Local Inconsistency Resolution: The Interplay Between Attention and Control in Probabilistic Models

Oliver E. Richardson [1,4,5,†] **Mandana Samiei** [2,5] **Mehran Shakerinava** [2,5]
Joseph D. Viviano [1,5] **Abdessamad El Kabid** [3,5] **Ali Parviz** [5] **Yoshua Bengio** [1,4,5]

[†]Corresponding author: `oliver.richardson@umontreal.ca`

[1]Université de Montréal [2]School of Computer Science, McGill University
[3]Institut Polytechnique de Paris [4]LawZero [5]Mila - Quebec Artificial Intelligence Institute

Abstract

We present a generic algorithm for learning and approximate inference with an intuitive epistemic interpretation: iteratively focus on a subset of the model and resolve inconsistencies using the parameters under control. This framework, which we call Local Inconsistency Resolution (LIR) is built upon Probabilistic Dependency Graphs (PDGs), which provide a flexible representational foundation capable of capturing inconsistent beliefs. We show how LIR unifies and generalizes a wide variety of important algorithms in the literature, including the Expectation-Maximization (EM) algorithm, belief propagation, adversarial training, GANs, and GFlowNets. In the last case, LIR actually suggests a more natural loss, which we demonstrate improves GFlowNet convergence. Each method can be recovered as a specific instance of LIR by choosing a procedure to direct focus (attention and control). We implement this algorithm for discrete PDGs and study its properties on synthetically generated PDGs, comparing its behavior to the global optimization semantics of the full PDG.

1 INTRODUCTION

What causes one to change their mind, to learn, act, and draw inferences? From a neurocognitive perspective, the mind operates by resolving internal inconsistencies, whether they arise from conflicting new information or from new awareness of contradictions among prior beliefs. Theories of cognitive dissonance (Festinger, 1962) famously explain much of human behavior as targeted adjustments to dispel the discomfort of psychological inconsistency. Predictive coding theories (Rao and Ballard, 1999; Friston, 2005) posit that discrepancies between expected and actual sensory inputs—known as prediction errors—are treated as inconsistency signals that drive learning via local adjustments to neural representation. Inspired by the deeper underlying neurocognitive principle, we introduce a computational framework that formalizes the process of *local inconsistency resolution* (LIR) in probabilistic models.

The locality—which manifests in two ways—is essential for those of us with computational limitations. First, inconsistency can be difficult to detect, and impossible to address constructively unless it falls within our field of *attention*. Second, we seldom have uniform *control* over our epistemic state; we are generally reluctant to revise our direct observations or bedrock principles, forcing us instead to adjust other more pliable beliefs. So in practice, we resolve inconsistencies *locally*—by looking at only a small part of the model, and modifying only a fraction of its parameters. This process isn't guaranteed to succeed: some inconsistencies simply cannot be seen or addressed with a narrow focus, and fixing one inconsistency can easily create others out of view. Nevertheless, we will show that LIR provides a powerful recipe for learning and (approximate) inference and demonstrate that many foundational techniques in the literature arise naturally as instances of it.

Our approach leans heavily on the theory of Probabilistic Dependency Graphs (PDGs), which are highly flexible graphical models capable of representing arbitrary (even inconsistent) probabilistic information, weighted by confidence (Richardson and Halpern, 2021). There is a natural way to measure how inconsistent a PDG is, and many standard loss functions can be

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

viewed as measuring the inconsistency of a PDG that describes the appropriate situation (Richardson, 2022). LIR operationalizes training models as the process of adjusting parameters to resolve this inconsistency.

Computing a PDG's degree of inconsistency is easy in some cases, but it is often intractable and is NP-hard in general (*e.g.*, the log number of solutions to a SAT problem and the log evidence of a latent variable model can both be represented as PDG inconsistencies). The problem is equivalent to inference in PDGs, which, like for other graphical models, can be achieved in polynomial time under the assumption of bounded tree-width (Richardson et al., 2023). Variational inference (Blei et al., 2017; Kingma and Welling, 2014; Jordan et al., 1999) can be understood as the practice of adopting extra beliefs to get an overapproximation of inconsistency that is easier to calculate (Richardson, 2022). Our approach also enables the opposite: focusing on small parts of the graph at a time to address tractable underapproximations of the global inconsistency.

2 PRELIMINARIES AND PPDGS

Variables and Probability. We write $\mathcal{V}X$ for the set of values that a (random) variable X can take on, and $\Delta\mathcal{V}X$ for the set of distributions over $\mathcal{V}X$. A conditional probability distribution (cpd) is a map $p(Y|X) : \mathcal{V}X \to \Delta\mathcal{V}Y$. If $\mathcal{X} = \{X_1, X_2, \ldots\}_{i\in I}$ is an (indexed) set of variables, we regard $\mathcal{X}$ itself as a variable that can take on joint settings $\mathcal{V}\mathcal{X} = \prod_{i\in I} \mathcal{V}X_i$. If $\mu \in \Delta\mathcal{V}\mathcal{X}$ is a joint distribution and $Y \subseteq \mathcal{X}$, we write $\mu(Y)$ for the marginal of μ on the variables Y.

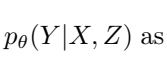
A *directed hypergraph* $(N, \mathcal{A})$ is a set of nodes N and a set of arcs $\mathcal{A}$, each $a \in \mathcal{A}$ of which is associated with a set $S_a \subseteq N$ of source nodes, and $T_a \subseteq N$ target nodes. We also write $S \xrightarrow{a} T \in \mathcal{A}$ to specify an arc a together with its sources $S = S_a$ and targets $T = T_a$.

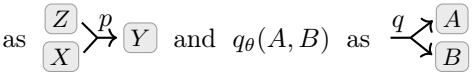
Geometry. For our purposes, a *pointed parameter space* Θ is a convex subset of $\mathbb{R}^n$ for some $n \geq 0$ (that may differ between parameter spaces) with a distinguished default value $\theta_0 \in \Theta$. A *vector field* over Θ is a differentiable map X assigning to each $\theta \in \Theta$ a vector $X_\theta \in \mathbb{R}^n$. The *gradient* of a twice differentiable map $f : \Theta \to \mathbb{R}$, which we write $\nabla_\Theta f(\Theta)$, is a vector field. Given a vector field X and an initial point $\theta_0 \in \Theta$, there is a unique trajectory $y(t)$ that solves the ordinary differential equation (ODE) $\{\frac{\mathrm{d}}{\mathrm{d}t} y(t) = X_{y(t)}, y(0) = \theta_0\}$. To refer to that solution compactly, we adopt the notation $\exp_{\theta_0}(X) := y(1)$. Although exp may appear to give us access only to $y(1)$, it is easily verified that $\exp_{\theta_0}(tX) = y(t)$ for all $t \geq 0$. Putting all the pieces together: the map $t \mapsto \exp_\theta(t\nabla_\Theta f(\Theta))$ is the smooth path beginning at θ that follows the gradient of f. It is known as *gradient flow*.

Probabilistic Dependency Graphs. A PDG is a directed (hyper)graph whose arcs carry probabilistic and causal information, weighted by confidence (Richardson and Halpern, 2021). We define an unweighted (but for our purposes, equivalent) variant whose explicit parametric nature will prove useful.

Definition 1. An *unweighted parametric PDG* $\mathcal{M}(\Theta) = (\mathcal{X}, \mathcal{A}, \Theta{=}\{\Theta_a\}_{a\in\mathcal{A}}, \mathbb{P}{=}\{\mathbb{P}_a\}_{a\in\mathcal{A}})$ is a directed hypergraph $(\mathcal{X}, \mathcal{A})$ whose nodes $\mathcal{X}$ are variables, and whose arcs $a \in \mathcal{A}$ are each associated with a parameter space Θ_a and a map $\mathbb{P}_a : \Theta_a \times \mathcal{V}S_a \to \Delta\mathcal{V}T_a$ that gives a cpd $\mathbb{P}_a(T_a|S_a;\theta)$ over a's target variables given values of its sources and a parameter setting $\theta \in \Theta_a$.

An unweighted PDG is the object obtained by fixing the parameters: a joint setting $\boldsymbol{\theta} = (\theta_a)_{a\in\mathcal{A}}$ yields a concrete unweighted PDG $\mathcal{M} = \mathcal{M}(\boldsymbol{\theta})$—which, at a syntactic level, is just a collection of cpds. When constructing PDGs from known cpds (*e.g.*, $p(Y|X)$), we use the mathematical symbol representing that cpd (*e.g.*, p) for both the arc a and its associated cpd $\mathbb{P}_a$. We typically depict (parametric) PDGs in graphical notation, specifying a cpd

$p_\theta(Y|X,Z)$ as 

and $q_\theta(A,B)$ as 

.

A PDG can be faithfully viewed as the special case of a parametric PDG in which each parameter space $\Theta_a = \{\theta_0\}$ is a singleton containing only the default value. Conversely, a parametric PDG may be viewed as a PDG by adding each Θ_a as an explicit variable.

PDG Semantics and Inconsistency. The power of PDGs comes from their semantics, which sew their (possibly inconsistent) cpds and confidences together to describe features of a joint distribution $\mu(\mathcal{X})$.

A PDG contains two kinds of information: qualitative structure, encoding the types of causal mechanisms (the hypergraph $\mathcal{A}$), and observational data, in the form of specific probabilistic beliefs (the cpds $\mathbb{P}$). In the standard presentation (Richardson, 2024), a PDG also comes with weight vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \overline{\mathbb{R}}^{\mathcal{A}}$ over the arcs, encoding confidence in information of each type: α_a can be thought of as the number of independent causal mechanisms by which S_a determines T_a (Richardson et al., 2024), while β_a can be thought of as the effective number of independent reports endorsing that the probability of T_a given S_a is $\mathbb{P}_a$. Corresponding to these two types of information, PDG semantics provide a way of scoring compatibility of a joint distribution $\mu(\mathcal{X})$ with information of each type.

