APPROXIMATING INCONSISTENCY AT SCALE IN PROBABILISTIC DEPENDENCY GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

We deliver significantly more performant algorithms and implementations for approximating the inconsistency of probabilistic dependency graphs (PDGs), a general problem that unifies and interpolates between many important problems in probabilistic modeling and machine learning. Previous methods are either unreliable (first-order optimizers with ad-hoc constraint penalties) or prohibitively expensive (second-order optimizers and convex solvers) for all but the smallest of PDGs. We provide the first sampling-based approach for this problem, combining a score-function gradient estimator with Rao–Blackwellization to optimize over parameterized Bayesian networks. Empirically, our method scales to substantially larger and more complex PDGs than existing approaches can handle. Our approach also naturally permits variational approximations of inconsistency, allowing the choice of Bayesian-network structure to trade computational efficiency for the tightness of the resulting upper bound.

1 INTRODUCTION

Inconsistency is a natural and often unavoidable feature of reasoning in rich, complex environments. In human societies, individuals and institutions may hold beliefs or endorse values that conflict with one another. In interactive settings—such as assistant games (Fern et al., 2014; Hadfield-Menell et al., 2016; Shah et al., 2021)—an agent's beliefs about a user's objectives may diverge from the user's true intentions. Even within a single agent, distinct context-specific models may be internally coherent but mutually incompatible and impossible to fully integrate into a unified world model. Mismatches can also arise between an agent's model and the world—for example, when a prior lacks support over an important hypothesis, or when assumed structural properties (such as independence relations) do not hold in reality. Yet as the number and complexity of these beliefs and models grow, even approximately measuring the degree of this inconsistency (let alone resolving it) poses a substantial computational challenge. Given the pervasiveness of such conflicts, it is important to develop general scalable tools for representing, measuring, and reasoning about inconsistency.

Probabilistic dependency graphs (PDGs) (Richardson & Halpern, 2021) provide a general framework for representing and working with inconsistent beliefs. PDGs are a highly expressive class of probabilistic graphical models which subsume Bayesian networks (BNs), factor graphs (FGs), as well as neural probabilistic models such as variational autoencoders (Kingma & Welling, 2014), and much of the rest of machine learning (Richardson, 2022; Richardson et al., 2026). Importantly, unlike BNs, which preserve global probabilistic consistency by construction, PDGs permit collections of local probabilistic relationships that are globally *inconsistent*, that is, that cannot be jointly realized within a single joint distribution. Beyond providing a language for representing inconsistent beliefs, Richardson & Halpern (2021) also propose a formal quantitative measure of inconsistency in PDGs. To this end, they first define an *incompatibility* function that evaluates how well a candidate joint distribution satisfies the set of local probabilistic beliefs encoded by the PDG. The *inconsistency* of a PDG is then defined as the minimum achievable incompatibility over all joint distributions, *i.e.*, the incompatibility of the most compatible distribution. Intuitively, this residual incompatibility which no carefully chosen joint can resolve represents the irreducible inconsistency in the PDG itself, and can only be resolved by altering it.

While computing the inconsistency of a PDG is, in general, NP-hard, Richardson et al. (2023) give polynomial-time algorithms for the setting where the PDGs' underlying hypergraph has bounded

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.

treewidth. The idea, building on standard techniques and representations in classical graphical models (Koller & Friedman, 2009, Ch 9–11), is to exchange the exponentially large joint distribution for a "cluster tree" over the PDG's structure. They propose two approaches for solving for the optimal tree marginals: a principled convex optimization method based on exponential conic programming, and a gradient-descent based alternative. The hardness, however, remains. Since cluster marginals grow exponentially with tree width, the approach quickly becomes intractable for even moderately complex PDGs.

To address this inference bottleneck, we develop the first sampling-based methods for approximating inconsistency via stochastic gradient descent. An important benefit of sampling-based approaches is that while existing methods strictly fail if you are unable to pay their exponential computational cost, sampling enables us to estimate gradients within a chosen computational budget, allowing for stochastic optimization even when exact updates are intractable. This trades the quality of incremental updates for lower computational cost. What makes this possible is optimizing over parameterized Bayesian networks (BNs) rather than over cluster marginals. BNs can be much more compact than cluster marginals, and they automatically define coherent joint distributions, avoiding the need to enforce calibration between factors via hard constraints or unreliable penalty terms. Most importantly, it is very efficient to sample from a Bayesian Network: simply traverse in topological order and sample conditional on already-sampled parents.

Optimizing over general parameterized Bayesian networks also enables a natural class of variational approximations of inconsistency. The BN structure can be chosen to reproduce the family represented by the cluster tree decomposition, in which case it inherits its benefits of possessing no non-optimal finite-logit local minima (Appendix B.4). Alternatively, we can impose additional conditional independencies based on domain knowledge or computational constraints, which may well already be present in the PDG itself. Optimization then searches a restricted family, whose minimum incompatibility provides an upper bound on the PDG's inconsistency, with the bound being tight when the family contains a global minimizer. The choice of BN structure therefore offers a trade-off between computational efficiency and the accuracy of the variational upper bound.

Furthermore, these additional conditional independencies can themselves represent substantive modeling beliefs whose compatibility with the supplied probabilities is important to assess. The variational problem is equivalent to computing the inconsistency of a specific PDG that encodes these structural beliefs, enforced as hard constraints. Existing methods have not explored this space of PDG inconsistency evaluation problems because they are not guaranteed a convex loss landscape. Our algorithms thus are the first to address this regime of inconsistency problems.

We derive the incompatibility gradient for a logit-parameterized BN and construct plug-in and Rao–Blackwellization-based estimators using forward samples. Source-coverage regularization discourages premature concentration before conditionals are learned. Both practical estimators have finite-sample bias because estimated probabilities enter a logarithm; we characterize the plug-in bias at fixed parameters. Our GPU implementations are evaluated against exact-gradient and cluster-tree baselines on controlled near-consistent PDGs and noisy, separately estimated beliefs, comparing achieved incompatibility and computational cost.

2 Related Work

Separately specified conditionals have long been reconciled through KL-based fitting and sampling (Arnold & Gokhale, 1998; Heckerman et al., 2000; Chen et al., 2011). PDGs generalize these settings and give an explicit inconsistency objective (Richardson & Halpern, 2021; Richardson, 2022). Existing PDG solvers optimize calibrated cluster marginals (Richardson et al., 2023); their cost grows exponentially with cluster width. Our choice of a candidate BN instead resembles structured variational inference (Hoffman & Blei, 2015), but fits supplied conditional beliefs rather than a posterior. It also differs from likelihood-based BN learning with missing data using EM (Dempster et al., 1977; Lauritzen, 1995) or gradients (Binder et al., 1997): candidate-weighted, reverse-KL terms need not form a data likelihood. Our gradients build on score-function estimation, control variates, and Rao–Blackwellization (Williams, 1992; Ranganath et al., 2014; Greensmith et al., 2004; Liu et al., 2019). Estimating the conditionals inside the logarithmic PDG loss introduces an additional bias absent when a sampled loss is evaluated exactly. Appendix A gives a fuller comparison.

3 PRELIMINARIES

We use finite variables, directed hyperedges $S \to T$, and directed acyclic graphs (DAGs). In a BN DAG, $\mathrm{Pa}_G(X)$ denotes the parents of X; its generated hypergraph contains one edge $\mathrm{Pa}_G(X) \to X$ per variable. Appendix B gives formal graph and cluster-tree definitions.

Probabilities and Variables. We write ΔS to denote the set of probability distributions over a finite set S. For distributions $p, q \in \Delta S$ over the same set, we write $\boldsymbol{D}(p \parallel q) := \sum_{s \in S} p(s) \log \frac{p(s)}{q(s)}$ for the relative entropy (KL divergence) of p relative to q. A variable X can take on values from a finite set $\mathcal{V}X$ of possible values. We can regard sets of variables $\boldsymbol{X}$ as variables themselves, with $\mathcal{V}\boldsymbol{X} = \prod_{X \in \boldsymbol{X}} \mathcal{V}X$. A conditional probability distribution (cpd) $\mathbb{P}(Y \mid X)$ is a map $\mathbb{P} : \mathcal{V}X \to \Delta\mathcal{V}Y$. Overloading notation, we write $\Delta\mathcal{V}(Y \mid X)$ to denote the set of cpds over Y given X.

Graphical models. A BN pairs a DAG with local cpds and represents $\mathbb{P}(\mathcal{X}) = \prod_{X \in \mathcal{X}} \mathbb{P}_X(X \mid \mathrm{Pa}_G(X))$. It encodes conditional independencies as well as probabilities (Pearl, 1988). A PDG instead permits arbitrary overlapping conditional beliefs.

Definition 1. A *PDG* $\mathcal{M} \doteq (\mathcal{X}, \mathcal{A}, \mathbb{P}, \boldsymbol{\beta}, \mathcal{A}_0)$ consists of a directed hypergraph $(\mathcal{X}, \mathcal{A} \cup \mathcal{A}_0)$ whose nodes are variables and with two kinds of arcs: structural ($\mathcal{A}_0$), and observational ($\mathcal{A}$). Observational arcs are labeled with cpds ($\mathbb{P}$) and confidence values ($\boldsymbol{\beta}$), so that each $S \xrightarrow{a} T \in \mathcal{A}$ is associated with:

- a cpd $\mathbb{P}_a(T \mid S)$ on the target variables given values of the source variables, and
- a weight $\beta_a \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ indicating the modeler's confidence in the cpd $\mathbb{P}_a(T \mid S)$.

Definition 2. The (observational) incompatibility $\mathrm{Inc}_{\mathcal{M}}(\mu)$ of a PDG $\mathcal{M}$ with respect to a candidate joint distribution $\mu \in \Delta\mathcal{V}\mathcal{X}$ is defined as

$$\mathrm{Inc}_{\mathcal{M}}(\mu) = \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \boldsymbol{D}\Big(\mu(S,T) \;\Big\|\; \mathbb{P}_a(T \mid S)\mu(S)\Big)$$

Incompatibility weights each belief discrepancy by its confidence. Here qualitative BN independencies are imposed by the candidate family; the loss contains observational terms only. The *inconsistency* is

$$\langle\!\langle \mathcal{M} \rangle\!\rangle \doteq \min_{\mu \in \Delta\mathcal{V}\mathcal{X}} \mathrm{Inc}_{\mathcal{M}}(\mu). \tag{1}$$

A PDG is *consistent* when this minimum is zero; an optimizing joint reconciles its local beliefs.

