). Across all parameter settings, structural parameters h and e remain the dominant factors governing both reconstruction quality and solver runtime.

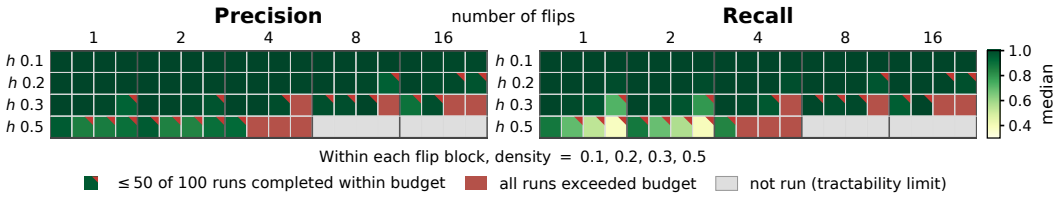


Figure 4: Reconstruction precision (left) and recall (right) for $N = 24$ graphs under weighted flip probabilities and weighted discrepancy penalties. Rows give hidden fraction h; columns group the number of flips and, within each group, densities $0.1, 0.2, 0.3, 0.5$. Encoding as in Figure 3.

Finally, we make shorter lags both less likely to be flipped and more costly to mismatch. Lag instances are selected with probabilities that increase with the lag, so perturbations are biased toward longer-lag relations. Figure 4 summarizes the resulting reconstruction quality. For small hidden fractions ($h = 0.1, 0.2$), performance remains near-perfect. The clearest difference from the uniform-flip settings is tractability under heavy perturbation: at $h = 0.2$ with 16 flips, 72 and 51 of 100 runs complete within the budget at densities 0.1 and 0.2, compared with at most 3 in either uniform-flip setting; at $h = 0.3$ with 8 flips, 50 and 27 runs complete, compared with 16 and 0 (unweighted) and 6 and 2 (weighted discrepancy, uniform flips). The precision and recall of completed runs are comparable to the other two settings, with precision again exceeding recall at $h = 0.5$. Budget exceedances remain governed primarily by structural complexity. Overall, biasing perturbations toward long lags, as suggested by statistical considerations about estimation error, keeps inverse inference tractable at substantially higher perturbation levels without degrading reconstruction quality.

4.3 Real data: METR-LA traffic

Applying CHRONOS to measured time series requires connecting statistical estimation of observed dependencies to latent-structure discovery. We evaluate this connection on METR-LA traffic data (Li et al., 2018), using the constraint-relaxed formulation to accommodate errors in estimated lag relations. We first compare estimators on traffic-matched simulations, fix their thresholds using tuning graphs, and assess estimation accuracy and downstream inversion on held-out graphs (Appendix A.8). We then apply the resulting pipeline to real detector measurements to examine the parsimonious latent explanations it recovers.

The data comprise 5-minute bins over 34,272 time steps, restricted to the 55 sensors with fewest missing values and deseasonalized by day-of-week and time-of-day. From the road-network distances accompanying the dataset we select observed sets of $N \in \{10, 12, 14\}$ sensors in two regimes: *dense*, maximizing internal road-proximity edges (a complete proximity clique, density 1.00), and *sparse*, minimizing them subject to the selected sensors remaining connected in the road-proximity graph (density 0.22, 0.20, 0.16, respectively). Dense sets group nearby detectors, where shared upstream flow makes latent common causes physically plausible; sparse sets retain road-network connectivity with fewer proximity edges.

We estimate $G_{\mathcal{O}}$ with rank-transformed vector autoregression (rank-VAR): each series is replaced by its standardized ranks (Spearman, 1904), a lag-12 VAR is fit, and, reading nonzero lagged coefficients

as directed dependence (Granger, 1969), a directed edge at lag ℓ is kept when the corresponding coefficient magnitude exceeds a tuned threshold; bidirected edges are kept where residual cross-correlations exceed a floor. We selected it over PCMCI (Runge et al., 2019) on simulations matched to METR-LA's autocorrelation and cross-dependence (Appendix A.8): rank-VAR gave higher directed F_1 at every tested latent count, its held-out scores differed from tuning scores by at most 0.02, and its estimated graphs were inverted within budget far more often (97–99% versus 61–62% of runs at 24 nodes with four or more latents). We invert with weighted constraint-relaxed CHRONOS (`maxlag` $= 12$) allowing at most two latent nodes (`hhmax` $= 2$) with no free latents ($K = 0$), so each hidden node must reduce discrepancy enough to overcome the overflow penalty.

Dense and sparse sets differ mainly in the estimated confounding signature in $G_{\mathcal{O}}$: bidirected edges fall from 19, 22, 34 (dense, $N = 10, 12, 14$) to 3, 7, 7 (sparse), while directed edge counts change far less ($72/90/110$ versus $56/71/85$). The recovered latent structure follows. On the dense $N = 10$ set, both latents attach to a single tightly interconnected subgroup of co-located detectors, with 10 latent-incident edges; on the sparse set they attach to four scattered sensors with 6 such edges. Figure 5 shows the sparse $N = 10$ case. Because the objective is constraint-relaxed, low-weight observed relations may be left unexplained rather than forcing extra structure, so some observed nodes carry no incident edge in the recovered graph. Road adjacency likewise does not guarantee a detectable dependence: in each sparse configuration one sensor appears in $G_{\mathcal{O}}$ with only a self-loop. The illustrated graph is one member of the returned equivalence class.