With respect to a PDG $\mathcal{M}$, the *observational incompatibility* of a joint probability measure $\mu \in \Delta\mathcal{V}\mathcal{X}$ is given by a weighted sum of relative entropy (KL) terms

$$OInc_{\mathcal{M}}(\mu) := \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \, \boldsymbol{D}\Big(\mu(T,S) \;\Big\|\; \mathbb{P}_a(T|S)\mu(S)\Big), \quad (1)$$

where $\boldsymbol{D}(p \parallel q) = \mathbb{E}_{x\sim p}[\log \frac{p(x)}{q(x)}]$. This quantity can be thought of as the excess cost of using codes optimized for our beliefs $\mathbb{P}$ weighted by the confidence $\boldsymbol{\beta}$ we have in them, when in fact $\mathcal{X} \sim \mu$. If $\boldsymbol{\beta} > \mathbf{0}$, then $\mathit{OInc}_{\mathcal{M}}(\mu) = 0$ if and only if μ satisfies the constraints imposed by every cpd of $\mathbb{P}$. Intuitively, the cpds of $\mathbb{P}$ are inconsistent when constraints are not simultaneously satisfiable, in which case the *observational inconsistency* $\langle\!\langle \mathcal{M} \rangle\!\rangle_0 := \inf_\mu \mathit{OInc}_{\mathcal{M}}(\mu)$ is an important measure of the magnitude of the unavoidable internal conflict between the probabilistic data in $\mathcal{M}$.

Modern machine learning usually prizes observational data above all else—in particular, above structural information such as causal influence or qualitative independencies. For this reason, the scoring function *OInc* and the corresponding observational inconsistency $\langle\!\langle \mathcal{M} \rangle\!\rangle_0$ will suffice for most of our examples. That said, we can also score μ by its incompatibility with the structural information in the weighted hypergraph $(\mathcal{A}, \boldsymbol{\alpha})$. This *structural deficiency* (*SDef*) is given by:[1]

$$\mathit{SDef}_{\mathcal{A},\boldsymbol{\alpha}}(\mu) := -\,\mathrm{H}(\mathcal{X}) + \sum_{a\in\mathcal{A}} \alpha_a\, \mathrm{H}(T_a \mid S_a), \quad (2)$$

and, roughly speaking, measures μ's failure to arise as a result of an independent causal mechanism along each hyperarc (Richardson et al., 2024). If $\mathcal{A}$ is a qualitative Bayesian Network and $\boldsymbol{\alpha} = \mathbf{1}$, for instance, then $\mathit{SDef}_{\mathcal{A},\boldsymbol{\alpha}}(\mu) \geq 0$ with equality iff μ has the conditional independencies of $\mathcal{A}$. With confidence $\gamma \geq 0$ in the structural information overall, the γ-*inconsistency* of $\mathcal{M}$ is the smallest possible overall incompatibility of any distribution with $\mathcal{M}$, and denoted

$$\langle\!\langle \mathcal{M} \rangle\!\rangle_\gamma := \inf_\mu \Big(\mathit{OInc}_{\mathcal{M}}(\mu) + \gamma\, \mathit{SDef}_{(\mathcal{A},\boldsymbol{\alpha})}(\mu) \Big). \quad (3)$$

Richardson (2022) argues that this inconsistency measure (3) is a "universal" loss function, largely by showing how it specializes to standard loss functions across a wide breadth of contexts. It follows that, at least at an abstract level, much of machine learning can be viewed as inconsistency resolution. We now take this idea a few steps further, by operationalizing the resolution process and allowing it to be done "locally".

3 THE LOCAL INCONSISTENCY RESOLUTION (LIR) ALGORITHM

There are two distinct senses in which inconsistency resolution can be *local*: we can restrict what we can see, or what we can do about it. Correspondingly, there are two "focus" knobs for our algorithm: one restricts our **attention** to the inconsistency of a subset of arcs $A \subseteq \mathcal{A}$, and the other restricts our **control** to only the parameters of a subset $C \subseteq \mathcal{A}$ of arcs as we resolve that inconsistency. The former makes for an underestimate of the inconsistency that is easier to calculate, while the latter makes for an easier optimization problem. These restrictions are not just cheap approximations, though: they are also appropriate modeling assumptions for actors that cannot see and control everything at once.

Attention and control need not be black and white. A more general approach would be to choose an *attention mask* $\varphi \in \overline{\mathbb{R}}^{\mathcal{A}}$ and a *control mask* $\chi \in [0,\infty]^{\mathcal{A}}$. Large $\varphi(a)$ makes a salient while $\varphi(a) = 0$ keeps it out of mind; similarly, large $\chi(a)$ gives significant freedom to change a's parameters, small $\chi(a)$ affords only minor adjustments, and $\chi(a) = 0$ prevents change altogether. We often say "full control" to mean $\chi(a) = \infty$, in which case we move θ_a to a fixed point.

In full generality, we refine the types of φ and χ one step further by breaking the control and attention in each arcs into natural subcomponents. The behavior of the attention mask φ described in the previous paragraph matches the role played by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in PDG semantics—which is why we needed only to define unweighted PDGs as our basic objects, but require weighted semantics for them. The parameter γ also describes a form of attention, to the overall qualitative information in the network. Thus, we can reinterpret all three classical weight parameters as components of the attention mask $\varphi = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$, a vector of dimension $2|\mathcal{A}| + 1$. To simplify notation, we write $\langle\!\langle \varphi \odot \mathcal{M} \rangle\!\rangle := \langle\!\langle \mathcal{M}, \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle\!\rangle_\gamma$ to denote the γ-inconsistency of the weighted PDG $(\mathcal{M}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. We will typically set $\gamma = 0$, rendering $\boldsymbol{\alpha}$ irrelevant (except in Section 4.4 where $\gamma = 1$ and $\boldsymbol{\alpha} = \boldsymbol{\beta}$), and also assume $\boldsymbol{\beta} = \boldsymbol{\alpha} = \mathbf{1}$ unless otherwise specified. We also allow $\chi(a)$ to be a vector of dimension $\dim \Theta_a$, so overall χ is a vector of dimension $\sum_{a\in\mathcal{A}} \dim \Theta_a$.

Algorithm 1 Local Inconsistency Resolution (LIR)

Require: procedure Refocus
Input: knowledge base $\mathcal{M}(\Theta)$,
Initialize $\boldsymbol{\theta}^{(0)} \leftarrow 0$;
for $t = 0, 1, 2, \ldots$ **do**
  $\varphi, \chi \leftarrow$ Refocus();
  $\boldsymbol{\theta}^{(t+1)} \leftarrow \exp_{\boldsymbol{\theta}^{(t)}} \Big\{ -\chi \odot \nabla_{\boldsymbol{\theta}} \Big\langle\!\Big\langle \varphi \odot \mathcal{M}(\boldsymbol{\theta}) \Big\rangle\!\Big\rangle \Big\}$;

Formalized in Algorithm 1, LIR is a heuristic algorithm that adjusts the parameters of a parametric PDG $\mathcal{M}(\Theta)$ as to locally resolve inconsistencies as follows. Start with a belief state, in the form of a parametric PDG

[1] If the underlying variables are continuous, then we redefine entropy as follows. Assume each variable comes with a base measure λ, like the Lebesgue or counting measure. Then define $\mathrm{H}_\mu(X) := \mathbb{E}_\mu[\log \frac{\mathrm{d}\mu(X)}{\mathrm{d}\lambda_X}]$, where $\frac{\mathrm{d}\mu}{\mathrm{d}\lambda}$ is the Radon-Nikodym derivative of $\mu(X)$ with respect to λ_X.

$\mathcal{M}(\Theta)$, and initialize the parameters to their default values. In each iteration, choose *focus* consisting of an attention mask $\varphi = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ and a control mask χ. Calculate $\langle\!\langle \varphi \odot \mathcal{M}(\Theta) \rangle\!\rangle_\gamma$, the inconsistency of the combined context and memory, weighted by attention. In many cases (*e.g.*, when $\boldsymbol{\beta} \geq \gamma\boldsymbol{\alpha}$) these problems can be solved with techniques in conic optimization (Richardson et al., 2023), but this may be intractable unless the attention is narrow or one can find a formula for it in closed form. Finally, resolve this local inconsistency by updating mutable memory via (an approximation to) gradient flow, changing the parameters associated with a in proportion to the degree of control $\chi(a)$.

The ODE described in the final line may be approximated with an inner loop running an iterative gradient-based optimization algorithm. Alternatively, if Refocus produces small χ, then it is well-approximated by a single gradient descent step of size χ. At the other extreme, if χ is infinite in every component, then we typically expect the final line to reduce to

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{\theta}} \langle\!\langle \varphi \odot \mathcal{M}(\boldsymbol{\theta}) \rangle\!\rangle\,, \tag{4}$$

at least in many cases of interest. For example:

Proposition 1. *If $\mathcal{M}(\Theta)$ is an unweighted parametric PDG whose parameterizations $\mathbb{P}$ are either constant or unconditional and log-concave and $\boldsymbol{\beta} \geq \gamma\boldsymbol{\alpha}$, the map $\boldsymbol{\theta} \mapsto \langle\!\langle \mathcal{M}(\boldsymbol{\theta}), \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle\!\rangle_\gamma$ is convex.*[2]

Surprisingly, inconsistency is not always convex in the parameters of *conditional* probability distributions—yet we conjecture that (4) holds nevertheless. Determining whether or not $\langle\!\langle \mathcal{M} + p(Y|X) \rangle\!\rangle$ is quasi-convex in p remains a key open question in the theory of PDGs.

In order to run Algorithm 1, we must select the procedure Refocus that provides attention and control masks. We focus primarily on the case where Refocus chooses non-deterministically from a fixed finite set of attention-control mask pairs $\mathbf{F} = \{(\varphi_i, \chi_i)\}_{i=1}^n$, which we call *foci*. Even so, Algorithm 1 is quite abstract. To give intuition for it, next we give a sample of important algorithms that are instances of LIR.