Optimizing calibrated cluster marginals suffices for unrestricted observational inconsistency (Richardson et al., 2023), but those tables grow exponentially with decomposition width. We instead use a BN to represent the candidate joint, allowing a more compact variational family.

4 METHODOLOGY

We now describe our method, which significantly scales the inference for PDGs (Richardson et al., 2023) (*i.e.*, inconsistency calculation) to much larger models, and yields a modest extension beyond the standard tractable parameter region (*i.e.*, does not require the condition $\mathcal{A}_0 \subseteq \mathcal{A}$). Roughly speaking, our algorithm works as follows: we parameterize distributions $\mu \in \Delta\mathcal{V}\mathcal{X}$ with Bayesian networks, whose parameters we learn via gradient descent on the incompatibility loss using sample-based gradient estimators and a decaying coverage regularizer. We now formalize this more precisely.

4.1 PARAMETERIZED BAYESIAN NETWORKS

A parameterized BN has a fixed DAG and differentiable local conditional distributions, inducing a joint μ_θ for each parameter vector $\theta \in \Theta$. The formal definition and discussion of complete parameterizations are in Appendix C. We focus on the *logit-parameterization*: assign a separate logit $\theta_{T=t|S=s}$ to each entry of each local cpd, giving $d = \sum_{S \xrightarrow{a} T \in \mathcal{A}} |\mathcal{V}S \times \mathcal{V}T|$ parameters. Then define the local conditional probabilities with a softmax:

$$\mu_{a,\theta}(t \mid s) = \frac{\exp\left(\theta_{T=t|S=s}\right)}{\sum_{t' \in \mathcal{V}T} \exp\left(\theta_{T=t'|S=s}\right)}.$$

We call a BN parameterized in this way a *logit-parameterized* Bayesian network. This parameterization is differentiable and interior-complete.

4.2 Variational Approximation of Inconsistency

Consider a PDG $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathbb{P}, \boldsymbol{\beta})$ and an interior-complete parameterized BN $\mathcal{B}_\Theta = (\mathcal{X}, \mathcal{E}, \boldsymbol{\mu}, \Theta)$. The parameterized BN $\mathcal{B}_\theta$ defines a *variational family* $\{\mu_\theta : \theta \in \Theta\} \subseteq \Delta\mathcal{V}\mathcal{X}$, *i.e.*, the set of strictly positive joint distributions that factorize according to the directed hypergraph $(\mathcal{X}, \mathcal{E})$, and hence satisfy the conditional independencies it encodes (each target set is independent of its non-descendants given its sources). This variational family can be used to learn a variational approximation of the inconsistency. In place of the inconsistency $\langle\!\langle \mathcal{M} \rangle\!\rangle = \min_{\mu \in \Delta\mathcal{V}\mathcal{X}} \mathrm{Inc}_\mathcal{M}(\mu)$, we solve the restricted problem,

$$\langle\!\langle \mathcal{M} \rangle\!\rangle^\Theta \doteq \inf_{\theta \in \Theta} \mathrm{Inc}_\mathcal{M}(\mu_\theta)\,.$$

Because the family is a subset of $\Delta\mathcal{V}\mathcal{X}$, the *variational inconsistency* $\langle\!\langle \mathcal{M} \rangle\!\rangle^\Theta$ is an upper bound on $\langle\!\langle \mathcal{M} \rangle\!\rangle$, with equality when an incompatibility minimizing distribution lies within the family. The gap between the two values measures the additional inconsistency incurred by assuming the given independence structure[1].

A BN equivalent to a cluster-tree decomposition preserves the unrestricted optimum; a sparser, expert-chosen BN trades representational flexibility for efficiency. Appendix C elaborates these choices.

4.3 Incompatibility Gradient Estimation

We approach the variational optimization problem by gradient descent on θ, which requires the gradient $\nabla_\theta \mathrm{Inc}_\mathcal{M}(\mu_\theta)$. Computing this gradient exactly is in general intractable. Below, we derive an estimate of this gradient that is amenable to sample-based approximation. For all results that follow, we assume that the probabilistic beliefs encoded in the PDG each have support over the entire domain of their target variables (*i.e.*, $\mathbb{P}_a(t \mid s) > 0$ for all $(s,t) \in \mathcal{V}S \times \mathcal{V}T$)[2], and have finite edge-confidence, *i.e.*, $\beta_a < \infty$.

Theorem 1. *It holds that*

$$\nabla_\theta \mathit{Inc}_\mathcal{M}(\mu_\theta) = \mathop{\mathbb{E}}_{z \sim \mu_\theta}\left[l(z) \cdot \nabla_\theta \log \mu_\theta(z)\right] \qquad \textit{where} \qquad l(z) \doteq \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \log \frac{\mu_\theta(z_T \mid z_S)}{\mathbb{P}_a(z_T \mid z_S)}.$$

We call $l : \mathcal{V}\mathcal{X} \to \mathbb{R}$ the *scoring statistic* of the PDG $\mathcal{M}$ relative to parametric BN μ_θ. Proofs are in Appendix D.

Corollary 2. *If μ_θ is logit-parameterized, then for each edge $X \to Y \in \mathcal{E}$ and $(x,y) \in \mathcal{V}X \times \mathcal{V}Y$, we have that*

$$\nabla \mathit{Inc}_\mathcal{M}(\mu_\theta)|_{\theta_{y|x}} = \mathop{\mathbb{E}}_{z \sim \mu_\theta}\left[\mathbf{1}_{z_X = x} \cdot (\mathbf{1}_{z_Y = y} - \mu_\theta(y \mid x)) \cdot l(z)\right]$$

The gradient in Theorem 1, specialized to logit parameters in Corollary 2, has the *score-function* (REINFORCE) form (Williams, 1992) for the expectation $\mathrm{Inc}_\mathcal{M}(\mu_\theta) = \mathbb{E}_{z \sim \mu_\theta}[l(z)]$. Although the scoring statistic $l(z)$ itself depends on θ, its expected derivative vanishes. This is the same cancellation used to derive the score-function gradient of the evidence lower bound (ELBO) in black-box variational inference (Ranganath et al., 2014).

[1] In fact, this inconsistency upper bound corresponds precisely to the inconsistency of a PDG with the given independence structure encoded via qualitative beliefs and enforced absolutely by pushing $\gamma \to \infty$.

[2] We could conceivably handle absolute (deterministic) constraints by optimizing with respect to ε-near-deterministic approximations of the beliefs, and then annealing $\varepsilon \to 0$ as training proceeds; however, we leave exploring this possibility for future work.

Consequently, provided that $l(z)$ can be evaluated exactly, independent samples $z^{(1)}, \ldots, z^{(N)} \sim \mu_\theta$ yield the unbiased Monte Carlo gradient estimator

$$\widehat{g}(\theta) \doteq \frac{1}{N} \sum_{i=1}^{N} l(z^{(i)}) \nabla_\theta \log \mu_\theta(z^{(i)}), \qquad \mathbb{E}[\widehat{g}(\theta)] = \nabla_\theta \mathrm{Inc}_{\mathcal{M}}(\mu_\theta),$$

Generating these samples is computationally cheap (one local draw per variable) in a Bayesian network: forward sampling visits the network in topological order, drawing each target from its local conditional distribution once its sources have been sampled. In the logit-parameterized setting, the score $\nabla \log \mu_\theta(z)$ is likewise computable from local cpds of the Bayesian network, with the explicit form given in Corollary 2.

The remaining difficulty is evaluating $l(z)$. Standard black-box variational inference assumes that the pointwise loss $l_{\text{BBVI}}(z) = \log p(x, z) - \log q_\theta(z)$ can be evaluated exactly at samples $z \sim q_\theta$. Here, however, the pointwise loss depends on $\log \mu_\theta(t \mid s)$ for each PDG edge $S \xrightarrow{a} T \in \mathcal{A}$. These conditionals need not coincide with the local cpds of the variational BN, and so evaluating them entails Bayesian-network conditional inference, which is NP-hard in general (Cooper, 1990). We therefore aim to approximate these conditional log-probabilities using samples.

Because the logarithm makes conditional estimation biased at finite sample sizes, we seek a low-cost compromise between bias and variance (Appendix D). We consider empirical conditional frequencies and a Rao–Blackwellization-based alternative.

Plug-in estimator. Given N independent forward samples $z^{(1)}, \ldots, z^{(N)} \sim \mu_\theta$, define the empirical counts $\hat{c}(U = u) \doteq \sum_{i=1}^{N} \mathbf{1}_{z_U^{(i)} = u}$. For each PDG edge $S \xrightarrow{a} T \in \mathcal{A}$, we estimate its conditional probabilities by $\hat{\mu}_{\text{plug-in}}(t \mid s) \doteq \frac{\hat{c}(s,t)}{\hat{c}(s)}$, whenever $\hat{c}(s) > 0$. Substituting these estimates into the scoring statistic gives

$$\widehat{l}_{\text{plug-in}}(z) \doteq \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \log \frac{\hat{\mu}_{\text{plug-in}}(z_T \mid z_S)}{\mathbb{P}_a(z_T \mid z_S)}, \quad \widehat{g}_{\text{plug-in}}(\theta) \doteq \frac{1}{N} \sum_{i=1}^{N} \widehat{l}_{\text{plug-in}}(z^{(i)}) \nabla_\theta \log \mu_\theta(z^{(i)}).$$

Note that since we only compute $\widehat{l}_{\text{plug-in}}(z)$ for $z \sim \mu_\theta$, we have that $\hat{\mu}_{\text{plug-in}}(z_T \mid z_S) > 0$, and since we also have $\mathbb{P}_a(z_T \mid z_S) > 0$ by assumption, the above gradient estimator is always well-defined. Although $\widehat{g}_{\text{plug-in}}$ is generally biased, it is still consistent: at fixed θ, it converges almost surely to $\nabla_\theta \mathrm{Inc}_{\mathcal{M}}(\mu_\theta)$ as $N \to \infty$, with bias decaying in $O(N^{-1})$.

Lemma 1 (Bias of plug-in estimator)**.** *For every fixed θ, the plug-in gradient estimator satisfies*

$$\|\mathbb{E}[\widehat{g}_{\textit{plug-in}}(\theta)] - \nabla_\theta \textit{Inc}_{\mathcal{M}}(\mu_\theta)\| \in O(N^{-1}),$$

where $\|\cdot\|$ denotes the Euclidean norm.