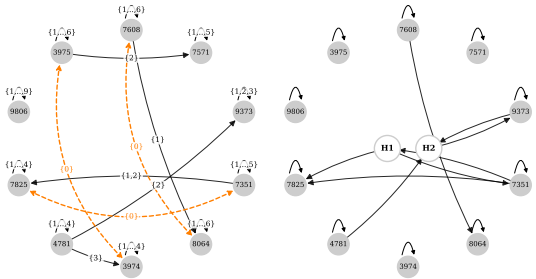


Figure 5: CHRONOS on real METR-LA traffic (sparse $N = 10$ sensor set, road-proximity density 0.22). Left: estimated $G_{\mathcal{O}}$, directed edges in black, bidirected as orange dashed arcs, each with its lag set; gray nodes are sensors (IDs truncated to the last four digits). Right: recovered $G_{\mathcal{V}}$ with inferred latents H_1, H_2 in white.

All six configurations solved to proven optimality within minutes at `hhmax` $= 2$. Sparse sets remained tractable at larger sizes, completing in about one hour at $N = 16$ and thirteen hours at $N = 18$, whereas dense sets with $N \geq 16$, all sets with $N \geq 20$, and larger latent ceilings (`hhmax` $= 6, 12$) exceeded a 24-hour budget, consistent with the synthetic finding that runtime grows with size and density.

These results demonstrate a feasible path from measured time series to explicit latent explanations within the tested size and latent-budget limits. Because the true traffic graph is unknown, the recovered structures remain candidate explanations of estimated predictive dependencies: neither the directed relations nor the residual-based bidirected relations independently establish causal effects or hidden common causes.

5 Conclusions

We recover latent causal structure from an observed lagged graph. Without loss of generality, we write the full system in a unit-lag normal form (Corollary 1). CHRONOS returns all consistent graphs minimizing latent nodes first, then edges. In the synthetic sweep, runtime and timeouts grow with graph size, latent fraction, and density. Every 12-node graph finishes, but at 36 nodes a growing share runs out of time. Completed searches can still yield many optimal graphs, increasingly so with graph size, density, and latent fraction. With exact input, the true graph is always a valid answer, so CHRONOS never needs more latent nodes than the truth. It often needs fewer, and more often in larger and denser graphs. With noisy input, the relaxed formulation still recovers the structure well when few nodes are latent. When errors fall mostly on long lags and penalties weigh short lags most, many more runs finish under heavy perturbation. On METR-LA traffic, the full pipeline, from estimating the observed graph to inverting it, finds compact latent explanations of the estimated graphs with at most two latent nodes.

AI USE STATEMENT

In this work, we used generative AI tools for none of the tasks with required disclosure. We have not used generative AI tools for [support qualitative and thematic data analysis, generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, interpret results] and [assist with translation] are not applicable to this work. Additionally, we used generative AI tools for [summarize or analyse existing literature, identify relevant literature]. We have reviewed all AI-assisted work. We asked Opus 5 to scan a large volume of literature to find published works in various areas directly related or adjacent to the paper's content. These include the most recent works related to causality experiments in time series, and works that preceded those, all the way back to the first published works about causality, so that we can get a good idea about the history and evolution of the field. After finding candidate publications for the literature review, we asked Opus 5 to summarise and compare their content to help us select the most relevant set of publications of high quality to put into this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Mohammadsajad Abavisani, David Danks, and Sergey Plis. GRACE-c: Generalized rate agnostic causal estimation via constraints. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=B_pCIsX8KL_.

Mohammadsajad Abavisani, Ksenia Solovyeva, David Danks, Vince D. Calhoun, and Sergey M. Plis. Causal graph recovery in neuroimaging through answer set programming. *arXiv preprint arXiv:2506.09286*, 2025.

Gilles Blondel, Marta Arias, and Ricard Gavaldà. Identifiability and transportability in dynamic causal networks. *International journal of data science and analytics*, 3(2):131–147, 2017.

Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pp. 294–321, 2012.

Doris Entner and Patrik O Hoyer. On causal discovery from time series data using fci. *Probabilistic graphical models*, 16, 2010.

Andreas Gerhardus and Jakob Runge. High-recall causal discovery for autocorrelated time series with latent confounders. *Advances in Neural Information Processing Systems*, 33:12615–12625, 2020.

Mingming Gong, Kun Zhang, Bernhard Schoelkopf, Dacheng Tao, and Philipp Geiger. Discovering temporal causal relations from subsampled data. In *International Conference on Machine Learning*, pp. 1898–1906. PMLR, 2015.

Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pp. 424–438, 1969.

Antti Hyttinen, Sergey Plis, Matti Järvisalo, Frederick Eberhardt, and David Danks. Causal discovery from subsampled time series data by constraint optimization. In *Conference on Probabilistic Graphical Models*, pp. 216–227. PMLR, 2016.

Erik L Jahn, Karthik Karnik, and Leonard Schulman. Causal identification in time series models. In Biwei Huang and Mathias Drton (eds.), *Proceedings of the Fourth Conference on Causal Learning and Reasoning*, volume 275 of *Proceedings of Machine Learning Research*, pp. 1435–1449. PMLR, 07–09 May 2025. URL https://proceedings.mlr.press/v275/jahn25a.html.

Erich Kummerfeld and Joseph Ramsey. Causal clustering for 1-factor measurement models. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1655–1664, 2016.

Erich Kummerfeld, Joe Ramsey, Renjie Yang, Peter Spirtes, and Richard Scheines. Causal clustering for 2-factor measurement models. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 34–49, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44851-9.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018. URL https://arxiv.org/abs/1707.01926.