4 UNIFYING ALGORITHMS AS LIR INSTANCES

4.1 The Classification Setting

Consider a parametric classifier $p_\theta(Y|X)$, perhaps arising from a neural network whose final layer is a softmax, and suppose we also have a labeled example (x, y). Let's capture this situation with a parametric PDG. Since $\mathcal{V}Y$ is a discrete label space space, we regard

[2]All proofs can be found in the appendix.

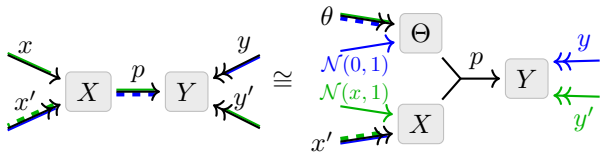
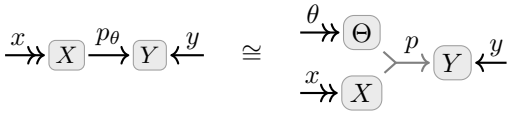


Figure 1: Illustrations of adversarial training as LIR. Foci are in green and blue; dashes indicate control. Right: explicit parameter variable Θ and exact symmetry.

labels as unconditional distributions over Y, viewing "hard" labels such as our $y \in \mathcal{V}Y$ as vertices of this simplex, while also allowing for label smoothing (see Müller et al., 2019, for an overview). Doing the same for x may be intractable, but fortunately is not necessary; in the typical case where the space of inputs $\mathcal{V}X$ is itself a manifold (*e.g.*, color images in $[0,1]^{W \times H \times 3}$), we can regard a value $x \in \mathcal{V}X$ as parameterizing a deterministic unconditional probability over X. To communicate this deterministic parameterization visually, we use a double-headed arrow, writing $\text{-}x \twoheadrightarrow \boxed{X}$. Combining the parametric classifier $p_\theta(Y|X)$ with the sample (x, y), we get a parametric PDG $\mathcal{M}(x, y, \theta) :=$



(5)

whose observational inconsistency is $-\log p_\theta(y|x)$, the standard training objective for such a classifier (Richardson, 2022). Each cpd plays major role in this inconsistency. What happens when we resolve this it with control over different arcs?

- Adjusting θ amounts to **training** the network in the standard way. In this case, the value χ of the control mask corresponds roughly to the product of the learning rate and the number of optimization iterations spent on the example (x, y).
- Adjusting y amounts to **inference**: it adjusts y to match distribution $p_\theta(Y|x)$.
- Adjusting x amounts to **forming an adversarial example**: it makes small changes to the input until the (fixed) network gives it the desired label.

Some readers may find the last point surprising. While adversarial examples (Goodfellow et al., 2014) are often presented as weird quirk of neural networks, they are, together with training and inference, one of the three basic resolutions ways of resolving this simple inconsistency. The sustained interest of the machine learning community on adversarial examples may appear to be a cultural phenomenon, but at a mathematical level, it is no accident (Shafahi et al., 2018). At this level of abstraction, there is no difference between the network parameters and inputs. Making the parameterization of p explicit and adding L2 regularization, the symmetry

becomes striking (see Figure 1, right).

With minor modifications to $\mathcal{M}(x, y, \theta)$, we can capture variations of training procedures.

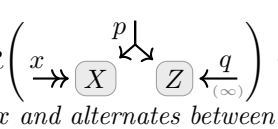
Stochastic Gradient Descent (SGD). Take the mutable state to be the classifier p as before. Define Refocus so that it draws a batch of samples $\{(x_i, y_i)\}_{i=1}^m$, places infinite attention on their empirical distribution $d(X, Y)$ (reflecting high confidence in the data). If $\eta := \chi(p)\varphi(p)$ is small, then LIR reduces to SGD with batch size m and learning rate η.

Adversarial training. Suppose we want to slightly alter x to obtain x' that is classified as y' instead of y. Adding x' and y' to $\mathcal{M}$ and relaxing $\mathbb{P}_x$ to be a Gaussian centered x rather than a point mass, we get the PPDG on the left of Figure 1. A LIR iteration whose focus consists of the edges marked in green (with control over the dashed green edge) is an adversarial attack with Euclidean distance (Biggio et al., 2013). The blue focus, by contrast, "patches" the adversarial example by adjusting the model parameters to again classify it correctly. Thus, LIR that alternates between the two and refreshes (x, y, y') is adversarial training, a standard defense to adversarial attacks (Goodfellow et al., 2014). Thus, it is just as sensible to train the inputs, as the network (Kishore et al., 2021).

4.2 The EM Algorithm and VAEs

Consider a latent variable model $p_\theta(Z, X)$. Given observations of the form $X{=}x$, how can we learn model parameters θ despite the missing data z? The standard answer is the EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008), which iteratively improves an estimate of θ by first taking an **E**xpectation over the missing values z (fixing the current estimate θ), and then performing **M**aximization over θ to update the parameters; altogether this amounts to computing:

$$\theta_{\mathrm{EM}}^{(t+1)} := \arg\max_{\theta} \mathbb{E}_{z \sim p(Z|x, \theta_{\mathrm{EM}}^{(t)})} \left[\log p(x, z|\theta) \right].$$

Proposition 2. *LIR*  *in which* Refocus *re-samples x and alternates between full control of p and q implements EM, in that $\theta_{EM}^{(t)} = \theta_{LIR}^{(2t)}$.*[3]

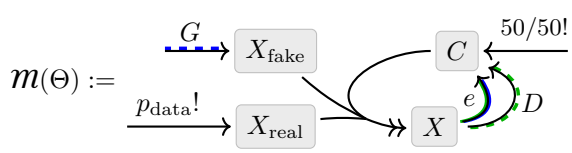
This result is closely related to one due to Neal and Hinton (1998), who view it as an intuitive explanation of why the EM algorithm works. Indeed, it is obvious in this form that every adjustment reduces the overall inconsistency, and that the M step need not fully solve the maximization problem, but merely make progress.

[3]The result can also be readily adapted to an entire dataset by replacing x with a high confidence empirical distribution, or mini-batched as in Section 4.1.

Variational inference as LIR. This form of the EM algorithm is closely related to variational inference. Indeed, the inconsistency of the PDG in Proposition 2 is equal to the evidence lower bound (ELBO), the optimization target for variational methods. Furthermore, allowing q to depend on x and breaking up $p(X, Z) = p(Z)d(X|Z)$ into two separate arcs yields a PDG containing the data of a variational autoencoder, or VAE (Kingma and Welling, 2014), and the resulting inconsistency is the variant of the ELBO used to train VAEs (Richardson, 2022). It follows that LIR where we control the encoder and decoder (but not the prior) and attend to various mini-batches of samples, trains a VAE with stochastic gradient descent. *Mean-field* variational methods are ones where the latents $Z = (Z_i)_{i=1}^n$ are real-valued, and the variational family is restricted to those of the form $q(Z) = \prod_{i=1}^n q_i(Z)$.

4.3 Generative Adversarial Networks

LIR can also capture the procedure for training Generative Adversarial Networks, or GANs (Goodfellow et al., 2020). The goal is to train a generator network G to generate images that cannot be distinguished from real ones. Formally, let X be either a generated image $X_{\text{fake}} \sim G$ or one from a dataset $X_{\text{real}} \sim p_{\text{data}}$, based on the outcome of a fair coin C. A discriminator D then predicts the outcome C of the coin flip from the image X. The only additional ingredient we will need that is not explicit in the original formulation is a belief e that the coin is equally likely heads as tails given X—intuitively, representing the goal of the generator and the opposite of the goal of the discriminator. All of these pieces of probabilistic information can be summarized by the (foci-annotated) PDG below.



GAN training is often written as a 2-player game $\min_G \max_D \mathcal{L}^{\text{GAN}}(G, D)$, where $\mathcal{L}^{\text{GAN}}(G, D) =$

$$\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x' \sim G}[\log(1 - D(x'))].$$

The Discriminator's Focus (φ_D, χ_D)**.** The discriminator has control over D, and attends to everything but e. That inconsistency of this PDG is what might be called the discriminator's objective: the expected KL divergence from D to the optimal discriminator. If D also disbelieves that any image is equally likely to be fake as real (by choosing to give negative attention, $\varphi_D(e) = -1$), then the inconsistency becomes $-\mathcal{L}^{\text{GAN}}$.

The Generator's Focus (φ_G, χ_G)**.** The generator has control over G. If it ignores D and attends only to

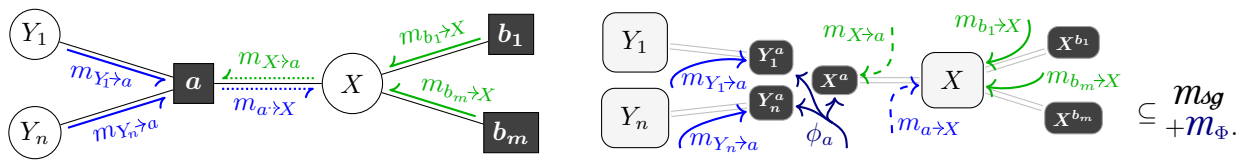


Figure 2: Left: an illustration of the local message-passing updates (6,7). Right: the PDG $\mathcal{M}_{\mathcal{G}}$ of Section 4.4.

e, the inconsistency is the Jensen-Shannon Divergence between G and p_{data}. If the generator also disbelieves the discriminator D (*i.e.*, $\varphi_G(D) = -1$), then the inconsistency becomes $+\mathcal{L}^{\text{GAN}}$. It follows that:

Proposition 3. *LIR($\mathcal{M}$) with foci alternating between (φ_G, χ_G) and (φ_D, χ_D) trains a GAN.*

4.4 Message Passing

A *factor graph* over a set of variables $\mathcal{X}$ is a set of factors $\Phi = \{\phi_a : \mathbf{X}_a \to \mathbb{R}_{\geq 0}\}_{a \in \mathcal{A}}$, where each $\mathbf{X}_a \subseteq \mathcal{X}$ is called the *scope* of a. Conversely, for $X \in \mathcal{X}$, let $\partial X := \{a \in \mathcal{A} : X \in \mathbf{X}_a\}$ be the set of factors with X in scope. The factor graph Φ specifies a distribution $\Pr_\Phi(\mathcal{X}) \propto \prod_a \phi_a(\mathbf{X}_a)$, and corresponds to a PDG $\mathcal{M}_\Phi = \left\{ \xrightarrow[(\alpha,\, \beta=1)]{\propto \phi_a} \boxed{\mathbf{X}_a} \right\}_{a \in \mathcal{A}}$ that specifies the same distribution $\Pr_\Phi(\mathcal{X})$ when γ=1.