Rao–Blackwellization-based estimator. For each source configuration s, the plug-in estimator can be understood to estimate $\mu_\theta(t \mid s)$ by averaging the point-mass distributions

$$\hat{\mu}^{(i)}_{\text{plug-in}}(t \mid s) \doteq \mathbf{1}_{z_T^{(i)} = t}$$

over samples with $z_S^{(i)} = s$. We can reduce the variance of these probability estimates by replacing each indicator with a tractable conditional expectation, a form of Rao–Blackwellization (Casella & Robert, 1996).

Specifically, choose a set of variables $W \subseteq \mathcal{X}$ such that $\mu_\theta(t \mid s, w)$ can be evaluated efficiently for value configurations $(s, t, w) \in \mathcal{V}S \times \mathcal{V}T \times \mathcal{V}W$. This quantity is of course generally intractable to evaluate for arbitrary W, as it depends on general BN conditional inference; however, as we show below, careful selections of W exploiting the structure of the variational BN can nonetheless yield very inexpensive inference problems. For each sample with source configuration s, define

$$\hat{\mu}^{(i)}_{\text{RB}}(t \mid s) \doteq \mathbb{E}\Big[\mathbf{1}_{z_T^{(i)} = t} \,\Big|\, z_S^{(i)} = s,\, z_W^{(i)}\Big] = \mu_\theta\Big(t \mid s, z_W^{(i)}\Big).$$

Each sample thus now contributes a conditional distribution over the target configurations. Conditional on $z_S^{(i)} = s$, this contribution has the same mean $\mu_\theta(t \mid s)$ as the indicator, and no greater variance. Averaging these contributions gives

$$\hat{\mu}_{\text{RB}}(t \mid s) \doteq \frac{1}{\hat{c}(s)} \sum_{i : z_S^{(i)} = s} \hat{\mu}_{\text{RB}}^{(i)}(t \mid s).$$

We consider two choices of W. First, take $W = \mathcal{X} \setminus (S \cup T)$, so that each sample contributes the conditional distribution of T given all remaining variables:

$$\hat{\mu}_{\text{RB}}^{(i)}(t \mid s) = \mu_\theta\left(t \mid z^{(i)}_{\mathcal{X} \setminus T}\right) = \mu_\theta\left(t \mid z^{(i)}_{\text{MB}(T)}\right).$$

Here, $\text{MB}(T)$ denotes the *Markov blanket* of T: the variables outside T that appear in a local BN cpd together with a variable in T. We can compute this conditional by multiplying the cpds involving T, with all other variables fixed at their sampled values, and normalizing over $\mathcal{V}T$. Since no variables outside T need to be marginalized out, the computation remains inexpensive even in Bayesian networks with large treewidth.

Under an additional structural condition, there is a second choice of W which yields probability estimates with no greater and potentially much lower variance than the Markov-blanket-based choice. For clarity and brevity, we describe this alternative here for BNs and PDGs with only single-target hyperedges; Appendix E gives the generalization to BNs and PDGs with multi-target hyperedges. It is worth noting, the single-target setting is general: any BN or PDG can be converted to this form by chain-rule decomposition while preserving, respectively, its joint distribution / independence constraints or its incompatibility function[3].

Since by assumption our BN edges have only a single target, there exists a BN hyperedge $X \xrightarrow{b} T \in \mathcal{E}$ with some $X \subset \mathcal{X}$. If S contains no descendants of T, take W to be $X \setminus S$, the parents of T in $\mathcal{B}$, discounting variables already in S. By the local Markov property, we have

$$\hat{\mu}_{\text{RB}}^{(i)}(t \mid s) = \mu_\theta\left(t \mid s, z_W^{(i)}\right) = \mu_{b,\theta}\left(t \mid z_X^{(i)}\right).$$

The final expression is a local cpd entry, requiring no additional inference. If $\mathcal{B}_\theta$ is logit-parameterized, this probability is simply obtained through a $O(|\mathcal{V}T|)$ softmax over the corresponding logits. This condition holds for at least $1/(n+1)$ of uniformly sampled single-target edges with n sources (Appendix E).

Thus, for our Rao–Blackwellization-based gradient estimator, we assign to each PDG edge $a \in \mathcal{A}$ a conditioning set W_a, chosen by the second BN parent set approach if the required condition is met, and the Markov-blanket approach otherwise, and obtain $\hat{\mu}_{\text{RB}}(t \mid s) = \frac{1}{\hat{c}(s)} \sum_{i : z_S^{(i)} = s} \mu_\theta(t \mid s, z_{W_a}^{(i)})$. Substituting these again into the pointwise loss, we get as before,

$$\widehat{l}_{\text{RB}}(z) \doteq \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \log \frac{\hat{\mu}_{\text{RB}}(z_T \mid z_S)}{\mathbb{P}_a(z_T \mid z_S)}, \qquad \widehat{g}_{\text{RB}}(\theta) \doteq \frac{1}{N} \sum_{i=1}^{N} \widehat{l}_{\text{RB}}(z^{(i)}) \nabla_\theta \log \mu_\theta(z^{(i)}).$$

Like the plug-in gradient estimator, this estimator is consistent but generally biased. While the variance reduction for the probability estimates does not by itself guarantee lower bias or variance for the resulting gradient estimator, we empirically find gradient descent via $\widehat{g}_{\text{RB}}$ to converge faster than $\widehat{g}_{\text{plug-in}}$, as we will show in the results section.

Coverage regularization to avoid early collapse Both estimators use forward samples, so a source assignment s receives $N\mu_\theta(s)$ samples on average. This allocation matches the weighting in the incompatibility objective:

$$\text{Inc}_{\mathcal{M}}(\mu_\theta) = \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \sum_s \mu_\theta(s)\, D_{\text{KL}}(\mu_\theta(T \mid s) \,\|\, \mathbb{P}_a(T \mid s)).$$

[3]For PDGs, each edge created by decomposing an edge a retains its weight β_a. Preservation of incompatibility then follows from the chain rule for KL divergence.

Our goal is therefore to accurately estimate a probability-weighted aggregate, rather than estimate every conditional equally accurately.

However, optimization can reduce incompatibility by making poorly matched source assignments less probable, before learning their associated conditionals. These assignments then receive fewer samples, further weakening the signal available to improve their conditionals. This feedback can cause premature concentration and hinder subsequent optimization. To discourage this behavior, we add a source-coverage regularizer:

$$\mathcal{L}_\lambda(\theta) = \mathrm{Inc}_{\mathcal{M}}(\mu_\theta) - \lambda \sum_{S \xrightarrow{a} T \in \mathcal{A}} \sum_{s \in \mathcal{V}S} \log \mu_\theta(s).$$

Analogously to the logarithmic barriers used in interior-point methods, this term discourages approaching the boundary where source probabilities vanish. We decrease λ geometrically during training, encouraging coverage early and progressively relaxing its influence as optimization proceeds.

Implementation. Both GPU estimators center the sampled payoff with a global baseline; Appendix D explains its same-batch correction. Updates use forward sampling and local BN calculations without variable elimination. For numerical stability, training and evaluation clip logarithm arguments at 10^{-10}, including zero empirical belief entries. Reported incompatibilities exclude coverage. The identities above concern positive, unclipped distributions; numerical details appear in Appendix F.

5 Results

We evaluate our algorithms on synthetic PDGs generated from a base assortment of randomly generated and real-world Bayesian networks. For a given base BN, we take the BN's own graph to provide the structure for the parameterized Bayesian network, representing the generative hypergraph that an expert may have suggested for the corresponding PDG, and use the BN's joint distribution to inform the probabilistic distributions encoded in randomly added PDG edges.

5.1 PDGs from Random Width-Controlled Bayesian Networks

Our first family contains 3,275 BNs: 655 random structures with 10–50 nodes and induced widths 1–18, each instantiated at cardinalities 2–6 with fresh Dirichlet CPTs. Each PDG contains $2n$ conditional beliefs obtained from its base BN, exactly where feasible and otherwise from 4.096 billion Rao–Blackwellized samples, with independent confidences $\beta_e \sim \mathrm{Exp}(1) + 0.01$. Exactly computed beliefs give zero inconsistency; sampled beliefs may retain a small residual. The base BN is therefore a zero- or near-zero-incompatibility witness, making optimization error easier to assess than when the minimum is unknown. Appendix F gives the full construction.

We evaluate a random sample of 1,627 PDGs from this collection. Candidate CPTs start uniformly. We compare plug-in and Rao–Blackwellized (RB) gradients, each with and without coverage regularization, to JAX differentiation of exact incompatibility. All five use Adam (learning rate 0.01, $\epsilon = 10^{-12}$) for up to $100{,}000$ updates; stochastic methods draw $128{,}000$ samples per update and use a global baseline. The coverage coefficient decays geometrically from 0.03 to 10^{-5} for plug-in and 10^{-6} for RB. Stochastic runs use one GPU; exact JAX uses CPUs with a 25-minute optimization limit. All four stochastic configurations complete these 1,627 PDGs; JAX yields final scores for 1,264, of which 697 complete all updates, under the resource limits in Appendix F. We also evaluate the cluster-tree methods of Richardson et al. (2023) under 25-minute limits; the figures below compare the five BN-parameterized methods.

We use nd^w as a proxy for the cost of exact inference on the base graph. In Figure 1, JAX attains the lowest incompatibility on small PDGs, but its cost rises steeply and its scored cohort thins at larger sizes. The stochastic methods scale more gently; the slowest plotted run takes 29.1 minutes. Across all 1,627 PDGs, the largest reported RB incompatibilities are 0.533 nats without coverage and 0.417 with coverage: both variants keep every run below 0.6 nats across the full size and width range. Plug-in leaves larger residuals, though coverage substantially improves it; coverage's effect on RB varies.

On the same 1,264-PDG subset, the convex baseline returns a solution in 74 cases (37 flagged inaccurate), while cluster-tree Adam completes no full run; Appendix F.3 details the failures.