Zijian Li, Yifan Shen, Kaitao Zheng, Ruichu Cai, Xiangchen Song, Mingming Gong, Guangyi Chen, and Kun Zhang. On the identification of temporal causal representation with instantaneous dependence. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 35203–35242, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5726513facc85f802be4a25e77fb9765-Paper-Conference.pdf.

Juan Miguel Ogarrio, Peter Spirtes, and Joe Ramsey. A hybrid causal search algorithm for latent variable models. In *Conference on probabilistic graphical models*, pp. 368–379. PMLR, 2016.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. Dynotears: Structure learning from time-series data. In *International conference on artificial intelligence and statistics*, pp. 1595–1605. Pmlr, 2020.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Causal inference on time series using restricted structural equation models. *Advances in neural information processing systems*, 26, 2013.

Sergey Plis, David Danks, Cynthia Freeman, and Vince Calhoun. Rate-agnostic (causal) structure learning. *Advances in neural information processing systems*, 28, 2015.

Jakob Runge. Causal network reconstruction from time series: From theoretical assumptions to practical estimation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(7), 2018.

Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances*, 5 (11):eaau4996, 2019.

Rujia Shen, Yi Guan, Liangliang Liu, Yang Yang, Boran Wang, Chao Zhao, and Jingchi Jiang. Causal discovery from time-series data with short-term invariance-based convolutional neural networks. *Mathematics*, 13(24), 2025. ISSN 2227-7390. doi: 10.3390/math13243979. URL https://www.mdpi.com/2227-7390/13/24/3979.

Ricardo Silva, Richard Scheines, Peter Spirtes, and Clark Glymour. Learning measurement models. Technical Report CMU-CALD-03-100, Carnegie Mellon University, 2003. URL https://www.cs.cmu.edu/~rbas/cmu-cald-03-100.pdf.

Ricardo Silva, Richard Scheines, Clark Glymour, and Peter L Spirtes. Learning measurement models for unobserved variables. *arXiv preprint arXiv:1212.2516*, 2012.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

Peter Spirtes, Christopher Meek, and Thomas Richardson. Causal inference in the presence of latent variables and selection bias. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pp. 499–506, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603859.

Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

Seth Sullivant, Kelli Talaska, and Jan Draisma. Trek separation for gaussian graphical models. *The Annals of Statistics*, 38(3):1665–1685, 2010. doi: 10.1214/09-AOS760.

Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896, 2008.

A Appendix

A.1 Computational setup

All experiments run on CPU only, with 8 solver threads per instance. Memory limits are 128 GB per instance for noiseless simulations and 32 GB for perturbation simulations. The real-data inversions use 64 GB per instance and a 24-hour solver time limit.

A.2 Runtime comparison across SCC decompositions

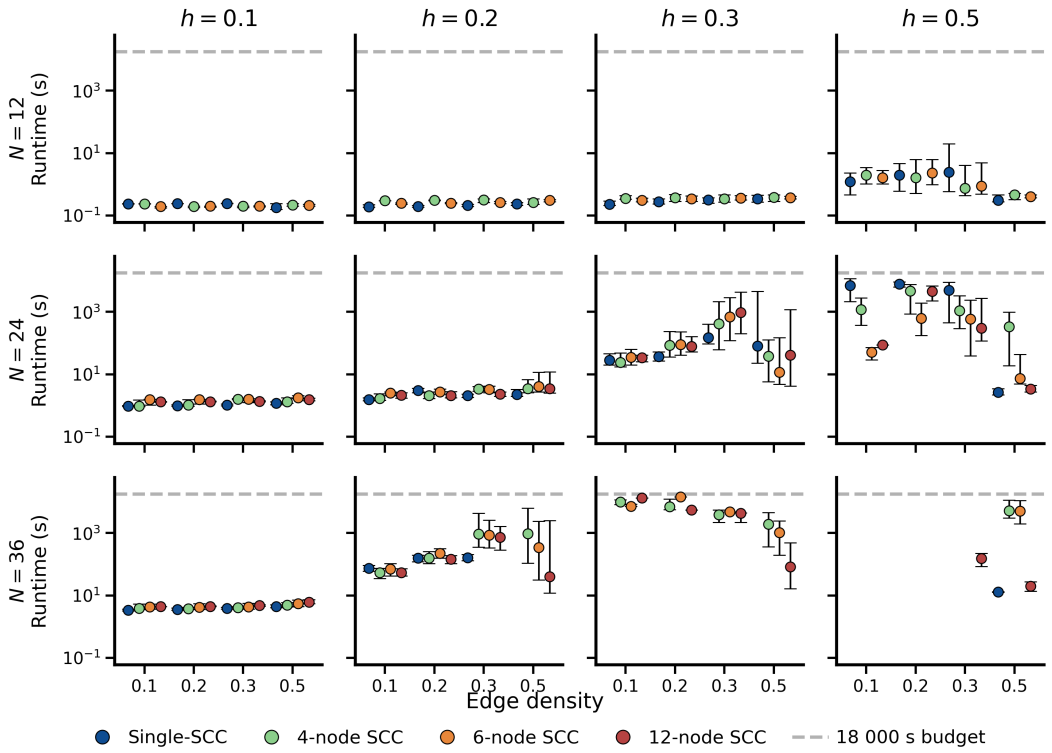


Figure 6: Runtime comparison across single-SCC and multi-SCC graph families. Rows correspond to $N \in \{12, 24, 36\}$ and columns correspond to hidden fraction h. Within each panel, median completed-run runtimes (markers) with interquartile ranges (whiskers) for edge densities $0.1, 0.2, 0.3$, and 0.5, are grouped by graph family. Across these experiments, multi-SCC graph families do not show a consistent runtime advantage over the corresponding single-SCC families.