Sum-product belief propagation (BP) (Kschischang et al., 2001) aims to approximate marginals of $\Pr_\Phi$ with local computations: interactions between neighboring factors and variables. The state consists "messages" $m_{X \to a}$ and $m_{a \to X}$ both (unnormalized) distributions over X, for each variable X and factor $a \in \partial X$ adjacent to it. After initialization, BP repeatedly recomputes:

$$m_{X \to a}(x) :\propto \prod_{b \in \partial X \setminus a} m_{b \to X}(x) \tag{6}$$

$$m_{a \to X}(x) :\propto \sum_{\mathbf{y} \in \mathcal{V}(\mathbf{X}_a \setminus X)} \phi_a(x, \mathbf{y}) \prod_{Y \in \mathbf{X}_a \setminus X} m_{Y \to a}(Y(\mathbf{y})), \tag{7}$$

where $Y(\mathbf{y})$ is the value of Y in the joint setting $\mathbf{y}$.

Observe that these calculations are (marginals of) products of factors, and thus correspond to inference in a "local" factor graph. The usual schematic illustration people draw to depict messages moving between variables and factors according to Equations (6) and (7) is given on the left of Figure 2. While this standard diagram is only a schematic, simply writing down the messages as unconditional probability distributions, we get a PDG $\mathcal{M}_{\mathcal{G}}$ that can be made to look very similar. Formally, a variable X^a for every pair (X, a) with $X \in \mathbf{X}_a$ along with edges asserting that $X^a = X$, we obtain the equivalent PDG on the right of Figure 2. Indeed, (6,7) minimize inconsistency of the controlled beliefs in their appropriate contexts (illustrated in Figure 2 (right) and formalized in Appendix B).

At the end, variable marginals $\{b_X\}_{X \in \mathcal{X}}$, which we regard as another PDG $\mathcal{B}$, are computed from the messages according to $b_X(x) \propto \prod_{a \in \partial X} m_{a \to X}(x)$.

Proposition 4. *If* Refocus *selects a focus nondeterministically from $\{a \to X, X \to a, X\}_{X \in \mathcal{X}, a \in \partial X}$ (see illustration above; details in Appendix B), then the possible runs of* LIR($\mathcal{M}_\Phi, \mathcal{M}_{\mathcal{G}} + \mathcal{B}$) *are precisely those of BP for different message schedules.*

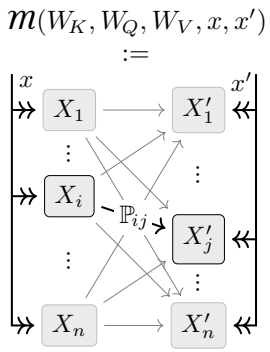
The same construction works for general cluster graphs, and we suspect it also reproduces the many message passing algorithms generated by α-divergences (Minka, 2005), because of the close relationship those divergences have with PDGs (Richardson, 2022, §5).

4.5 Transformer Layers

The key innovation of the transformer architecture (Vaswani et al., 2017), the basis of modern language models, is that of (scaled dot-product) (masked) *self-attention*. This notion of attention can be viewed as particularly useful prescription for how to set attention in a special case of our framework.

Suppose that we are looking at a sequence of n tokens, whose current (pre-transformation) representations at some layer of the model are the vectors $x_1, \ldots, x_n \in \mathbb{R}^d$. The transformation is parameterized by three matrices $W_K, W_Q, W_V \in \mathbb{R}^{d \times d}$, that generate key $k_i = W_K x_i$, query $q_i = W_Q x_i$, and value $v_i = W_V x_i$ representation vectors of each x_i. The output of the transformation are the vectors $x'_1, \ldots, x'_n$ given by $x'_j = \sum_{i=1}^n \alpha_{i|j} v_i$, where $\alpha_{i|j} := \text{softmax}_i(\langle k_i, q_j \rangle)$.

We can capture this with a relatively simple PDG (right). Start by assembling $2n$ $\mathbb{R}^d$-valued variables $\mathcal{X} = \{X_i\}_{i=1}^n \cup \{X'_i\}_{i=1}^n$. Start with a bipartite graph: for each $(i, j) \in [n]^2$, add an arc with isotropic Gaussian cpd $\mathbb{P}_{ij} = \mathcal{N}(X'_j \mid v_i, I)$; finally, add two additional arcs specifying the joint values of x and x'. The resulting PDG encodes the transformer as the optimal resolution of these many conflicting beliefs.



Proposition 5. *LIR($\mathcal{M}$) in which attention is set to $\varphi_{ij} = \exp\langle k_i, q_j\rangle$, when controlling x', leads to inference in the transformer layer,* i.e., $x'_j = \sum_i \alpha_{i|j} v_i$.

Notably, this straightforward model shows us that the softmax normalization must be row-wise, which is a common point of confusion; it tells us why attention $\alpha_{i|j}$ is a conditional probability pointing backwards in the network. Controlling the parameters W_K, W_Q, W_V trains the layer in a sense, but we do not know whether or not this is equivalent to the standard way of training.

4.6 Generative Flow Networks

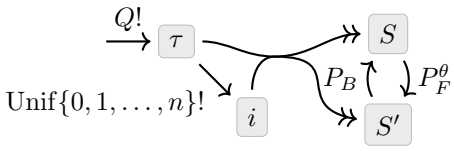
Generative Flow Networks, or GFlowNets (Bengio et al., 2021) aim to draw samples $x \in X$ proportional to a density $R(x)$ by modeling the flow of particles on a directed acyclic graph $(\mathcal{S} \supseteq X, \mathbb{A})$ from a source node s_0 to a sink node s_f. There are several different ways of parameterizing and training a GFlowNet. The most popular approach, called Trajectory Balance (TB) (Malkin et al., 2022a), involves parameterizing (1) a forward policy $P_F(s' \mid s)$ supported on $(s, s') \in \mathbb{A}$ that locally chooses edges given current state, (2) a policy $P_B(s \mid s')$ that travels backwards along edges, and (3) a normalizing constant $Z \approx \sum_{x \in X} R(x)$. The the training procedure aims to ensure probabilistic consistency between P_F and P_B: for every trajectory $\tau = (s_0, s_1, s_n = x, s_{n+1} = s_f)$, we want to ensure that

$$P_F(\tau) := \prod_{i=0}^{n} P_F(s_{i+1}) \approx \prod_{i=1}^{n} P_B(s_i \mid s_{i+1}) =: P_B(\tau),$$

where $P_B(\cdot \mid s_f) = R(x)/Z$. For any distribution $Q(\tau)$ with full support over possible trajectories, the quantity

$$\mathcal{L}_{\text{TB}}(Q) := \mathop{\mathbb{E}}_{\tau \sim Q}\left[\log^2 \frac{P_F(\tau)Z}{R(x)P_B(\tau \mid x)}\right]$$

is positive and equals zero iff all constraints are satisfied. In the special case where $Q(\tau) = P_F^\theta(\tau)$ and P_B is fixed, the full gradient $\nabla_\theta \mathcal{L}_{\text{TB}} = \nabla_\theta \boldsymbol{D}(P_F^\theta(\tau) \parallel P_B(\tau))$ is the same as that of reverse KL (Malkin et al., 2022b). Yet Q need not equal P_F (known as off-policy training), so GFlowNet objectives are thought to extend beyond KL. Yet by using a certain adaptive refocus strategy, TB can be captured with LIR, which is fundamentally based on KL. Concretely, we define a PDG $\mathcal{M}(\theta) :=$



(8)

where S and S' take possible values $\mathcal{V}S = \mathcal{V}S' = \mathcal{S}$, τ ranges over trajectories according to Q, i is an index in the trajectory chosen uniformly at random, and the complicated hyperarc simply ensures that $\tau[i] = S$ and $\tau[i+1] = S'$. These are essentially the minimal ingredients needed to put P_F, P_B, and Q together.

Now for the key ingredient. Surprise draws attention; thus, we set the attention in P_F is the forward policy's excess surprisal relative to the baseline P_B, that is, $\varphi(P_F) = \mathrm{I}_{P_F}[\tau] - \mathrm{I}_{P_B}[\tau] = \log \frac{P_B(\tau)}{P_F(\tau)}$, and vice versa: $\varphi(P_B) = \mathrm{I}_{P_B}[\tau] - \mathrm{I}_{P_F}[\tau] = \log \frac{P_F(\tau)}{P_B(\tau)}$. We then get:

Proposition 6. *The inconsistency of the PDG $\mathcal{M}$ from (8) with centered surprisal-based attention φ is*

$$\langle\!\langle \varphi \odot \mathcal{M} \rangle\!\rangle = \mathop{\mathbb{E}}_{\tau \sim Q}\left[\frac{1}{|\tau|}\log^2 \frac{P_F(\tau)Z}{R(x)P_B(\tau \mid x)}\right] =: \mathcal{L}_{\text{ModTB}}.$$

Furthermore, the gradients of $\mathcal{L}_{\text{ModTB}}$ differ only by a factor of two if φ is regarded as fixed and not a function of θ. Therefore, LIR on $\mathcal{M}$ with attention φ and control of P_F (and optionally P_B) amounts to training a GFlowNet with "re-weighted" TB loss.