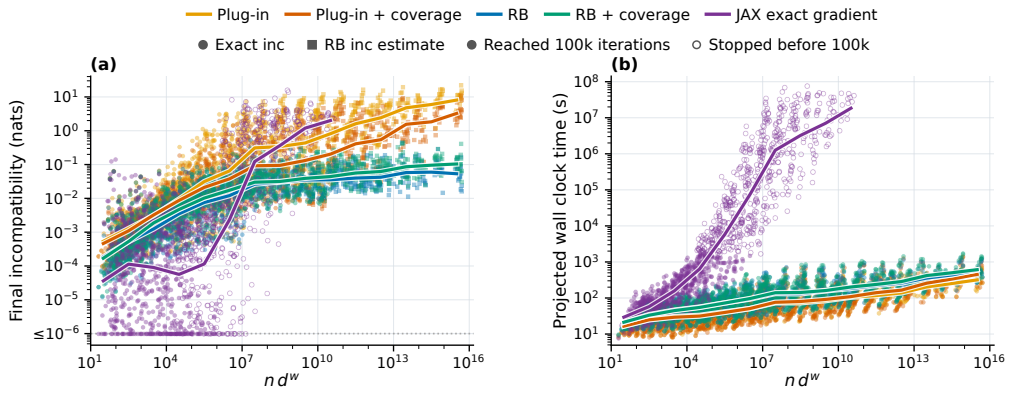


Figure 1: Width-grid PDGs versus nd^w. **(a)** Final incompatibility (circles: exact evaluation; squares: 1.024×10^9-sample RB estimate). **(b)** Compilation-plus-training time to 100,000 updates; hollow markers project early-stopped runs using observed iteration time. Lines are geometric means within size bins. Four sampling series each have 1,627 runs; JAX has 1,264.

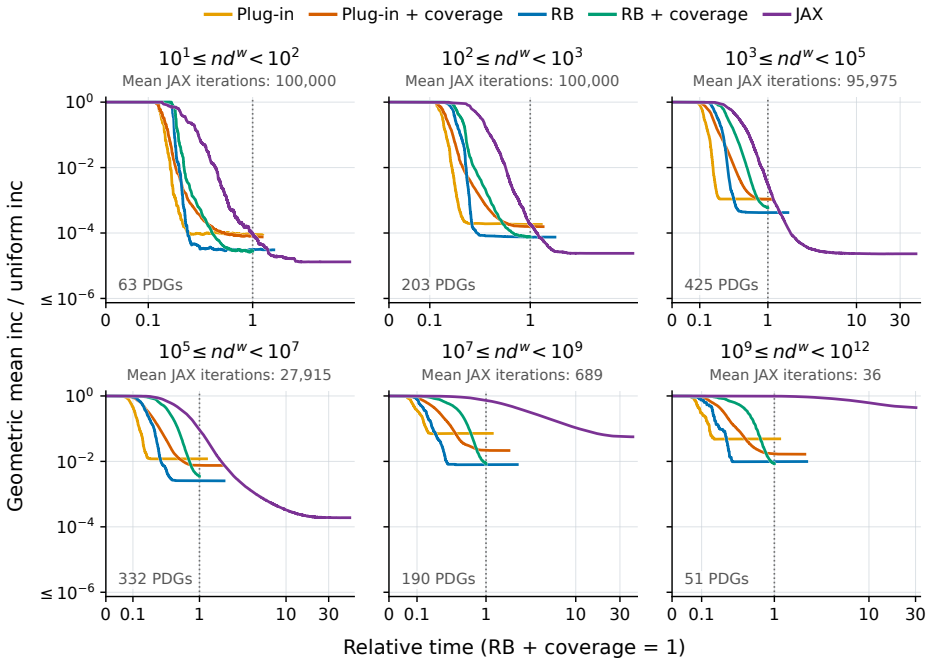


Figure 2: Geometric-mean incompatibility divided by uniform-initialization incompatibility, for six nd^w ranges (63, 203, 425, 332, 190, 51 PDGs). Time is relative to RB with coverage on the same PDG (dotted line). Completed runs retain their final score; ratios are floored at 10^{-6} before averaging. Panel headings report mean completed JAX iterations.

Figure 2 highlights rapid early learning: RB and plug-in, particularly with coverage, reach useful incompatibilities much sooner than JAX. JAX can subsequently refine the fit on smaller PDGs, but on the two largest displayed ranges its time limit permits too few updates to match RB. These results motivate a hybrid strategy for future evaluation: quickly learn a candidate BN with a sampling-based method, then initialize JAX with its CPTs for further refinement wherever exact gradients are affordable. Coverage can delay initial progress while avoiding high plug-in plateaus. These curves use the common cohort with scored JAX runs.

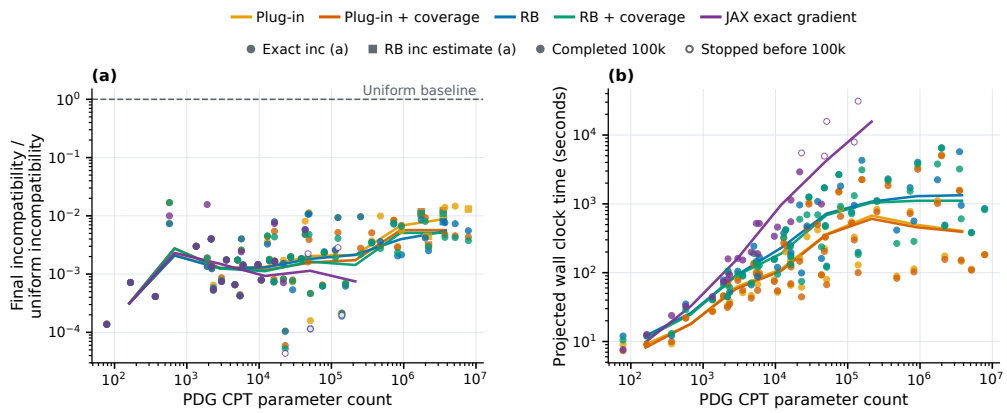


Figure 3: Noisy BIF PDGs versus PDG CPT entries. **(a)** Final incompatibility relative to the exact uniform-joint value (circles: exact evaluation; squares: 1.024×10^9-sample RB estimate). **(b)** Compilation-plus-training time for 100,000 updates; hollow points project early stops. Lines are geometric means over available runs. Stochastic methods use GPUs (51 cases each); JAX uses CPUs (26 scored cases).

5.2 PDGs from Empirically Perturbed Bayesian-Network Beliefs

Different studies or sensors can report overlapping conditional distributions from separate observations, with different sample sizes and conditioning settings. To model the task of synthesizing these local reports into a joint, we use 17 BIF benchmark BNs from the `bnlearn` repository (Scutari, n.d.) and construct PDGs with $2n$, $4n$, or $8n$ beliefs per network: 51 PDGs. Each belief initially equals an exact base-BN conditional. We then draw a belief-specific $N_e = \mathrm{round}(10 + 1000X_e)$, $X_e \sim \mathrm{Exp}(1)$, sample every CPT row independently from a multinomial of size N_e, and set $\beta_e = N_e/1000$. Rows of a belief share N_e; confidence therefore increases linearly with the sample count. Sampling is independent across reports, while their modeled variables may be dependent. These are synthetic observations from realistic BN structures, with the numerical floor described above. The resulting beliefs need not agree with any one joint distribution.

The candidate BN again uses the base graph and uniform initialization. Four stochastic methods use the settings above and complete all 51 cases; exact JAX runs on CPUs for at most one hour and has 26 scored cases. We divide final incompatibility by that of the uniform joint and plot against the number of scalar PDG CPT entries.

Every scored run improves on the uniform joint (Figure 3). Geometric-mean final-to-uniform ratios across 51 PDGs are 0.00214 (plug-in), 0.00185 (plug-in with coverage), 0.00187 (RB), and 0.00175 (RB with coverage): roughly 470–570-fold reductions. These large gains persist as exact optimization exceeds the computational budget. On the 23 PDGs where JAX produced no result within the resource limits, the stochastic methods still achieve roughly 300–400-fold reductions, compared with 780–890-fold on the 26 JAX-scored PDGs. Improvement therefore remains at the hundreds-fold scale beyond the reach of exact optimization under these limits. JAX reaches 0.00113 on its smaller cohort. Stochastic runtime rises less sharply with size, although hardware differs and hollow points are projections.

6 Conclusion

Optimizing a BN with sample-based score gradients makes PDG inconsistency approximation practical across larger, more complex models. Both controlled and noisy experiments demonstrate effective reconciliation of local beliefs. Exact attained incompatibility upper-bounds inconsistency, but practical estimates do not certify it: sampling bias, optimization error, and restrictive BN structure remain limitations. Adapting the candidate structure is an important next step.

AI USE STATEMENT

Generative AI tools assisted with implementation and debugging, experiment orchestration, literature search, mathematical exposition, drafting, editing, and figure preparation. The authors are responsible for the correctness and originality of the submitted work, including its claims, citations, code, and empirical results.

REPRODUCIBILITY STATEMENT

Section 4 specifies the candidate parameterization, gradient estimators, and coverage regularizer. The theoretical assumptions and proofs are given in Appendices B.4, C, D, and E. Section 5 describes both experimental datasets and the evaluated configurations; Appendix F provides dataset construction procedures, training and baseline settings, computational resources, and evaluation and timing conventions. The figure captions identify the evaluated cohorts, normalization choices, exact versus sampled scores, and projected runtimes. These descriptions distinguish the optimization procedures from their evaluation and document the comparisons reported in the paper.

REFERENCES

Barry C. Arnold and D. V. Gokhale. Distributions most nearly compatible with given families of conditional distributions. *TEST*, 7(2):377–390, 1998. doi: 10.1007/BF02565119. URL https://link.springer.com/article/10.1007/BF02565119.

John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997. doi: 10.1023/A:1007421730016. URL https://link.springer.com/article/10.1023/A:1007421730016.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL https://github.com/jax-ml/jax.

George Casella and Christian P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83 (1):81–94, 1996.

Shyh-Huei Chen, Edward H. Ip, and Yuchung J. Wang. Gibbs ensembles for nearly compatible and incompatible conditional models. *Computational Statistics & Data Analysis*, 55(4):1760–1769, 2011. doi: 10.1016/j.csda.2010.11.006. URL https://doi.org/10.1016/j.csda.2010.11.006.

Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL http://www.jstor.org/stable/2984875.

Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. *J. Artif. Int. Res.*, 50(1):71–104, May 2014. ISSN 1076-9757.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/c3395dd46c34fa7fd8d729d8cf88b7a8-Paper.pdf.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000. URL https://jmlr.org/papers/v1/heckerman00a.html.

Matthew Hoffman and David Blei. Stochastic structured variational inference. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 361–369. PMLR, 2015. URL https://proceedings.mlr.press/v38/hoffman15.html.