Figure 6 compares solver runtimes for single-SCC and multi-SCC graph families across $N \in \{12, 24, 36\}$, hidden fractions $h \in \{0.1, 0.2, 0.3, 0.5\}$, and edge densities $\{0.1, 0.2, 0.3, 0.5\}$.

Overall, these results do not show a runtime benefit from decomposing the generating graph into multiple SCCs. Across the tested parameter settings, the multi-SCC graph families exhibit runtime distributions of similar scale to the corresponding single-SCC cases, and in several settings they are slightly worse. The qualitative dependence on N, h, e remains the dominant effect: runtimes increase and timeouts become more common as these parameters increase, regardless of whether the graph is a single SCC or partitioned into multiple SCCs.

This negative result is informative for interpreting the main experiments. In the present inverse-latent setting, simply introducing multiple SCCs in the generating family does not by itself make the optimization problem easier. Any computational advantage from SCC structure, if present, is not large enough to produce a clear empirical gain under the current formulation and parameter sweep.

A.3 SOLVER OUTCOMES

Figure 7 shows the distribution of solver outcomes across parameter settings, including graph families with internal (multi-SCC) structure.

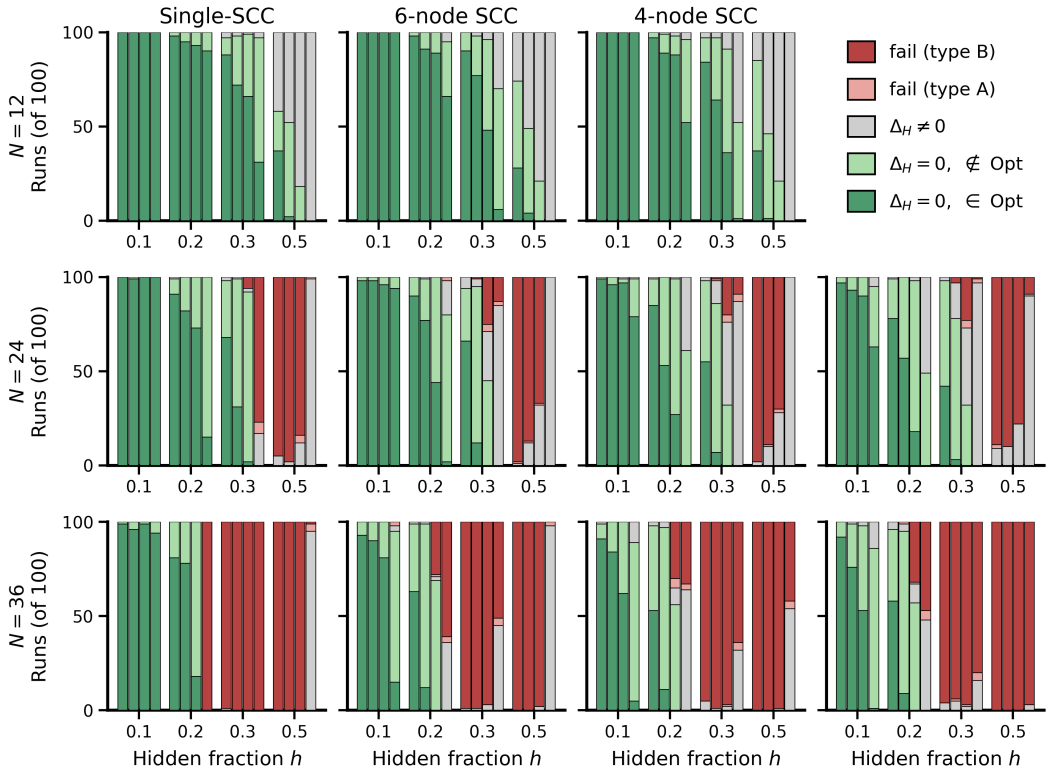


Figure 7: Outcome counts across graph families and parameter settings for $N \in \{12, 24, 36\}$. Each panel corresponds to one graph family, and within each panel the horizontal axis gives the hidden fraction $h \in \{0.1, 0.2, 0.3, 0.5\}$. For each hidden-fraction group, the four adjacent bars correspond to edge densities $0.1, 0.2, 0.3$, and 0.5. Each stacked bar summarizes 100 instances for that parameter setting. The dark green segment ($\Delta H = 0, \in \mathrm{Opt}$) denotes completed runs in which $G_{\mathcal{V}}$ itself appears in the optimal equivalence class. The light green segment ($\Delta H = 0, \notin \mathrm{Opt}$) denotes completed runs in which the recovered optimal latent count matches the generating latent count, but $G_{\mathcal{V}}$ is not among the optimal solutions. The gray segment ($\Delta H \neq 0$) denotes completed runs in which the optimal reconstruction uses a different number of latent variables than $G_{\mathcal{V}}$. The light red segment (timeout, type A) denotes runs in which the solver found at least one optimal solution but the budget expired before it finished enumerating the optimal equivalence class. The dark red segment (timeout, type B) denotes runs in which the solver had not yet proven optimality when the budget expired.

A.4 EQUIVALENCE CLASS SIZES OVER COMPLETED INVERSE INFERENCE RUNS

Figure 8 shows the distribution of equivalence class sizes across parameter settings. As discussed in the main text, equivalence classes tend to grow with graph size, edge density, and latent fraction, and exhibit heavy-tailed behavior in more complex regimes.