This modified variant of trajectory-balance ($\mathcal{L}_{\text{ModTB}}$), is closely related to the original ($\mathcal{L}_{\text{TB}}$); in particular, it has the same global minimum in which all flow constraints are satisfied (Bengio et al., 2023). That said, there is a meaningful difference in training dynamics induced by the two losses, because of learning rates and batching. This is the first instance we know of in which the PDG framework has departed from standard practice for choosing a loss function in an established community. So how does it compare? As we will see in Section 5 and Appendix E, initial experiments suggest that $\mathcal{L}_{\text{ModTB}}$ actually works better than $\mathcal{L}_{\text{TB}}$. For an overview of experimental results, see Figure 3.

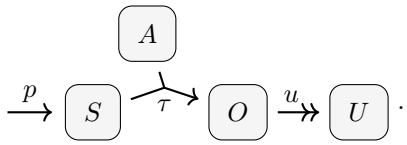
4.7 Decision Making

In this final illustration of the expressive power of LIR, we return to psychological models of agents, as we show how some standard decision rules can be viewed as inconsistency minimization. PDG inconsistencies can represent expected costs (Richardson, 2024, §4, §6), and thus can jointly represent an agent's probabilistic beliefs and utilities. What does it mean to make decisions to as to minimize inconsistency in this context?

Let's formalize the traditional setup of decision theory with a PDG. Suppose we are trying to choose an action (a setting of the variable A), S but we are uncertain about the current state of the world (captured by a variable S). Let O be a variable representing the final outcome. To connect these pieces of the puzzle, the standard assumption is that we have

1. a belief $p(S)$ about the state of the world,
2. a utility function $u : \mathcal{V}O \to \mathbb{R}$ on outcomes, and
3. an understanding $\tau(O \mid S, A)$ of how our action A and the state S determine the outcome O.

It is easy to encode this information in a PDG:



(9)

What's missing is the idea that a higher numerical utility should be "better" than a lower one. For this, we can add a "soft constraint" (see Richardson, 2024, §4.2.2). Let $\mathcal{M}$ represent the PDG above augmented with such a soft constraint.

To give a stylized interpretation, imagine that there is a part of you that you cannot control called "Faith". Faith is optimistic: she disbelieves outcomes that are undesirable, creating epistemic conflict in proportion to disutility. If you have no control over Faith, but still have to pay attention to her, then selecting actions to minimize inconsistency is a decision rule that interpolates between

(a) maximizing expected utility (when $\beta_p \gg \beta_b$), and
(b) maximizing maximum utility (when $\beta_p \ll \beta_b$),

where β_p is your confidence in your prior probabilistic belief $p(S)$, and β_b is the attention to the soft constraint b (*i.e.*, your "degree of faith").

Proposition 7. *Let $\mathcal{M}$ be the PDG in (9) together with the cpd $b(\mathtt{T}{=}\mathtt{t} \mid U{=}u) = k \cdot \exp(u)$ for some $k > 0$.*

(a) If $\beta_p = \infty$ and $\beta_b < \infty$, then the action(s) $a \in \mathcal{V}A$ that minimize the inconsistency are those that maximize expected utility. Formally, for all $\gamma \geq 0$,

$$\arg\min_{a \in \mathcal{V}A} \langle\!\langle \mathcal{M} + A{=}a \rangle\!\rangle_\gamma = \arg\max_{a \in \mathcal{V}A} \mathop{\mathbb{E}}_{\substack{s \sim p \\ o \sim \tau \mid s,a}} \big[u(o)\big].$$

(b) If $\beta_p < \infty$ and $\beta_b = \infty$, then the action(s) $a \in \mathcal{V}A$ that minimize overall inconsistency are those that can lead to the best possible outcome, i.e.*,*

$$\arg\min_{a \in \mathcal{V}A} \langle\!\langle \mathcal{M} + A{=}a \rangle\!\rangle_\gamma = \arg\max_{a \in \mathcal{V}A} \max_{s \in \mathcal{V}S} \mathop{\mathbb{E}}_{o \sim \tau \mid s,a} [u(o)].$$

We suspect that it may also be possible to implement other decision rules such as minimax or maximin. Like the GAN objective however, these decision rules look like two-player games, and so will likely require two different focuses of LIR, rather than just one. We leave a careful exploration of this avenue to future work.

We conclude with a high-level observation. One's preferences, beliefs, and actions can be jointly modeled with a PDG; when when that PDG is inconsistent, there are three possible resolutions (just as in Section 4.1). You can resolve the inconsistency by changing your action, which amounts to maximizing expected utility (Proposition 7.1); this is thought of as the rational approach. Alternatively, you can change your preferences so as to become content with your current situation, which is perhaps a more zen perspective. These two resolutions of the internal conflict are two halves of the famous Serenity Prayer (Niebuhr, 1933):

> *Give us the courage to change what must be altered, serenity to accept what can not be helped, and insight to know the one from the other.*

That "insight" amounts to the choice of the control mask χ. There is also a third way to resolve the inconsistency: change beliefs $p(S)$ about the state of the world (*i.e.*, wishful thinking). However, such a strategy may lead to more inconsistency in the future, making it undesirable and ineffective. This example clearly shows that there can be good reasons for restricting control χ beyond computational savings or the limits of one's power.

5 IMPLEMENTATION AND EXPERIMENTS

Implementation. We implemented LIR by extending an existing code-base for PDGs. The primary challenge was in getting (sufficiently high-quality) gradients of the inconsistency efficiently enough to run the procedure on small examples and use an ODE solver. More details can be found in Appendix A.1.

Synthetic Experiments. As a proof-of-concept of our generic LIR implementation, we evaluate it by comparing the effect of various Refocus procedures on a dataset of randomly generated PDGs. We compare the global uniform attention, in which all arcs are active ($\boldsymbol{\beta} = \mathbf{1}$), to various more local foci, in which only a subset of edges are active ($\beta(a) = 1$ for some arcs a and $\beta(a') = 0$ for others). In all, we study four refocus strategies: global attention (all arcs active), local (half of arcs randomly deactivated), node-neighborhood (arcs active if they are sources or targets of a a randomly chosen node), and random smooth masks ($\boldsymbol{\beta}$ drawn from an exponential distribution). In each case, we evaluate the behavior of LIR based on several criteria: (1) the (rate of) reduction of inconsistency from initialization; (2) the computational cost; (3) the degree to which the optimal distribution $\mu^* \in \arg\min OInc_{\mathcal{M}}(\mu)$ is distorted during the process. Details about these experiments, can be found in Appendix A.2.

Modified TB Losses. Arising from our observations in 4.6, we noted that the original TB loss omits a trajectory length scaling factor (see Proposition 6 for the definition of $\mathcal{L}_{\text{ModTB}}$, the Modified TB loss). We analyze whether including this scaling factor during training produces meaningfully different results under four HyperGrid environments of varying difficulty. See Appendix E for all experimental details.

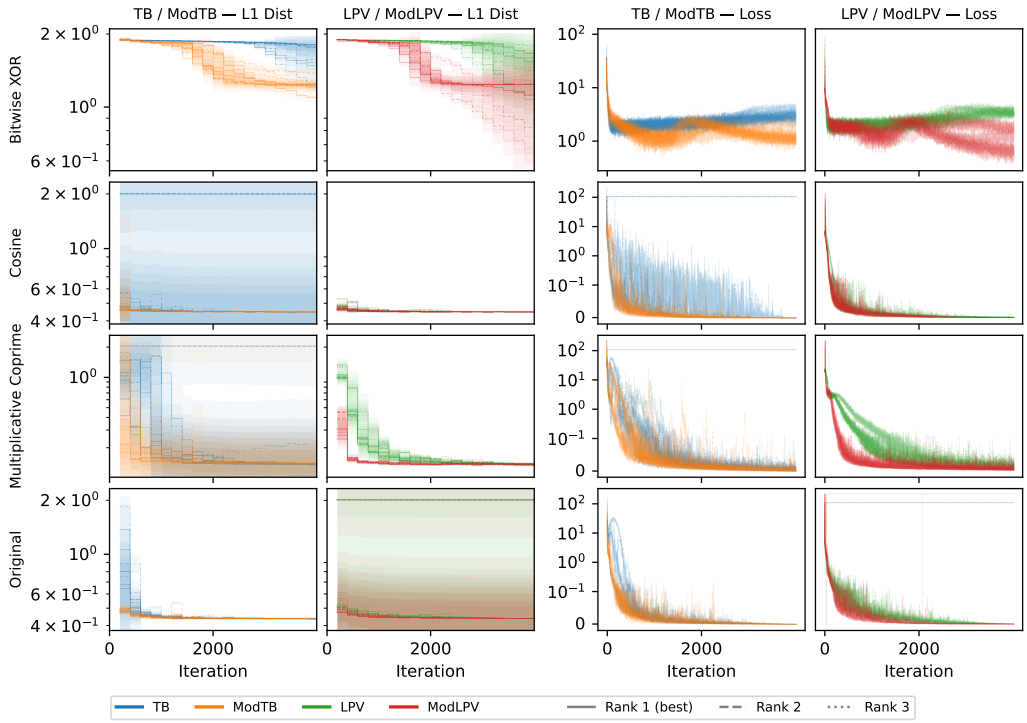


Figure 3: Performance (L1 distance between the estimated and true posterior distribution; left) and loss (computed using the unnormalized variant at evaluation time for visualization at the same scale; right) of the Trajectory Balance (TB) and Log-Partition Variance (LPV) losses and their normalized variants (Mod TB and Mod LPV, respectively). Figures show traces from the 3 top hyperparameter configurations (ranks) found with 5 seeds each rank. The performance curves show overlaid on the traces a 1D KDE fit in log space at each timestep, showing relative per-step density.