Jaime Shinsuke Ide, Fabio Gagliardi Cozman, and Fabio Tozeto Ramos. Generating random Bayesian networks with constraints on induced width. In *European Conference on Artificial Intelligence*, pp. 323–327, 2004. URL https://sites.poli.usp.br/p/fabio.cozman/Publications/Article/ide-cozman-ramos-ecai2004.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, Cambridge, MA, 2009.

Steffen L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19(2):191–201, 1995. doi: 10.1016/0167-9473(93)E0056-A. URL https://doi.org/10.1016/0167-9473(93)E0056-A.

Steffen L. Lauritzen. *Graphical Models*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press, Oxford, 1996. ISBN 978-0-19-852219-5.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon Mcauliffe. Rao-blackwellized stochastic gradients for discrete distributions. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4023–4031. PMLR, 2019. URL https://proceedings.mlr.press/v97/liu19c.html.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988. ISBN 978-1-55860-479-7.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pp. 814–822, 2014. URL https://proceedings.mlr.press/v33/ranganath14.html.

Oliver Richardson and Joseph Y Halpern. Probabilistic dependency graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):12174–12181, May 2021. doi: 10.1609/aaai.v35i13.17445. URL https://ojs.aaai.org/index.php/AAAI/article/view/17445.

Oliver E. Richardson. Loss as the inconsistency of a probabilistic dependency graph: Choose your model, not your loss function. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 2706–2735. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/richardson22b.html.

Oliver E. Richardson, Joseph Y. Halpern, and Christopher De Sa. Inference for probabilistic dependency graphs. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pp. 1741–1751. PMLR, 2023. URL https://proceedings.mlr.press/v216/richardson23a.html.

Oliver Ethan Richardson, Mandana Samiei, Mehran Shakerinava, Joseph D. Viviano, Abdessamad El Kabid, Ali Parviz, and Yoshua Bengio. Local inconsistency resolution: The interplay between attention and control in probabilistic models. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 4141–4149. PMLR, 2026. URL https://proceedings.mlr.press/v300/richardson26a.html.

Marco Scutari. bnlearn bayesian network repository. Online repository, n.d. URL https://www.bnlearn.com/bnrepository/. Accessed 2026-09-24.

Rohin Shah, Pedro Freire, Neel Alex, Rachel Freedman, Dmitrii Krasheninnikov, Lawrence Chan, Michael D Dennis, Pieter Abbeel, Anca Dragan, and Stuart Russell. Benefits of assistance over reward learning, 2021. URL https://openreview.net/forum?id=DFIoGDZejIB.

Bas van Opheusden, Luigi Acerbi, and Wei Ji Ma. Unbiased and efficient log-likelihood estimation with inverse binomial sampling. *PLOS Computational Biology*, 16(12):e1008483, 2020. doi: 10.1371/journal.pcbi.1008483.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

A EXTENDED RELATED WORK

Reconciliation of conditional beliefs. The compatibility of separately specified conditional distributions has a long history. For finite bivariate variables, Arnold & Gokhale (1998) studied incompatible conditionals and used a KL-based discrepancy to find a joint distribution that is closest to them. Dependency networks likewise admit cyclic, potentially inconsistent local conditionals; Heckerman et al. (2000) use a Gibbs-like sampling procedure to obtain a joint for inference. Gibbs ensembles offer another sampling-based reconciliation method (Chen et al., 2011). We build on this line of work by considering the overlapping and differently conditioned beliefs of a general probabilistic dependency graph (PDG). We fit a user-chosen Bayesian-network (BN) family directly to its incompatibility objective, targeting a minimum within that family.

PDG inference. Richardson & Halpern (2021) introduced PDGs and their incompatibility-based semantics, including the representation of beliefs that cannot all be satisfied by one joint distribution. Richardson (2022) showed how several familiar losses, including the variational evidence lower bound, arise as inconsistencies of particular PDGs. These papers establish the objective studied here. The discrete inference algorithms of Richardson et al. (2023) express the optimization over calibrated cluster marginals and solve a convex exponential-cone formulation (alongside a gradient-based baseline). Their compactness depends on the treewidth of the PDG's hypergraph. Our candidate BN encodes a chosen set of independencies, which may be much sparser than a cluster representation of that hypergraph. Restricting the search can lower computation while raising the minimum incompatibility; the restricted optimum is an upper bound on the unrestricted optimum, and optimization can still stop short of that restricted optimum. Local inconsistency resolution (Richardson et al., 2026) offers a complementary approach: it alternates attention to parts of a PDG with changes to selected model components. We optimize one candidate joint against the global incompatibility objective.

Variational optimization and BN learning. Choosing a tractable family of distributions is standard in variational inference, and structured families can retain dependencies omitted by mean-field approximations (Hoffman & Blei, 2015). Here "variational" describes the restriction of the *candidate joint* in a minimum-incompatibility problem; conventional variational inference instead fits an approximate posterior to observed data. BN parameter learning with incomplete observations supplies another useful comparison: expectation-maximization maximizes observed-data likelihood by averaging over missing values (Dempster et al., 1977; Lauritzen, 1995), while Binder et al. (1997) derive locally computable gradients for learning fixed-structure BNs with hidden variables. In contrast, each PDG term here averages a KL divergence from a candidate conditional to a supplied local belief over candidate source configurations. The divergence direction and candidate-weighted source configurations matter when beliefs conflict; a collection of supplied conditionals need not correspond to one data likelihood.

Sample-based gradients. The score-function identity underlying REINFORCE (Williams, 1992) and black-box variational inference (Ranganath et al., 2014) gives gradients from discrete samples without differentiating through the sampling operation. Control variates and Rao–Blackwellization are established ways to reduce variance, including for discrete stochastic gradients (Greensmith et al., 2004; Liu et al., 2019). Our specific difficulty is that a PDG loss at a sampled assignment contains conditionals of the candidate BN for *PDG* source and target sets, which need not be BN local conditional probability tables. Estimating those conditionals from the same forward samples used for the score gives a finite-sample biased plug-in gradient. Conditioning sampled target indicators on tractable BN neighborhoods can reduce the variance of the *conditional-probability estimates*. The classical Rao–Blackwell variance guarantee by itself does not imply that the complete gradient estimator has lower variance: the logarithm, random source counts, and reuse of samples in the score and conditional estimate all intervene. This distinction separates the proposed estimator analysis from standard applications in which the sampled loss can be evaluated exactly.

B GRAPH AND CLUSTER-TREE BACKGROUND

B.1 GRAPHS AND BAYESIAN NETWORKS

Definition 3. A *directed hypergraph* $(N, \mathcal{A})$ is a set of nodes N, and a set of *(hyper)arcs* $\mathcal{A}$, each $a \in \mathcal{A}$ of which is associated with a set of source nodes $S_a \subseteq N$, and target nodes $T_a \subseteq N$. We also write $S \xrightarrow{a} T \in \mathcal{A}$ to specify an arc a with sources $S_a = S$ and targets $T_a = T$. We often leave N implicit (or calculate it from $\mathcal{A}$ as $N = \cup_{a\in\mathcal{A}} S_a \cup T_a$), referring to the entire hypergraph as $\mathcal{A}$.

A *directed graph* can be viewed as the special case where every arc has precisely one source and target node. When describing dependencies, however, a directed graph $G = (N, A)$ is often a compact representation of a different directed hypergraph—namely, $\mathcal{A}_G = \{\text{Pa}_G(v) \to v\}_{v\in N}$, where $\text{Pa}_G(v) \doteq \{u \in N : (u \to v) \in A\}$ are the *parents* of node v. This is particularly common in dependency, causality, and graphical models, where the hyperarcs in $\mathcal{A}_G$ are the ones naturally annotated with functions and probability tables. We call $\mathcal{A}_G$ the hypergraph *generated by* the graph G. For an illustration, see Fig 4 (right).

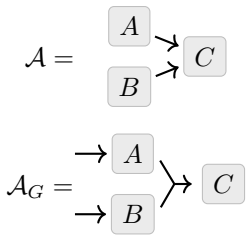


Figure 4: Top: a directed acyclic graph (dag) $\mathcal{A}$. Bottom: the alternate directed hypergraph $\mathcal{A}_G$ that the dag $\mathcal{A}$ often represents; its arcs encode joint dependence.

A *path* in a directed graph (N, A) is a sequence of nodes $X_1 \to \ldots \to X_n$ such that each $(X_i \to X_{i+1}) \in A$. We say $X \in N$ is a descendant of $Y \in N$ if there is a directed path from Y to X, and conversely that X is an ancestor of Y if there is a directed path from X to Y. We call a directed graph $(\mathcal{X}, \mathcal{A})$ *acyclic* (making it a 'dag') if for each arc $S \xrightarrow{a} T \in \mathcal{A}$, there does not exist a directed path from any target $Y \in T$ to any source $X \in S$. A hypergraph is *connected* if the undirected graph joining variables that share a hyperedge is connected. A *tree* is a connected undirected graph with no cycles.

Probabilistic Graphical Models. A *Bayesian network (BN)* $\mathcal{B} = (\mathcal{X}, \mathcal{A}, \mathbb{P})$ is a directed hypergraph $(\mathcal{X}, \mathcal{A})$ whose nodes $\mathcal{X}$ are variables and whose arcs $\mathcal{A} = \mathcal{A}_G$ are generated from a dag G, together with a conditional probability distribution $\mathbb{P}_a(T \mid S)$ for each edge $S \xrightarrow{a} T \in \mathcal{A}$. $\mathcal{B}$ induces a joint distribution over all variables, given by:

$$\mathbb{P}(\mathcal{X}) = \prod_{X\in\mathcal{X}} \mathbb{P}_X(X \mid \text{Pa}_G(X)) = \prod_{a\in\mathcal{A}_G} \mathbb{P}_a(T_a \mid S_a) \qquad \in \Delta\mathcal{V}\mathcal{X}.$$

Thus a BN encodes more than just the probabilistic data of its local cpds $\mathbb{P}$: it also encodes conditional independencies in its graphical structure $\mathcal{A}$ (Pearl, 1988).

B.2 INCOMPATIBILITY AND STRUCTURAL BELIEFS

In a PDG $\mathcal{M}$, each arc $S \xrightarrow{a} T \in \mathcal{A}$ represents an individual probabilistic belief or statistical relationship, while the qualitative arcs $\mathcal{A}_0$ represent beliefs about causal structure. These beliefs need not be consistent, *i.e.*, there need not exist any distribution $\mu \in \Delta\mathcal{V}\mathcal{X}$ which jointly satisfies each relationship. Richardson & Halpern (2021) propose a measure for quantifying the *degree* of inconsistency in a PDG. In order to define inconsistency, they first define the *incompatibility* of a PDG with any given candidate distribution $\mu \in \Delta\mathcal{V}\mathcal{X}$.