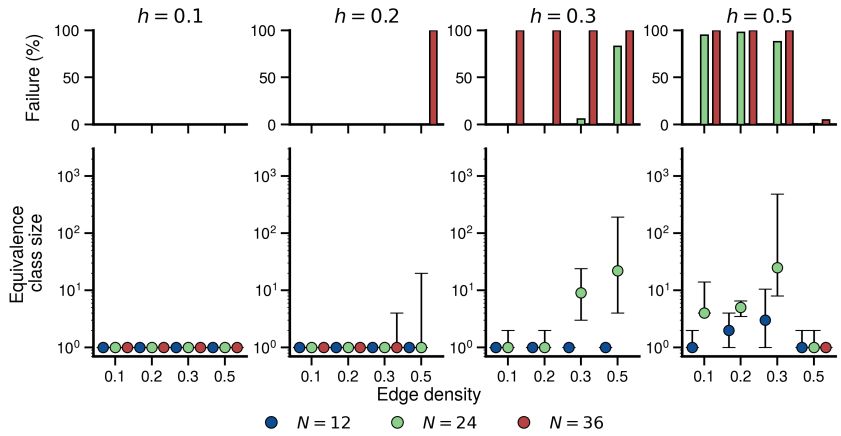


Figure 8: Equivalence class sizes over completed inverse inference runs, with $N \in \{12, 24, 36\}$ shown by color and marker shape. The bottom row reports the number of optimal latent causal graphs consistent with the observed lagged structure under the minimality objective, summarized as the median (marker) with interquartile range (whiskers) on a log scale; these are conditioned on runs that proved optimality and completed enumeration within the time budget. The top row reports the corresponding timeout rate (percentage of the 100 runs that did not complete). Columns correspond to hidden fraction $h \in \{0.1, 0.2, 0.3, 0.5\}$, and the horizontal axis within each panel is edge density.

A.5 Latent count differences over completed inverse inference runs

Figure 9 shows the distribution of latent count differences across parameter settings. Δ_H is always non-negative for optimal graphs given correct input since, in the worst case, the reconstructed graph will simply be $G_{\mathcal{V}}$.

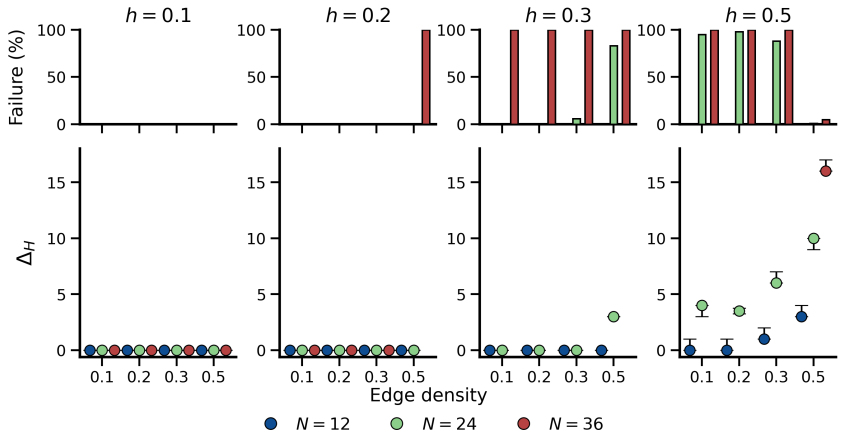


Figure 9: Latent count difference $\Delta_H = |H_{\text{true}}| - |H_{\text{rec}}|$ over completed inverse inference runs, with $N \in \{12, 24, 36\}$ shown by color and marker shape. The bottom row summarizes Δ_H as the median (marker) with interquartile range (whiskers); positive values indicate that inverse inference recovers a strictly more compact latent structure than $G_{\mathcal{V}}$ while preserving the observed lagged relations. The top row reports the timeout rate (percentage of the 100 runs that did not complete in the allocated time). Columns correspond to hidden fraction $h \in \{0.1, 0.2, 0.3, 0.5\}$, and the horizontal axis within each panel is edge density.

A.6 Proof of Theorem 1

We restate the theorem for convenience.

Theorem 1. *For all* $\mathcal{I} \subset \mathcal{V},\ x \in \mathcal{V} \setminus \mathcal{I}$: $\mathrm{Mar}(\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I}), \{x\}) = \mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I} \cup \{x\})$.

Let $\mathcal{O} = \mathcal{V} \setminus (\mathcal{I} \cup \{x\})$ denote the variables that remain observed after both marginalization steps. Define the intermediate lag-set graph $G^\star = \mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I})$ over the node set $\mathcal{O} \cup \{x\}$. We must show

that for every $A, B \in \mathcal{O}$, the directed and bidirected lag sets agree between the two sides. Because both sides are determined entirely by paths and forks in the original unit-lag graph $G_{\mathcal{V}}$, we prove the theorem by showing that the two sides characterize exactly the same sets of paths and forks.

Proof. We prove the directed and bidirected parts separately.

Part I: Directed lag sets. We show that for all $A, B \in \mathcal{O}$, the directed lag set produced by the LHS equals that produced by the RHS.

Characterization of the RHS. By definition of Mar, a lag ℓ belongs to the RHS directed lag set $d^{\mathrm{R}}(A, B)$ if and only if there exists a directed path $p : A = v_0 \to v_1 \to \cdots \to v_\ell = B$ in $G_{\mathcal{V}}$ with all intermediate nodes $v_1, \ldots, v_{\ell-1} \in \mathcal{I} \cup \{x\}$.