A cousin of TB is the Log Partition Variance loss, or "VarGrad" (Richter et al., 2020; Malkin et al., 2023), which admits an analogous modified variant. For a minibatch of trajectories $(\tau_i)_{i=1}^m$ drawn from Q, let

$$s_i := \log \frac{P_F(\tau_i)}{R(x_i)\, P_B(\tau_i \mid x_i)}, \qquad \bar{s} = \frac{1}{m}\sum_i s_i$$

be the mini-batch mean, and and define

$$\mathcal{L}_{\text{ModLPV}} = \frac{1}{m}\sum_{i=1}^{m} \frac{(s_i - \bar{s})^2}{|\tau_i|}.$$

Both were implemented using `torchgfn` (Viviano et al., 2026). Figure 3 shows 5 seeds from the top 3 hyperparameter settings found after a 50 Optuna trials (Akiba et al., 2019). All models covered all modes in all environments, but training stability varied, with larger differences apparent in the more difficult compositional environments The modified losses generally show faster convergence at the earlier stages of training. For the challenging environments (e.g., *Bitwise XOR*), there was a substantial difference also in the fidelity of the learned distribution. Of note, the best gradient clipping (norm-based) value found for the best LPV models was much more aggressive (see Table 1), while the modified variant was much more tolerant of larger gradient norms. See Appendix E Table 3 for more.

Environment	LPV	ModLPV	Ratio
Original	0.025	0.23	$\sim 10\times$
Cosine	0.10	0.045	$\sim 2\times$
Bitwise XOR	0.033	1.19	$\sim$ **36**$\times$
Multiplicative Coprime	0.013	0.84	$\sim$ **65**$\times$

Table 1: Median gradient clipping threshold (norm-based) across the top-10 Optuna trials for each loss.

6 CONCLUSION

We have introduced Local Inconsistency Resolution (LIR), a framework that unifies and explains a wide variety of important algorithms, as stemming from different allocations of attention and control. It has led us to a slightly better formulation of trajectory balance, a standard loss for training GFlowNets.

Many fundamental questions remain. Are there interesting non-trivial assumptions that ensure convergence? Can we learn adaptive Refocus mechanisms (drawing inspiration from the transformer architecture)? Can we use it to discover new hybrid algorithms? Answering these questions could advance a new kind of intelligent system—one that is more interpretable and potentially safer: behavior emerges from resolving conflicts among explicit adjustable beliefs rather than rigidly pursuing a fixed goal in the world.

Acknowledgments

This work was supported in part by funding from NSERC and CIFAR.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA. Association for Computing Machinery.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. (2023). Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Š rndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *Advanced Information Systems Engineering*, pages 387–402. Springer Berlin Heidelberg.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22.

Festinger, L. (1962). Cognitive dissonance. *Scientific American*, 207(4):93–106.

Friston, K. (2005). A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37:183–233.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kishore, V., Chen, X., Wang, Y., Li, B., and Weinberger, K. Q. (2021). Fixed neural network steganography: Train the images, not the network. In *International Conference on Learning Representations*.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. (2023). Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022a). Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. (2022b). Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E. J., Everett, K. E., Zhang, D., and Bengio, Y. (2023). GFlownets and variational inference. In *The Eleventh International Conference on Learning Representations*.

McLachlan, G. J. and Krishnan, T. (2008). *The EM algorithm and extensions*. John Wiley & Sons.

Minka, T. (2005). Divergence measures and message passing. Technical Report MSR-TR-2005–173, Microsoft Research, Cambridge, U.K.

Müller, R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? *Advances in neural information processing systems*, 32.

Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Springer.

Niebuhr, R. (1933). page 1.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87.

Richardson, O. E. (2022). Loss as the inconsistency of a probabilistic dependency graph: Choose your model, not your loss function. *AISTATS '22*, 151.

Richardson, O. E. (2024). *A Unified Theory of Probabilistic Modeling, Dependence, and Inconsistency*. PhD thesis.

Richardson, O. E. and Halpern, J. Y. (2021). Probabilistic dependency graphs. *AAAI '21*.

Richardson, O. E., Halpern, J. Y., and Sa, C. D. (2023). Inference in probabilistic dependency graphs. *UAI '23*.

Richardson, O. E., Peters, S., and Halpern, J. Y. (2024). Qualitative mechanism independence. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 69471–69504. Curran Associates, Inc.

Richter, L., Boustati, A., Nüsken, N., Ruiz, F., and Akyildiz, O. D. (2020). Vargrad: a low-variance gradient estimator for variational inference. *Advances in Neural Information Processing Systems*, 33:13481–13492.

Shafahi, A., Huang, W. R., Studer, C., Feizi, S., and Goldstein, T. (2018). Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Viviano, J. D., Younis, O. G., Choi, S., Schmidt, V., Bengio, Y., and Lahlou, S. (2026). torchgfn: A pytorch gflownet library.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes.**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **No.**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes.**

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes.**
   (b) Complete proofs of all theoretical results. **Yes.**
   (c) Clear explanations of any assumptions. **Yes.**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **At least some.**
   (b) All the training details (*e.g.*, data splits, hyperparameters, how they were chosen). **Yes.**
   (c) A clear definition of the specific measure or statistics and error bars (*e.g.*, with respect to the random seed after running experiments multiple times). **Yes.**
   (d) A description of the computing infrastructure used. (*e.g.*, type of GPUs, internal cluster, or cloud provider). **Yes.**

4. If you are using existing assets (*e.g.*, code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Not Applicable**
   (b) The license information of the assets, if applicable.**Not Applicable**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Not Applicable**
   (d) Information about consent from data providers/curators. **Not Applicable**
   (e) Discussion of sensible content if applicable, *e.g.*, personally identifiable information or offensive content. **Not Applicable**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

Local Inconsistency Resolution: The Interplay between Attention and Control in Probabilistic Models

Supplementary Materials

A Experimental and Implementation Details

A.1 Implementation

We use PyTorch for our implementation. One of the most basic PDG computations is to compute the incompatibility, denoted by f, of a PDG $\mathcal{M}$ given joint distribution μ and attention mask φ. The attention mask decides how much each arc contributes to the incompatibility. As a result, many variables may have no contribution at all. We prune these variables for computational efficiency.

The γ-inconsistency of a PDG $\mathcal{M}$ (Equation (3)) is obtained by finding the joint distribution μ that minimizes incompatibility. We represent the joint distribution μ as a point on a simplex and use the Adam optimizer for minimizing incompatibility w.r.t. μ. This inner optimization procedure is warm-started from the last computed joint distribution to increase computational efficiency.

To obtain the gradient of inconsistency w.r.t. learnable parameters, denoted by θ, we use the *envelope theorem* to simplify differentiation through the inner optimization problem

$$\nabla_\theta \min_\mu f(\mu, \theta) \;=\; \nabla_\theta f(\mu^*(\theta), \theta),$$

where $\mu^*(\theta) = \arg\min_\mu f(\mu, \theta)$. Hence, there is no need to backpropagate through the inner optimization: we simply plug in the numerically optimized joint distribution μ^* and compute gradients of the incompatibility w.r.t. θ. This gradient is then scaled by the control mask χ. For the outer optimization of θ we implement both standard optimizers such as Adam as well as ODE solvers such as Runge-Kutta.

A.2 Synthetic Experiments

Local inconsistency resolution employs attention mechanisms to selectively focus computational resources on subsets of the model during training. However, the choice of refocus strategy fundamentally impacts both efficacy of inconsistency resolution and the computational efficiency of the learning process. To systematically evaluate different attention strategies, we designed a suit of synthetic experiments using PDGs with controllable structural properties and known inconsistencies.

We generate synthetic PDGs with deliberately introduced inconsistencies to create a controlled experimental environment. Each PDG is constrained as follows:

Variables Each PDG contains n discrete random variables $\{X_1, \ldots, X_n\}$, where each variable X_i has a small finite domain $\mathcal{V}X_i$ (consisting of 2 or three possible values).

Graph Structure For these preliminary experiments, we study a simple class of discrete PDGs whose underlying hypergraph structure is an ordinary directed graph. The edge set is constructed in two phases:

1. **Base Chain Construction** We first create a chain structure with $\lfloor m/2 \rfloor$ edges, where m is the target number of edges:

$$\mathcal{A}_{\text{chain}} = \{(i \to i+1) : i \in [1, \lfloor m/2 \rfloor]\}$$

2. **Conflict Edge Addition:** We then add additional edges preferentially targeting nodes that already have incoming edges, creating conflict points:

$$\mathcal{A}_{\text{conflict}} = \{(i \to j) : j \in \text{Targets}(\mathcal{A}_{\text{chain}}), i \neq j\}$$

This construction guarantees that certain nodes receive multiple incoming edges with potentially conflicting conditional probability specifications, ensuring non-zero inconsistency.

Conditional Probability Distributions For each edge $(i \to j) \in \mathcal{A}$, we assign a randomly initialized conditional probability table $\mathbb{P}_{ij}(X_j|X_i)$ drawn uniformly from the probability simplex. Crucially, these CPDs are *not* jointly consistent, there generally does not exist a joint distribution μ that satisfies all CPDs simultaneously, guaranteeing measurable inconsistency.

We evaluate four PDG configurations, denoted as `chain_`(n)`v_`(m)`e` for n variables and m edges.

PDG	Variables	Edges	Initial Inconsistency
`chain_4v_3e`	4	3	0.159294
`chain_5v_4e`	5	4	0.168060
`chain_6v_5e`	6	5	0.209741
`chain_7v_6e`	7	6	0.145324

A.2.1 Inconsistency Measure

As is the case for most of our results, our Given a PDG $\mathcal{M}$ with edge set $\mathcal{A}$ and a joint distribution μ over variables $\mathcal{X} = (X_1, \ldots, X_n)$, recall (from (1)) that the (observational) incompatibility is measured as:

$$OInc(\mu, \mathcal{M}) = \sum_{(i,j)\in\mathcal{A}} \beta_{ij} \cdot D_{\text{KL}}\left(\mu(X_j|X_i) \,\|\, \mathbb{P}_{ij}(X_j|X_i)\right), \tag{10}$$

where:

- $\mu(X_j|X_i)$ is the conditional distribution induced by the joint distribution μ
- $\mathbb{P}_{ij}(X_j|X_i)$ is the conditional probability table specified by edge $i \to j$
- $\beta_{ij} \in [0, 1]$ is the attention weight for edge $i \to j$
- $D_{\text{KL}}(P\|Q) = \boldsymbol{D}(P \,\|\, Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$ is the Kullback-Leibler divergence.