Incompatibility measures the extent to which a given candidate joint distribution μ fails to respect each probabilistic relationship encoded by the PDG, weighted by confidence[4]. The *inconsistency* of a PDG is then defined as the minimum achievable incompatibility over all candidate joint distributions.

We call a PDG $\mathcal{M}$ *consistent* if $\langle\!\langle \mathcal{M} \rangle\!\rangle = 0$, or equivalently, if there exists a joint distribution satisfying all of its positively weighted observational constraints. When $\langle\!\langle \mathcal{M} \rangle\!\rangle > 0$, the PDG is *inconsistent*,

[4]In addition to observational incompatibility, Richardson & Halpern (2021) define a second *structural* incompatibility, which measures the degree to which μ fails to respect various qualitative beliefs encoded in the PDG (e.g., independence relations, etc.). They then define incompatibility by $\text{Inc}^{\gamma}_{\mathcal{M}}(\mu) = \text{Inc}_{\mathcal{M}}(\mu) + \gamma\text{SInc}_{\mathcal{M}}(\mu)$, where $\gamma \geq 0$ encodes the relative confidence in the qualitative beliefs. In this paper, we handle only PDGs with qualitative beliefs representable with Bayesian Network structures, and $\gamma \in \{0, \infty\}$ so that conditional independencies are either strictly enforced or not assumed at all.

and so there is no distribution which jointly satisfies all of the encoded beliefs, and the value of $\langle\!\langle \mathcal{M} \rangle\!\rangle$ quantifies their intrinsic incompatibility. The minimizer $\mu^* \doteq \arg\min_{\mu \in \Delta\mathcal{VX}} \text{Inc}_{\mathcal{M}}(\mu)$ is also of special interest, as it represents the distribution which most closely satisfies the probabilistic relationships encoded in the PDG, and is a sort of naive reconciliation which "averages out" any inconsistencies in the beliefs.

B.3 Cluster-tree representations

In prior work on PDG inference, Richardson et al. (2023) argue that PDG inference and inconsistency estimation are extremely closely related, if not equivalent: inference in μ^* makes it easy to calculate $Inc_{\mathcal{M}}$. In the case where $\mathcal{A}_0 \subseteq \mathcal{A}$, answering queries about the incompatibility-minimizing distribution μ^* can be reduced to the problem of approximating the inconsistency of a PDG in near-linear time. They present two approaches to learning the optimal μ^* and the inconsistency $\langle\!\langle \mathcal{M} \rangle\!\rangle$: one formulating the problem as an exponential conic program and solving via convex optimization, and another directly applying gradient descent on the incompatibility objective. For both algorithms, instead of optimizing over the full joint distribution, they exploit a tree decomposition of the PDG's hypergraph to optimize instead over calibrated cluster marginals. Below, we define the tree decomposition of a hypergraph formally, following some prerequisite definitions.

Definition 4. Given a hypergraph $\mathcal{G} = (N, \mathcal{A})$, a *tree decomposition* of $\mathcal{G}$ is a tree $(\mathcal{C}, \mathcal{T})$ whose nodes $C \in \mathcal{C}$, called *clusters*, are subsets of N, satisfying:

1. for every hyperedge $S \xrightarrow{a} T \in \mathcal{A}$, there exists a cluster $C \in \mathcal{C}$ such that $S \cup T \subseteq C$; and
2. for every node $X \in N$, the set of clusters containing X forms a connected subtree of $(\mathcal{C}, \mathcal{T})$.

The *width* of a tree decomposition is $\max_{C \in \mathcal{C}} |C| - 1$, and the *treewidth* of $\mathcal{G}$ is the minimum width of any tree decomposition of $\mathcal{G}$. An elimination order removes variables successively, connecting each eliminated variable's remaining neighbors. Its *induced width* is the largest number of remaining neighbors at any elimination step; minimizing this over orders gives treewidth.

We can represent a factorized probabilistic distribution over a tree decomposition as follows.

Definition 5. Let $(\mathcal{C}, \mathcal{T})$ be a tree decomposition of a hypergraph $(\mathcal{X}, \mathcal{A})$ over variables $\mathcal{X}$. A *cluster tree distribution* over $(\mathcal{C}, \mathcal{T})$ is a collection $\boldsymbol{\mu}_{\mathcal{C}} \doteq \{\mu_C \in \Delta\mathcal{V}C\}_{C \in \mathcal{C}}$, assigning to each cluster $C \in \mathcal{C}$ a probability distribution over the variables in that cluster. $\boldsymbol{\mu}$ is said to be *calibrated* if, for every pair of neighboring clusters $(C, C') \in \mathcal{T}$, the two cluster marginals agree on their shared variables, *i.e.*, $\mu_C(C \cap C') = \mu_{C'}(C \cap C')$. In this case, $\boldsymbol{\mu}_{\mathcal{C}}$ determines a unique joint over $\mathcal{X}$ factorizing over this cluster tree:

$$\mu_{\mathcal{C}}(\mathcal{X}) = \frac{\prod_{C \in \mathcal{C}} \mu_C(C)}{\prod_{(C,C') \in \mathcal{T}} \mu_C(C \cap C')}.$$

Exact inference in standard graphical models, including belief propagation, can be viewed as the process of calibrating a cluster tree distribution (Koller & Friedman, 2009, Ch. 10). For learning inconsistency, it is sufficient to optimize over the relevant class of cluster trees instead of the full space of joint distributions.

Theorem 3 (Richardson et al. (2023))**.** *Let $\mathcal{M}$ be a PDG whose underlying hypergraph admits a tree decomposition $(\mathcal{C}, \mathcal{T})$, and let $\Delta_{(\mathcal{C},\mathcal{T})}(\mathcal{X})$ denote the set of cluster tree distributions over $(\mathcal{C}, \mathcal{T})$. Then*

$$\langle\!\langle \mathcal{M} \rangle\!\rangle = \min_{\mu \in \Delta_{(\mathcal{C},\mathcal{T})}(\mathcal{X})} \textit{Inc}_{\mathcal{M}}(\mu).$$

The primary limitation of a cluster tree distribution is that its size scales exponentially in the width of the decomposition. One response is to relax calibration constraints, as in loopy belief propagation; another is to impose additional independencies to obtain a tractable variational family. We take the latter approach for PDG inconsistency, using a Bayesian network to represent the candidate joint.

B.4 Cluster-equivalent logits

Root a junction tree of the PDG hypergraph. For a non-root cluster C, let S be its separator from its parent and let $Y = C \setminus S$. Use a BN with one root block C_0 and, for each non-root cluster, one block Y whose BN parents are S; joint block conditionals can equivalently be expanded into single-target

conditionals by the chain rule. Consider the unregularized incompatibility as a function $f(m)$ of positive, calibrated cluster marginals $m = (m_C)_C$. Each PDG term depends on a cluster containing its source and target, and has the form

$$\beta_e \sum_{s,t} m_{ST}(s,t) \log \frac{m_{ST}(s,t)}{m_S(s)\mathbb{P}_e(t \mid s)}.$$

For $\beta_e \geq 0$, this is convex in m_{ST}: it is the perspective of relative entropy, with the fixed belief distribution in its second argument. Thus f is convex on the convex polytope of calibrated cluster marginals.

Now fix any finite logit vector θ_0 and its positive calibrated marginals m^0. If it is not globally optimal, choose calibrated m^1 with $f(m^1) < f(m^0)$, allowing m^1 on the boundary. For $0 \leq u < 1$, the interpolation $m^u = (1-u)m^0 + um^1$ remains positive and calibrated, while convexity gives $f(m^u) \leq (1-u)f(m^0) + uf(m^1) < f(m^0)$ for every $u > 0$. The corresponding BN rows are

$$q^u_{C_0} = m^u_{C_0}, \qquad q^u_C(y \mid s) = \frac{m^u_C(s,y)}{m^u_S(s)} \quad (C \neq C_0).$$

They vary continuously with u near zero and reproduce m^u by the junction-tree factorization. Taking rowwise log probabilities as logits and adding the constant row offsets of θ_0 produces a continuous logit path beginning exactly at θ_0. Every arbitrarily small positive u lowers incompatibility. Hence no finite logit vector with suboptimal incompatibility can be a local minimum. This conclusion uses freely varying local CPT logits for a BN encoding the full cluster-tree family; it need not hold for an arbitrary interior-complete but nonregular parameter map or a sparser BN family.

C Parameterized families

Definition 6. A *parameterized Bayesian network* $\mathcal{B}_\Theta = (\mathcal{X}, \mathcal{A}, \boldsymbol{\mu}, \Theta)$ consists of a fixed hypergraph $(\mathcal{X}, \mathcal{A})$ generated by a dag, together with real parameter space $\Theta \subseteq \mathbb{R}^d$ and a collection of differentiable maps $\boldsymbol{\mu} = [\mu_a : \Theta \to \Delta\mathcal{V}(T_a \mid S_a)]_{a \in \mathcal{A}}$. Each parameter setting $\theta \in \Theta$ determines a BN with local cpds $\mathbb{P}_a(T \mid S) \doteq \mu_a(\theta) = \mu_{a,\theta}(T \mid S)$. We write μ_θ for its induced joint distribution.

A parameterization is *complete* if it represents every joint distribution that factorizes according to the BN structure, and *interior-complete* if it represents precisely only the strictly positive distributions (*i.e.*, having full support over $\mathcal{VX}$) with this factorization. When choosing the hypergraph $(\mathcal{X}, \mathcal{E})$ for our parameterized BN $\mathcal{B}_\Theta$, there are a few natural strategies. First, following Richardson et al. (2023), we could choose a structure based solely on the PDG hypergraph $(\mathcal{X}, \mathcal{A})$ whose variational family is guaranteed to contain an incompatibility minimizing distribution, independent of the particular probabilistic beliefs encoded. For example, we could simply take the cluster-tree decomposition of the PDG hypergraph used by Richardson et al. (2023) and construct an equivalent BN encoding the exact same independence structure (Lauritzen, 1996). This approach preserves exactness by design, so that $\langle\!\langle \mathcal{M} \rangle\!\rangle^\Theta = \langle\!\langle \mathcal{M} \rangle\!\rangle$, but inherits the weaknesses of the cluster-tree approach discussed earlier.