Characterization of the LHS. The LHS first computes $G^\star = \mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I})$ and then marginalizes x. A lag ℓ belongs to the LHS directed lag set $d^{\mathrm{L}}(A, B)$ if and only if there exists a "generalized directed path" from A to B in $G^\star$ with total lag ℓ whose intermediate nodes are all equal to x. Such a path takes one of two forms:

(a) A direct edge: $\ell \in d_{G^\star}(A, B)$; or

(b) A path through x: there exist $\ell_0 \in d_{G^\star}(A, x)$, lags $\ell_1, \ldots, \ell_{k-1} \in d_{G^\star}(x, x)$ (for some $k \geq 1$), and $\ell_k \in d_{G^\star}(x, B)$ such that $\ell = \ell_0 + \ell_1 + \cdots + \ell_k$.

We establish equality by showing containment in both directions.

$d^{\mathrm{R}}(A, B) \subseteq d^{\mathrm{L}}(A, B)$. Let $\ell \in d^{\mathrm{R}}(A, B)$, witnessed by a path p in $G_{\mathcal{V}}$ with intermediate nodes in $\mathcal{I} \cup \{x\}$. If no intermediate node of p equals x, then all intermediate nodes lie in $\mathcal{I}$, so $\ell \in d_{G^\star}(A, B)$ by definition of $G^\star$, giving form (a). Otherwise, let $0 < i_1 < \cdots < i_m < \ell$ be the positions at which $v_{i_j} = x$. Decompose p at these positions:

- Segment $A = v_0 \to \cdots \to v_{i_1} = x$ has length i_1 and intermediate nodes in $\mathcal{I}$, so $i_1 \in d_{G^\star}(A, x)$.
- For $1 \leq j < m$: segment $x = v_{i_j} \to \cdots \to v_{i_{j+1}} = x$ has length $i_{j+1} - i_j$ and intermediate nodes in $\mathcal{I}$, so $i_{j+1} - i_j \in d_{G^\star}(x, x)$.
- Segment $x = v_{i_m} \to \cdots \to v_\ell = B$ has length $\ell - i_m$ and intermediate nodes in $\mathcal{I}$, so $\ell - i_m \in d_{G^\star}(x, B)$.

The total lag is $i_1 + \sum_{j=1}^{m-1}(i_{j+1} - i_j) + (\ell - i_m) = \ell$, which is exactly form (b).

$d^{\mathrm{L}}(A, B) \subseteq d^{\mathrm{R}}(A, B)$. Let $\ell \in d^{\mathrm{L}}(A, B)$. For form (a): $\ell \in d_{G^\star}(A, B)$ means there exists a path from A to B of length ℓ in $G_{\mathcal{V}}$ with intermediate nodes in $\mathcal{I}$. Since $\mathcal{I} \subset \mathcal{I} \cup \{x\}$, this path also witnesses $\ell \in d^{\mathrm{R}}(A, B)$. For form (b): each component lag is witnessed by a path in $G_{\mathcal{V}}$ with intermediate nodes in $\mathcal{I}$: a path q_0 from A to x (length ℓ_0), paths $q_1, \ldots, q_{k-1}$ from x to x (lengths $\ell_1, \ldots, \ell_{k-1}$), and a path q_k from x to B (length ℓ_k). Concatenating $q_0, q_1, \ldots, q_k$ yields a path from A to B in $G_{\mathcal{V}}$ of length ℓ with all intermediate nodes in $\mathcal{I} \cup \{x\}$. Hence $\ell \in d^{\mathrm{R}}(A, B)$.

Since $d^{\mathrm{R}}(A, B) \subseteq d^{\mathrm{L}}(A, B)$ and $d^{\mathrm{L}}(A, B) \subseteq d^{\mathrm{R}}(A, B)$, it follows that $d^{\mathrm{R}}(A, B) = d^{\mathrm{L}}(A, B)$ for all $A, B \in \mathcal{O}$.

Part II: Bidirected lag sets. We show that for all $A, B \in \mathcal{O}$ (with the ordering convention of the main text), the bidirected lag sets agree.

Characterization of the RHS. A lag ℓ belongs to $b^{\mathrm{R}}(A, B)$ if and only if there exist a fork source $Z \in \mathcal{I} \cup \{x\}$ and paths $p_A : Z \to \cdots \to A$ (length ℓ_A) and $p_B : Z \to \cdots \to B$ (length ℓ_B), with all intermediate nodes in $\mathcal{I} \cup \{x\}$ and Z itself in $\mathcal{I} \cup \{x\}$, such that $\ell = \ell_B - \ell_A > 0$ (or $\ell = 0$ with the ordering convention $A < B$).

Characterization of the LHS. The LHS bidirected lags between A and B in $\mathrm{Mar}(G^\star, \{x\})$ arise from two sources:

(i) *Pre-existing bidirected lags*: $\ell \in b_{G^\star}(A, B)$, which correspond to forks in $G_{\mathcal{V}}$ from some $Z \in \mathcal{I}$ with both arms passing through $\mathcal{I}$ only.

(ii) *Forks through x*: either (a) x is the fork source, with directed paths in $G^\star$ from x to A and from x to B (each possibly via self-loop edges $d_{G^\star}(x, x)$), or (b) a pre-existing bidirected edge $b_{G^\star}(x, A)$ (or $b_{G^\star}(A, x)$) is combined with a directed path from x to B in $G^\star$ (or symmetrically for B).