The optimal joint distribution μ^* for a given set of CPDs is obtained by solving:

$$\mu^* = \arg\min_{\mu\in\Delta\mathcal{V}\mathcal{X}} OInc(\mu, \mathcal{M}), \tag{11}$$

and the *inconsistency* of $\mathcal{M}$ is given by

$$\langle\!\langle \mathcal{M} \rangle\!\rangle = OInc(\mu^*, \mathcal{M}) = \inf_{\mu\in\Delta\mathcal{V}\mathcal{X}} OInc(\mu, \mathcal{M}). \tag{12}$$

A.2.2 Refocus Strategies

We evaluate three refocus strategies that control the β weights.

Uniform Focus All edges receive full attention at all times:

$$\beta_{ij}^{\text{uniform}} = 1 \quad \forall (i \to j) \in \mathcal{A} \tag{13}$$

This strategy optimizes all edges simultaneously, maximizing global consistency but requiring the most computational resources.

Partial Focus At each training iteration t, we randomly select a subset $\mathcal{A}_{\text{active}}(t) \subseteq \mathcal{A}$ with $|\mathcal{A}_{\text{active}}(t)| = \lfloor |\mathcal{A}|/2 \rfloor$, and focus attention only on this subset:

$$\beta_{ij}^{\text{partial}}(t) = \begin{cases} 1 & \text{if } (i \to j) \in \mathcal{A}_{\text{active}}(t) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

This strategy reduces computational cost by ignoring half the graph at any given time, but may achieve slower or incomplete convergence.

Hub Focus At each iteration t, we select a random focus node $X_{\text{focus}}(t)$, and attend to all edges incident to this node:

$$\beta_{ij}^{\text{hub}}(t) = \begin{cases} 1 & \text{if } X_i = X_{\text{focus}}(t) \text{ or } X_j = X_{\text{focus}}(t) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

This strategy focuses on potential inconsistencies that are, in a sense, localized to a single variable.

A.2.3 Training Procedure

For each PDG and focus strategy, we perform the following training procedure:

1. **Initialization:** Convert each fixed CPD to a learnable parameterized conditional probability distribution (`ParamCPD`) initialized from the original CPD values.

2. **Initial Joint Distribution:** Compute the initial optimal joint distribution μ^*_{init} by solving:

$$\mu^*_{\text{init}} = \arg\min_{\mu} OInc(\mu, \mathcal{M}_{\text{init}}) \tag{16}$$

   using the Adam optimizer with $\gamma = 0$ (no entropy regularization) for 50 iterations.

3. **LIR Training:** Apply LIR with the specified refocus strategy for $T = 20$ timesteps. At each timestep t, our implementation of LIR updates CPD parameters θ by approximating the solution to the ODE

$$\theta_{t+1} \leftarrow \text{SolveODE}\Big[\dot{\theta} = \nabla_\theta \langle\!\langle \mathcal{M}(\theta), \boldsymbol{\beta} \rangle\!\rangle;\ \texttt{init} = \theta_t\Big]$$

   by applying `#outer_iterations` gradient-based steps of learning rate η. (We have effectively set a uniform control mask $\chi = \eta$ equal to the learning rate.) The attention weights $\boldsymbol{\beta}$ are determined by the chosen focus strategy. In this particular experiment, that outer gradient step is done by the Adam optimizer.

   Within each outer iteration $o = 1, 2, \cdots$, we approximate the optimal joint distribution $\mu^*_{(t,o)}$ by fine-tuning the previous solution $\mu^*_{(t,o)}$ for `#inner_iterations` steps. For more details, see Section A.1 and the code in `lir_simpler.py`.

   Our experiments use the following values:

   - 10 outer iterations per timestep to approximate the ODE solution
   - 20 inner iterations for optimizing μ^* per outer iteration
   - Learning rate $\eta = 0.05$

4. **Final Joint Distribution:** After training, compute the final optimal joint distribution μ^*_{final} using the updated CPDs, for the purposes of analysis.

A.2.4 Evaluation Metrics

We evaluate each refocus strategy using three complementary metrics:

Resolution Percentage The resolution percentage measures the reduction in inconsistency:

$$\text{Resolution} = \frac{OInc(\mu^*_{\text{init}}, \mathcal{M}_{\text{init}}) - OInc(\mu^*_{\text{final}}, \mathcal{M}_{\text{final}})}{OInc(\mu^*_{\text{init}}, \mathcal{M}_{\text{init}})} \times 100\% \qquad = \frac{\langle\!\langle \mathcal{M}_{\text{init}} \rangle\!\rangle - \langle\!\langle \mathcal{M}_{\text{final}} \rangle\!\rangle}{\langle\!\langle \mathcal{M}_{\text{init}} \rangle\!\rangle} \times 100\% \tag{17}$$

Higher resolution indicates more effective inconsistency reduction. Fig 4 compares the initial and final inconsistency values across multiple PDGs and three inconsistency-resolution refocus strategies: global focus, partial focus, and hub focus. Across all PDGs, inconsistency values consistently decreased after training, confirming that the optimization procedure effectively resolved contradictions in the dependency structure. The magnitude of reduction varied by both graph complexity and refocus procedure:

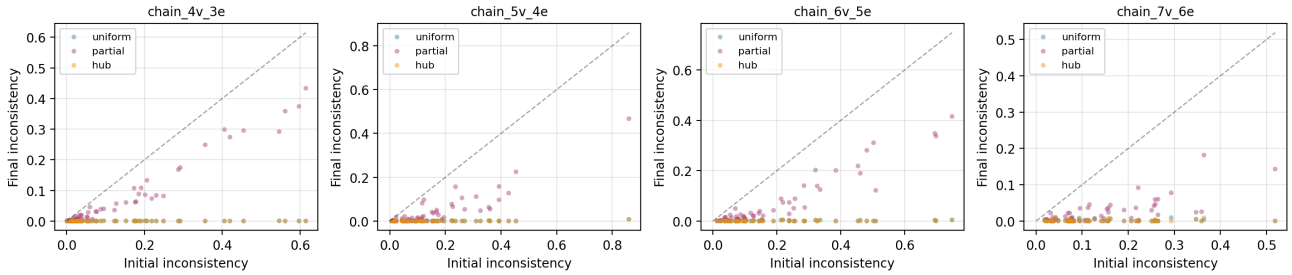


Figure 4: Initial vs. final inconsistency across PDGs for *uniform*, *partial*, and *hub* focus strategies.

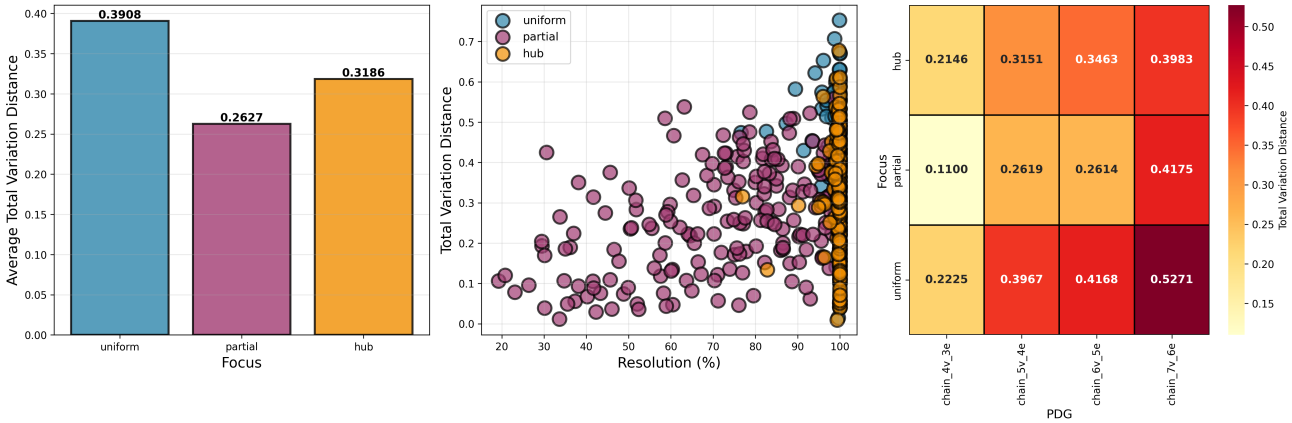


Figure 5: A breakdown of TV distortion through the LIR process across refocus strategies and PDGs

Distortion Analysis: Measuring Changes to the Joint Distribution There is a simple but inadvisable and destructive way to instantly resolve all inconsistencies: set all beliefs to uniform (or delete them altogether). Does something like this happen in our case? While resolution percentage quantifies how effectively a refocus strategy reduces inconsistency, it does not capture how much the underlying probabilistic beliefs change during the resolution process. Two strategies might achieve similar resolution but differ substantially in the magnitude of changes they induce to the optimal joint distribution μ^*. To address this, we introduce a distortion metric that measures the distance between the initial and final joint distributions. We employ the *Total Variation (TV) distance*, a standard metric for comparing probability distributions, defined as:

$$D_{TV}(\mu^*_{\text{init}}, \mu^*_{\text{final}}) = \frac{1}{2} \sum_{\mathbf{x} \in \mathcal{V}\mathcal{X}} |\mu^*_{\text{init}}(\mathbf{x}) - \mu^*_{\text{final}}(\mathbf{x})| = \max_{A \subseteq \mathcal{V}\mathcal{X}} \left|\mu^*_{\text{init}}(A) - \mu^*_{\text{final}}(A)\right|. \tag{18}$$

The reported TV distortion in Figure 5 is the distance between initial μ_{init} and final optimized μ_{final}, not a regularized penalty for cpd change. So strategies that resolve inconsistency more aggressively can naturally produce larger mu shifts, and therefore larger TV distance.