Second, we may use domain knowledge to impose independence relations that are not directly implied by the PDG hypergraph, but which we expect to hold in application. In this case, the variational gap measures precisely this additional inconsistency between the expert-assumed independence relations and the beliefs encoded in the PDG. Finally, following a more typical motivation for variational optimization, we may deliberately select a simpler model class, assuming away dependencies which are genuine, but are perhaps too weak to matter or too expensive to model tractably. This variational approximation provides a trade-off between computational efficiency and the tightness of the resulting upper bound on inconsistency.

D Gradient and estimator details

The derivation in Theorem 1 applies to every finite, differentiable interior parameterization of the candidate BN and every finite collection of strictly positive PDG beliefs with finite weights. In particular, no derivative of a PDG belief appears: its conditional distribution is fixed. The edge weights remain inside the pointwise statistic and therefore inside the score-function gradient.

D.1 SCORE-FUNCTION IDENTITY

Proof. Note that $\mathrm{Inc}_{\mathcal{M}}(\mu_\theta) = \mathbb{E}_{z\sim\mu_\theta}[l(z)]$. Writing $\nabla\mu_\theta(z) = \mu_\theta(z)\nabla\log\mu_\theta(z)$, the product rule gives

$$\nabla\mathrm{Inc}_{\mathcal{M}}(\mu_\theta) = \mathop{\mathbb{E}}_{z\sim\mu_\theta}\big[l(z)\,\nabla\log\mu_\theta(z)\big] + \mathop{\mathbb{E}}_{z\sim\mu_\theta}\big[\nabla l(z)\big].$$

It suffices then to show that the second term vanishes. This follows from the fact that

$$\nabla l(z) = \sum_{S\xrightarrow{a}T\in\mathcal{A}} \beta_a\big[\nabla\log\mu_\theta(z_S, z_T) - \nabla\log\mu_\theta(z_S)\big],$$

and for any set of variables $U \subseteq \mathcal{X}$, the expected score of the marginal on U is

$$\mathop{\mathbb{E}}_{z\sim\mu_\theta}\big[\nabla\log\mu_\theta(z_U)\big] = \sum_{u\in\mathcal{V}U}\mu_\theta(u)\,\nabla\log\mu_\theta(u) = \nabla\sum_{u\in\mathcal{V}U}\mu_\theta(u) = \nabla 1 = 0. \qquad \square$$

D.2 LOGIT GRADIENT AND PLUG-IN BIAS

Corollary 2. *If μ_θ is logit-parameterized, then for each edge $X \to Y \in \mathcal{E}$ and $(x, y) \in \mathcal{V}X \times \mathcal{V}Y$, we have that*

$$\nabla \mathit{Inc}_{\mathcal{M}}(\mu_\theta)|_{\theta_{y|x}} = \mathop{\mathbb{E}}_{z\sim\mu_\theta}\left[\mathbf{1}_{z_X=x}\cdot(\mathbf{1}_{z_Y=y} - \mu_\theta(y\mid x))\cdot l(z)\right]$$

Proof. Fix a BN edge $X \xrightarrow{b} Y \in \mathcal{E}$ and a logit $\theta_{y|x}$. The BN factorization and the softmax derivative give

$$\frac{\partial}{\partial\theta_{y|x}}\log\mu_\theta(z) = \mathbf{1}_{z_X=x}\left(\mathbf{1}_{z_Y=y} - \mu_{b,\theta}(y\mid x)\right).$$

Indeed, every local factor except the one for this BN edge is constant with respect to this logit, and differentiating the log-softmax of its row gives the displayed difference. Substitute this score into Theorem 1; the latter theorem already accounts for the parameter dependence of $l(z)$. $\square$

Lemma 1 (Bias of plug-in estimator)**.** *For every fixed θ, the plug-in gradient estimator satisfies*

$$\|\mathbb{E}[\widehat{g}_{\textit{plug-in}}(\theta)] - \nabla_\theta \mathit{Inc}_{\mathcal{M}}(\mu_\theta)\| \in O(N^{-1}),$$

where $\|\cdot\|$ denotes the Euclidean norm.

Proof. Fix θ. For $q \in \Delta\mathcal{V}\mathcal{X}$, define

$$F(q) := \sum_{z:q(z)>0} q(z)\,\nabla_\theta\log\mu_\theta(z)\sum_{S\xrightarrow{a}T\in\mathcal{A}}\beta_a\log\frac{q(z_T\mid z_S)}{\mathbb{P}_a(z_T\mid z_S)}.$$

The score $\nabla_\theta\log\mu_\theta(z)$ is held fixed as q varies. By the definition of the plug-in estimator and Theorem 1,

$$F(\hat{\mu}_\theta) = \widehat{g}_{\text{plug-in}}(\theta), \qquad F(\mu_\theta) = \nabla_\theta\mathrm{Inc}_{\mathcal{M}}(\mu_\theta).$$

First, F is bounded throughout the probability simplex. The scores are bounded over the finite state space, and for each edge,

$$\sum_{z:q(z)>0} q(z)\left|\log\frac{q(z_T\mid z_S)}{\mathbb{P}_a(z_T\mid z_S)}\right| \le \mathrm{H}_q(T\mid S) + \max_{s,t}\log\frac{1}{\mathbb{P}_a(t\mid s)}.$$

The right-hand side is uniformly bounded in q, since $\mathrm{H}_q(T\mid S) \le \log|\mathcal{V}T|$ and the PDG beliefs are strictly positive. Thus, finite edge weights imply that F is bounded, including when some entries of q are zero.

Since μ_θ is strictly positive, F is twice continuously differentiable near μ_θ. Taylor's theorem therefore gives

$$\|F(q) - F(\mu_\theta) - DF(\mu_\theta)[q-\mu_\theta]\| \le C\|q-\mu_\theta\|^2$$

for q sufficiently close to μ_θ, where DF denotes differentiation with respect to the probability table. Increasing C extends this bound to the entire simplex: outside that neighborhood, $\|q - \mu_\theta\|$ is bounded away from zero, while the left-hand side is bounded.

The empirical distribution satisfies

$$\mathbb{E}[\hat{\mu}_\theta] = \mu_\theta, \qquad \mathbb{E}\big[\|\hat{\mu}_\theta - \mu_\theta\|^2\big] = \frac{1}{N} \sum_{z \in \mathcal{VX}} \mu_\theta(z)\big(1 - \mu_\theta(z)\big) \le \frac{1}{N}.$$

Consequently, substituting $q = \hat{\mu}_\theta$ and taking expectations cancels the linear term in the Taylor expansion, giving

$$\|\mathbb{E}[\widehat{g}_{\text{plug-in}}(\theta)] - \nabla_\theta \text{Inc}_{\mathcal{M}}(\mu_\theta)\| \le C\, \mathbb{E}\big[\|\hat{\mu}_\theta - \mu_\theta\|^2\big] \le \frac{C}{N}. \qquad \square$$

The constant in Lemma 1 depends on the fixed parameter setting, PDG beliefs, weights, and finite state space; the result is not a uniform-in-parameter guarantee near the boundary of the BN family. Almost-sure consistency follows directly from the strong law for every empirical cell: all cells of μ_θ are positive, so the conditional ratios and their logarithms eventually converge to their population values.

The logarithm then introduces an additional statistical difficulty: no estimator based on a fixed finite sampling budget can estimate the log-probability of an event without bias for every event probability[5]. Our aim is therefore to construct computationally efficient gradient estimators with low bias and variance under a given sampling budget. We consider two approaches: a plug-in estimator based on empirical conditional frequencies, and a Rao–Blackwellization-based estimator that replaces sampled indicators with tractable conditional expectations.

The score baseline used in optimization. The implementation centers the sample payoffs by their batch mean and uses the finite-sample correction

$$\widehat{g}_{\text{centered}} = \frac{1}{N-1} \sum_{i=1}^{N} \big(L_i - \overline{L}\big) \nabla_\theta \log \mu_\theta(z^{(i)}), \qquad \overline{L} = \frac{1}{N} \sum_{i=1}^{N} L_i.$$

Here L_i is the estimated pointwise loss assigned to sample i. If $L_i = l(z^{(i)})$ is evaluated exactly, this estimator is unbiased: expanding the batch mean, cross-sample terms vanish because independent samples have mean-zero scores, and the $N - 1$ normalization cancels the remaining factor. With the empirical plug-in or RB loss, all L_i share the estimated conditional tables; the correction does not remove their plug-in bias. Its purpose is to reduce stochastic variation in optimization.

E Rao–Blackwellization of conditional estimates

We make precise the variance comparison used in the main text. Fix a PDG edge $S \xrightarrow{a} T$, a source assignment s with $\mu_\theta(s) > 0$, and a target assignment t. For one forward sample set $I_t = \mathbf{1}_{z_T = t}$ and $R_W = \mu_\theta(t \mid s, z_W)$, where the latter conditional is evaluated exactly. Conditional on $z_S = s$,

$$\mathbb{E}[R_W \mid z_S = s] = \mathbb{E}[I_t \mid z_S = s] = \mu_\theta(t \mid s), \qquad \text{Var}(R_W \mid z_S = s) \le \text{Var}(I_t \mid z_S = s).$$

Both statements follow from the tower property and the law of total variance. Conditional on the random number $\hat{c}(s) = m > 0$ of samples with source assignment s, their target contributions are independent and identically distributed under $\mu_\theta(T \mid s)$. Consequently the RB and indicator averages have the same conditional mean and respective conditional variances equal to their single-sample variances divided by m. If $W_1 \subseteq W_2$ (apart from variables already in S), conditioning on the coarser W_1 gives no greater variance than conditioning on W_2.

This comparison concerns the conditional-probability estimate, not the complete gradient $\widehat{g}_{\text{RB}}$. Taking logarithms of estimated ratios and multiplying by scores computed from the same samples does not

[5] Already for Bernoulli samples with unknown success probability p, the expectation of any statistic of N samples is a polynomial in p of degree at most N, and therefore cannot equal $\log p$ for all $p \in (0, 1)$. This restriction does not apply to estimators using a random, potentially unbounded number of samples; inverse binomial sampling provides one such unbiased estimator (van Opheusden et al., 2020).

preserve the variance ordering. The RB gradient is generally biased at finite N for the same reason as the plug-in gradient. At fixed θ, it is consistent: the strong law applies to the finitely many bounded conditional contributions, and every relevant source configuration has positive probability. Exact BN conditionals are strictly positive, so logarithms of the resulting ratios converge.