We prove the two containments.

$b^{\mathrm{R}}(A, B) \subseteq b^{\mathrm{L}}(A, B)$. Let $\ell \in b^{\mathrm{R}}(A, B)$, witnessed by fork source $Z \in \mathcal{I} \cup \{x\}$ and paths p_A, p_B with $\ell = \ell_B - \ell_A$.

Case 1: $Z \in \mathcal{I}$ and neither p_A nor p_B visits x. Both arms have intermediate nodes in $\mathcal{I}$, so this fork exists in $G^\star$, yielding $\ell \in b_{G^\star}(A, B)$, which survives to the LHS via source (i).

Case 2: $Z = x$. Decompose each arm at revisits to x, as in Part I. The arm $p_A : x \to \cdots \to A$ decomposes into zero or more self-loop segments (each with lag in $d_{G^\star}(x, x)$) followed by a final segment with lag in $d_{G^\star}(x, A)$, totaling ℓ_A. Similarly, p_B decomposes into segments totaling ℓ_B. These decomposed arms constitute a fork through x in $G^\star$ with the required lag difference, yielding $\ell \in b^{\mathrm{L}}(A, B)$ via source (ii).

Case 3: $Z \in \mathcal{I}$ and at least one arm visits x. Suppose p_B visits x (the case where p_A visits x is symmetric; if both visit x, apply the argument to both arms). Decompose p_B at its first visit to x: let $p_B = (Z \to \cdots \to x) \circ (x \to \cdots \to B)$, where the first segment $Z \to \cdots \to x$ has length ℓ_{Zx} with intermediate nodes in $\mathcal{I}$, and the second segment decomposes (as in Case 2) into self-loops on x and a final segment to B, with total length $\ell_B - \ell_{Zx}$.

The fork from $Z \in \mathcal{I}$ with one arm to A (length ℓ_A, through $\mathcal{I}$) and another arm to x (length ℓ_{Zx}, through $\mathcal{I}$) produces the bidirected lag $|\ell_A - \ell_{Zx}| \in b_{G^\star}(A, x)$. The directed path from x to B in $G^\star$ (from the decomposed second segment of p_B) has total lag $\ell_B - \ell_{Zx}$. This is exactly source (ii)(b) in the LHS characterization, with the resulting lag difference $|(\ell_{Zx} + (\ell_B - \ell_{Zx})) - \ell_A| = |\ell_B - \ell_A| = \ell$. Hence $\ell \in b^{\mathrm{L}}(A, B)$.

$b^{\mathrm{L}}(A, B) \subseteq b^{\mathrm{R}}(A, B)$. Let $\ell \in b^{\mathrm{L}}(A, B)$.

If ℓ arises from source (i), i.e., $\ell \in b_{G^\star}(A, B)$, then it is witnessed by a fork from $Z \in \mathcal{I}$ with both arms through $\mathcal{I}$. Since $\mathcal{I} \subset \mathcal{I} \cup \{x\}$, this same fork witnesses $\ell \in b^{\mathrm{R}}(A, B)$.

If ℓ arises from source (ii), we distinguish the same two sub-cases as in the LHS characterization.

Sub-case (a): x is the fork source. The directed paths from x to A and from x to B in $G^\star$ (each composed of self-loop and final-segment lags) are witnessed by paths in $G_{\mathcal{V}}$ with intermediate nodes in $\mathcal{I}$, exactly as in the directed-lag argument of Part I. Concatenating these witnesses yields two directed paths from x to A (length ℓ_A) and from x to B (length ℓ_B) in $G_{\mathcal{V}}$, with all intermediate nodes in $\mathcal{I} \cup \{x\}$. This is a valid fork with source $Z = x \in \mathcal{I} \cup \{x\}$ and lag $|\ell_B - \ell_A| = \ell$, so $\ell \in b^{\mathrm{R}}(A, B)$.

Sub-case (b): a pre-existing bidirected edge $b_{G^\star}(x, A)$ (or $b_{G^\star}(A, x)$) is combined with a directed path from x to B. The bidirected lag $\ell_{xA} \in b_{G^\star}(x, A)$ is witnessed by a fork in $G_{\mathcal{V}}$: a source $Z \in \mathcal{I}$ with paths through $\mathcal{I}$ to x (length ℓ_{Zx}) and to A (length ℓ_{ZA}). The directed path from x to B in $G^\star$ (total lag ℓ_{xB}) is witnessed by concatenated paths in $G_{\mathcal{V}}$ with intermediate nodes in $\mathcal{I}$, as in Part I. Composing the arm $Z \to \cdots \to x \to \cdots \to B$ (length $\ell_{Zx} + \ell_{xB}$, intermediate nodes in $\mathcal{I} \cup \{x\}$) with the arm $Z \to \cdots \to A$ (length ℓ_{ZA}, intermediate nodes in $\mathcal{I}$) yields a valid fork in $G_{\mathcal{V}}$ with source $Z \in \mathcal{I}$ and the appropriate lag difference ℓ. Hence $\ell \in b^{\mathrm{R}}(A, B)$.

Since both directed and bidirected lag sets agree for all $A, B \in \mathcal{O}$, we conclude $\mathrm{Mar}(\mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I}), \{x\}) = \mathrm{Mar}(G_{\mathcal{V}}, \mathcal{I} \cup \{x\})$. □

A.7 Core inverse ASP encoding

Listing 3: Core inverse ASP encoding (grounding reduction, symmetry breaking, and optimization)

```
hidden(1000+I) :- I=1..H, hhmax(H).
node(X) :- observed(X).
```

```
node(H) :- hidden(H).
{ enabled(H) } :- hidden(H).
:- enabled(H), H > 1001, not enabled(H-1).
active_hidden(H) :- enabled(H).
{ edge(X,Y,1) } :- observed(X), observed(Y).
{ edge(H,Y,1) } :- enabled(H), observed(Y).
{ edge(X,H,1) } :- observed(X), enabled(H).
{ edge(H1,H2,1) } :- enabled(H1), enabled(H2).
:- enabled(H), not edge(H,_,1), not edge(_,H,1).
h_sig(H, Out, In, Total) :- enabled(H), Out = #count{ Y : edge(H,Y,1) },
    In = #count{ X : edge(X,H,1) }, Total = Out + In.
:- enabled(H), enabled(H+1), h_sig(H,_,_,T1), h_sig(H+1,_,_,T2), T1 < T2.
:- enabled(H), enabled(H+1), h_sig(H,O1,_,T), h_sig(H+1,O2,_,T), O1 < O2.
:- enabled(H), enabled(H+1), h_sig(H,O,I1,T), h_sig(H+1,O,I2,T), I1 < I2.
diff_out(H1,H2,N) :- enabled(H1), enabled(H2), edge(H1,N,1), not edge(H2,
    N,1), N < 1000.
diff_in(H1,H2,N)  :- enabled(H1), enabled(H2), edge(N,H1,1), not edge(N,
    H2,1), N < 1000.
:- enabled(H), enabled(H+1), h_sig(H,O,I,T), h_sig(H+1,O,I,T), M = #min{
    N : diff_out(H,H+1,N); N : diff_out(H+1,H,N) }, diff_out(H+1,H,M).
has_diff_out(H,H+1) :- enabled(H), enabled(H+1), diff_out(H,H+1,_).
has_diff_out(H,H+1) :- enabled(H), enabled(H+1), diff_out(H+1,H,_).
:- enabled(H), enabled(H+1), h_sig(H,O,I,T), h_sig(H+1,O,I,T), not
    has_diff_out(H,H+1), M = #min{ N : diff_in(H,H+1,N); N : diff_in(H+1,
    H,N) }, diff_in(H+1,H,M).
#minimize { 1@3,H : enabled(H) }.
#minimize { 1@2,X,Y : edge(X,Y,1) }.
out_edge(X,Y,1) :- edge(X,Y,1).
out_active(H) :- active_hidden(H).

#show out_active/1.
#show out_edge/3.
```

A.8 Traffic-matched simulations and estimator selection

To choose an estimator and freeze its thresholds before touching the real data, we simulated series matched to METR-LA. Each synthetic $G_{\mathcal{V}}$ (self-loops included with probability 0.1) drives a linear first-order process $x_t = Ax_{t-1} + \eta_t$ with Gaussian noise and $T = 34{,}272$. Two scalars are fitted per graph by bisection: a diagonal persistence term matching METR-LA's median lag-1 autocorrelation (0.837), and a multiplier on off-diagonal weights matching its cross-predictive gain (0.022), the increase in one-step predictive R^2 from adding other sensors' lags to a sensor's own. A is rescaled to spectral radius 0.95 at every bisection step. On METR-LA, a gradient-boosted predictor improved on a linear one by only about 0.01 in predictive R^2 and added no cross-predictive gain, supporting the linear generator; the real marginals are heavy-tailed (kurtosis 14.3), which the Gaussian generator does not reproduce but to which rank-based estimation is insensitive. Answer keys are the forward marginalization (Listing 1) of each $G_{\mathcal{V}}$.

For $N = 12$ ($k \in \{0, 1, 2, 3\}$ latents) and $N = 24$ ($k \in \{0, 2, 4, 6\}$) we generated separate tuning and held-out sets of 320 graphs per (N, k). On the tuning set we swept PCMCI's significance level (ParCorr, $\tau_{\max} = 12$) and rank-VAR's directed threshold and bidirected floor, selecting the value that maximized directed F_1 at $k = 0$ and the mean of directed and bidirected F_1 at $k > 0$. Thresholds were then frozen and applied unchanged to held-out graphs, where no score fell more than 0.01 below its tuning value. Frozen values were PCMCI $\alpha = 0.001$ at every k, and for rank-VAR a directed threshold of 0.04 at $k = 0$ and 0.06 at $k > 0$ with a bidirected floor of 0.02 at every $k > 0$. Because the latent count of real sensor sets is unknown, for METR-LA we use the fully observed directed threshold (0.04) together with the bidirected floor 0.02, the frozen value at every $k > 0$ for both N.

	Directed F_1		Bidirected F_1	
k	PCMCI	rank-VAR	PCMCI	rank-VAR
0	0.771	**0.789**	–	–
1	0.657	**0.693**	0.905	0.894
2	0.591	**0.633**	0.564	0.552
3	0.538	**0.571**	0.393	0.388

Table 1: Held-out estimator scores on 12-node traffic-matched graphs at frozen thresholds (self-loops included). Rank-VAR is higher on directed edges at every k; bidirected scores are within 0.012.

Inverting the estimated graphs with CHRONOS separates the two estimators further. At $N = 12$, every run completed for both. At $N = 24$, runs on rank-VAR graphs completed within budget in 100, 97, 99, and 97 of 100 cases at $k = 0, 2, 4, 6$, versus 100, 97, 62, and 61 for PCMCI graphs.