Lemma 2. *Choose $n+1$ distinct variables uniformly from a BN and choose one uniformly as target T, using the rest as sources S. The probability that S contains no descendant of T is at least $1/(n+1)$.*

Proof. Interpret the random edge as choosing $n+1$ distinct variables uniformly and then choosing its target uniformly among those $n+1$ variables, with the remaining variables as sources. Every finite DAG induced on the selected variables has at least one vertex with no descendant among the other selected variables: choose the last selected vertex in a topological ordering of the entire BN. Conditional on the selected set, each such vertex is the target with probability $1/(n+1)$. Hence the probability that the sources contain no descendant of the target is at least $1/(n+1)$. □

Multiple PDG targets. For a single-target BN, suppose the PDG target set T is an antichain in the BN DAG and S contains no descendant of any vertex in T. Let X_t be the BN parents of $t \in T$, and set $W = (\bigcup_{t\in T} X_t) \setminus S$. The local Markov property gives the tractable joint conditional

$$\mu_\theta(t \mid s, z_W) = \prod_{t\in T} \mu_\theta(t_t \mid z_{X_t}) .$$

To see this, condition on the displayed parent assignments and all source variables. No source is downstream of a target, and no target is downstream of another; successively applying the BN local Markov property to the target nodes yields the product. The right-hand side uses only local BN conditional entries.

For a BN with joint target blocks, the same idea has a useful sufficient condition. Let $Y_1, \dots, Y_k$ be the distinct BN target blocks intersecting T, with parent sets $X_1, \dots, X_k$. Suppose S is disjoint from these blocks and contains no descendant of any variable in them, and no directed path runs from one selected block to another. Put $W = (\bigcup_j X_j) \setminus S$. The BN factorization then yields

$$\mu_\theta(t \mid s, z_W) = \prod_{j=1}^{k} \sum_{y_j \in \mathcal{V}(Y_j \setminus T)} \mu_{j,\theta}\big(t_{Y_j \cap T}, y_j \mid z_{X_j}\big) .$$

The empty complement requires no summation. The stated condition is sufficient, not necessary: it ensures that none of the variables being conditioned on conveys information from a selected target block beyond its parents. This evaluation only marginalizes within local BN blocks. When the condition fails, the Markov-blanket conditional from the main text remains available.

F Experimental details

F.1 Width-grid generation details

Our first family comprises 3,275 Bayesian networks generated using the PMMixed Markov chain Monte Carlo algorithm of Ide et al. (2004). We target node counts $n \in \{10, \dots, 50\}$ and induced widths $w \in \{1, \dots, 18\}$, measured by minimum-weight elimination on the moralized BN graph. For each (n, w), we run a fresh chain with width bounded by w and at most six parents per node. After $50n^2$ warm-up steps, we retain the first graph attaining width w within 128,000 additional steps. We attempt widths in increasing order, stopping at the first unsuccessful search for each node count; this leaves some larger widths unrepresented for smaller networks (see Figure 5 in appendix). The resulting 655 structures are each instantiated with uniform variable cardinality $d \in \{2, 3, 4, 5, 6\}$ and independently sampled $\mathrm{Dirichlet}(\mathbf{1})$ CPT rows. For each BN, we construct a PDG with $2n$ distinct conditional beliefs $S \to T$: we sample T uniformly, draw the initial size of S from the BN's in-degree distribution, and select conditioning variables uniformly without replacement. We replace conditioning variables that are d-separated from T given the others by resampling them, removing them if no suitable replacement is found within five attempts. Each belief's target is the base BN conditional $P(T \mid S)$, computed exactly by variable elimination where feasible and otherwise estimated using Rao–Blackwellized counts from 4.096 billion samples. Finally, we independently draw confidences $\beta_e \sim \mathrm{Exp}(1) + 0.01$.

F.2 Training and evaluation configuration

Implementation details. We compute the plug-in and Rao–Blackwellized PDG conditionals from sampled source configurations, then aggregate their log ratios with the BN's local logit scores. Both estimators use a global baseline formed from the same sample batch: their sampled payoff is centered before the score projection, with an $N/(N-1)$ correction to its scale. The estimated conditionals still give finite-sample bias. Source masses in the coverage gradient are also estimated from the forward sample batch. Our GPU implementation uses JAX (Bradbury et al., 2018) and Pallas kernels for sampling, counting, and score aggregation; a stochastic update therefore requires no variable elimination. Exact-gradient comparisons instead evaluate BN marginals by variable elimination and differentiate the resulting incompatibility with JAX.

All five BN-parameterized methods use Adam (Kingma & Ba, 2015) with $(\beta_1, \beta_2) = (0.9, 0.999)$, constant learning rate 0.01, and $\epsilon = 10^{-12}$. Stochastic methods use a global baseline and $N = 128{,}000$ samples per update. Their coverage schedule is $\lambda_k = 0.03(\lambda_{\text{final}}/0.03)^{k/(K-1)}$, $k = 0, \ldots, K-1$, $K = 100{,}000$, with endpoints 10^{-5} for plug-in and 10^{-6} for RB, fixed from preliminary experiments. There is no learning-rate decay. Each configuration uses one optimization seed per PDG; the reported geometric means average across PDGs, rather than repeated training seeds. GPU assignments comprise NVIDIA RTX A4000, RTX A6000, and A100 devices, according to cluster availability; exact-gradient optimization uses CPUs. Jobs request at most 64 GiB of host memory.

Exact-gradient feasibility. For the width-grid comparison, we classify instances whose variable-elimination plan contains a factor with more than 2×10^{11} entries as beyond the resource budget. Materializing such a factor alone requires over 800 GB in float32, before autodifferentiation storage. This criterion concerns the chosen dense elimination plan, not every possible exact-inference implementation. Smaller plans can also exceed the memory or time budget. Missing exact-gradient scores do not enter averages or count as numerical incompatibility values.

Timing and scoring. Reported time includes compilation and training, excluding scheduling, setup, checkpoint I/O, and separate evaluations. The width-grid JAX stopping limit counts optimization time, so its reported total can exceed 25 minutes after compilation is included. The noisy-BIF JAX budget is one hour including setup, compilation, and training. For a run completing $k < 100{,}000$ iterations, projected time is $t_{\text{compile}} + 100{,}000\, t_{\text{train}}/k$. Exact-gradient curves include per-iteration objective computation in their training time. Exact final evaluations use float64 variable elimination when tractable; remaining final scores use 1.024×10^9-sample RB estimates. Metrics exclude coverage regularization. Final exact scoring renormalizes stored target rows in float64. JAX training uses stored targets and guards the source-mass denominator; these numerical conventions can differ slightly from final scoring even though both clip logarithm arguments at 10^{-10}. The width-grid trajectory panels compare the same PDGs across all five methods, normalize time by each PDG's RB-with-coverage compilation-plus-training time, and retain final values after termination. Approximate-method checkpoints are scored every $5{,}000$ iterations, and exact-gradient runs at every iteration.

Noisy-BIF construction. The 17 source networks are Alarm, Andes, Asia, Barley, Child, Diabetes, Hailfinder, Hepar2, Insurance, Link, Mildew, Munin, Munin1, Pathfinder, Pigs, Water, and Win95pts. Belief edges follow the target, source-size, and relevance-filter sampling procedure above, with fresh edges at each density, so the $2n$, $4n$, and $8n$ versions are not nested. Their unsampled targets are exact base-BN conditionals; finite multinomial sampling is applied separately to each row and no pseudocounts are added. The 10^{-10} logarithmic floor handles empirical zeros in optimization and scoring. Mean confidence before rounding is 1.01. Of 26 scored exact-gradient BIF runs, 21 complete $100{,}000$ iterations and five stop early. Across the five methods, 220 final scores are exact and ten use RB evaluation. Geometric-mean trend lines use common logarithmic bins with each method's available results; they are descriptive, not paired comparisons when cohorts differ.

F.3 Cluster-tree baselines

We also ran the cluster-tree methods of Richardson et al. (2023) on the 1,264 PDGs with scored exact-gradient runs. The convex method used the reference exponential-cone formulation with CVXPY's default solver (CLARABEL in the campaign); the gradient method used the reference cluster-tree parameterization and its native normalization and calibration penalties, with Adam at

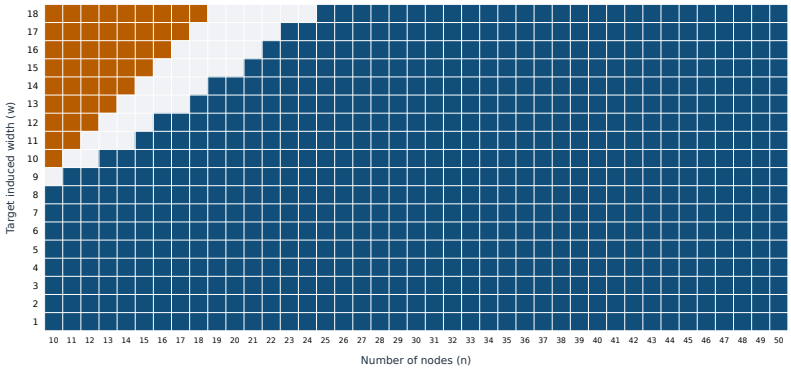


Figure 5: Coverage of our synthetically generated Bayesian networks with target induced widths 1–18. Blue cells contain a generated BN structure instantiated at each domain size $d \in \{2, 3, 4, 5, 6\}$; gray cells were not generated within the search budget. Orange cells are impossible because induced width cannot exceed $n - 1$.

learning rate 0.01 for at most $100{,}000$ iterations. Each attempt had a 25-minute algorithm deadline and ran in a four-CPU, 64-GiB SLURM worker allocation. The one-hour SLURM limit applied to the worker allocation, not to each algorithm attempt.

Recorded outcome	Convex	Cluster-tree Adam
Solver success	74	—
Time limit	421	315
Numerical failure	—	260
Out of memory	640	221
Other failure	129	468
Total attempts	1,264	1,264
Runs with at least one Adam iteration	—	345

Of the 74 convex successes, 37 had solver status `optimal` and 37 `optimal_inaccurate`. None of the cluster-tree Adam runs completed the full $100{,}000$ iterations. Its 260 recorded numerical failures arose after at least one iteration from a nonfinite native training objective. These are resource-limited outcomes, not values of incompatibility for failed attempts. The convex cluster-tree program minimizes over a family sufficient for the unrestricted PDG optimum, whereas our sampled and exact-gradient methods optimize within the chosen BN family; their attained objectives therefore answer different optimization questions when that BN family is restrictive.