As seen in Figure 5, the hub refocus strategy induces higher distortion than alternatives, particularly for the larger PDGs. The partial focus strategy appears to induce less distortion, but in fact produces more distortion per unit of inconsistency resolution. All results presented here have additional distortion owing to the optimizer (here we've used Adam instead of a dedicated ODE solver), and the fact that we do not pre-multiply by the inverse Fisher matrix to get natural gradients.

Resolution Performance Figure 6 presents a resolution percentage heatmap comparing the effectiveness of three inconsistency-resolution focus strategies (uniform, partial, and hub) across four PDGs. The color scale represents the percentage reduction in inconsistency, with higher values (green) indicating faster convergence. Across all PDGs, both uniform and hub focus consistently achieved high resolution percentages, particularly in larger graphs such as `chain_7v_6e`, where both surpassed 90%. Partial focus, however, displayed considerably

Table 2: Inconsistency resolution performance by refocus strategy

Refocus Strategy	Avg Resolution (%)	Avg TV Dist.
Uniform	99.23	0.3908
Partial	72.19	0.2627
Hub	99.37	0.3186

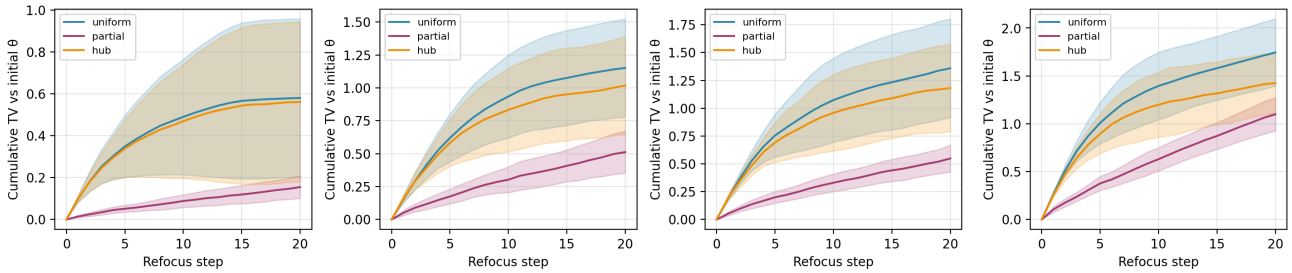


Figure 6: Mean cumulative total variation distance between current CPD parameters (θ) and their initial values (θ_0) across LIR refocus steps, shown separately for each PDG and refocus strategy (uniform, partial, hub), with shaded bands indicating ± 1 standard deviation over CPD replicates.

weaker performance in smaller PDGs.

Fig 7 summarizes the resolution performance of the three inconsistency-resolution strategies—uniform, partial, and hub across four PDGs. Across all PDGs, the uniform and hub strategies consistently outperform the partial approach.

The partial focus strategy catches up somewhat in larger graphs (up to 83.5%), suggesting that localized updates may become more effective when many more interdependencies exist. Overall, these results indicate that the uniform and hub focus may scale better across larger PDGs (at least with this kind of topology), while partial focus is typically less effective, and sensitive to graph density and connectivity.

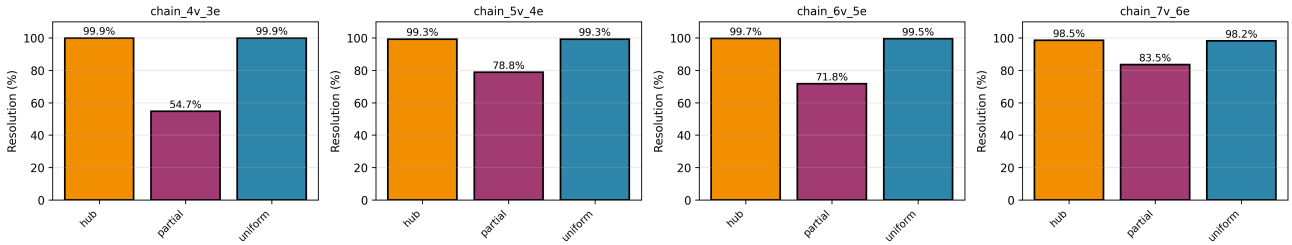


Figure 7: Resolution performance by focus strategy across four PDGs. Uniform and hub approaches consistently achieve higher inconsistency reduction than local updates, with performance converging at larger graph sizes.

B Details on Belief Propagation

We now define the foci. Modulo a small subtlety, the following is essentially true: (6) selects $C_{X\to a} := \{m_{X\to a}\}$ so as to minimize 1-inconsistency in context $A_{X\to a} := \{m_{b\to X}\}_{b\in\partial X\setminus a} \cup \{m_{X\to a}\}$, while (7) selects $C_{a\to X} := \{m_{a\to X}\}$ so as to minimize the 1-inconsistency in context $A_{a\to X} := \{\phi_a, m_{a\to X}\} \cup \{m_{Y\to a}\}_{Y\in\mathbf{X}_a\setminus X}$.

The only wrinkle is that we do not attend to the *structural* aspect of the edges e that we're updating; *i.e.*, we must select φ to effectively set $\alpha_e = 0$. Intuitively: although all of the input messages summarize causal information, we're trying to capture that information with a distribution. Thus, it's not appropriate to attend to the causal structure of the edges that we're modifying. Thus, for each $f \in \bigcup_{a\in\mathcal{A},X\in\mathbf{X}_a}\{a\to X, X\to a, X\}$, define a

focus $(\varphi_f, \chi_f) \in \mathbf{F}$ according to

$$\varphi_f(a) := \begin{cases} \binom{1}{1} & \text{if } a \in A_f \setminus C_f \\ \binom{1}{0} & \text{if } a \in C_f \\ \binom{0}{0} & \text{otherwise} \end{cases}, \qquad \chi_f(a) := \begin{cases} \infty & \text{if } a \in C_v \\ 0 & \text{otherwise.} \end{cases}$$

where $\binom{\phi_1}{\phi_2}$ scales β_a by ϕ_1 and α_a by ϕ_2. With these definitions, Proposition 4 follows easily.

C Note on Densities, Mass Functions, and Constants

Before we move on to the proofs, we start by explicitly addressing an (ultimately inconsequential) slight-of-hand. While $\boldsymbol{D}(\delta_x(X) \parallel p(X)) = -\log p(X{=}x) = \infty$ when $p(X{=}x) = 0$, as is the case when p is the measure corresponding to a positive density on real numbers (such as a normal distribution), there is an important sense in which it effectively equals $-\log p(x) + \text{const}$, where $p(x) = \frac{\mathrm{d}p}{\mathrm{d}\lambda}(x)$ is the density with respect to the base (Lebesgue) measure λ. For example, if we discretize the space $\mathcal{V}X$ of values of X as a countable collection of voxels of size ϵ, resulting in a new discrete variable X_ϵ and a cpd p^ϵ over its values, then

$$\boldsymbol{D}(\delta_x(X_\epsilon) \parallel p^\epsilon(X_\epsilon)) = -\log p(x) + o(\epsilon^2) + \log \frac{1}{\epsilon}.$$

We obtain essentially the same limit if we use a definition of the Dirac distribution $\delta_x(X)$ as a limit of normal distributions whose variance goes to zero. In our case, since we are only ever interested in the gradient of the inconsistency with respect to parameters θ, and $\nabla_\theta \log \frac{1}{\epsilon} = 0$, so $\lim_{\epsilon \to 0} \nabla_\theta \boldsymbol{D}(\delta_x(X_\epsilon) \parallel p^\epsilon(X_\epsilon)) = -\nabla_\theta \log p_\theta(x)$.

We therefore perform the remainder of our calculations as though $\boldsymbol{D}(\delta_x(X) \parallel p^\theta(X)) = -\log p_\theta(x)$.

The same slight-of-hand is standard practice in variational inference for essentially these reasons. For example, suppose we have a latent variable model $p(X, Z)$ where X and Z are both real-valued variables that we measure in meters; the density $p(x, z)$ is then probability per square meter. In the expression $\text{ELBO}(x) = \mathbb{E}_q \left[\log \frac{p(x,z)}{q(z)} \right]$, the units of $p(x, z)$ and $q(z)$ do not cancel before the logarithm, and therefore the result is affected by changing our units of measurement from meters to feet. However, the effect is to add a constant, and so this semantic point becomes irrelevant when the quantity is used as a loss function.

D Proofs

First, some extra details for Proposition 1. By parameteriations $\mathbb{P}$ log-concave, we mean that, for every $a \in \mathcal{A}$, and $(s, t) \in \mathcal{V}(S_a, T_a)$, the function

$$\theta \mapsto -\log \mathbb{P}_a^\theta(T_a = t \mid S_a = a) \quad : \Theta_a \to [0, \infty]$$

is convex. This is true for many families of distributions of interest. For example, if S_a, T_a is discrete, and the cpd is parameterized by stochastic matrices $\theta := \mathbf{P} = [p_{s,t}] \in [0, 1]^{\mathcal{V}(S_a, T_a)}$, then

$$-\log \mathbb{P}_a^{\mathbf{P}}(T_a = t | S_a = s) = -\log(p_{s,t})$$

which is clearly convex in $\mathbf{P}$.

To take another example: if $\mathbb{P}_a$ is linear Gaussian, *i.e.*, $\mathbb{P}_a(T|S) = \mathcal{N}(T|\mathbf{A}s + b, \sigma^2)$, parameterized by $(\mathbf{A}, b, 1/\sigma^2)$, then

$$-\log \mathbb{P}_a^{(\mathbf{A}, b, \sigma^2)}(t|s) = -\frac{1}{2} \log \frac{2\pi}{\sigma^2} + \frac{1}{2} \left(\frac{t - \mathbf{A}s + b}{\sigma} \right)^2$$

which is convex in $(\mathbf{A}, b, \frac{1}{\sigma^2})$. Now, for the proof.

Proposition 1. *If $\mathcal{M}(\Theta)$ is an unweighted parametric PDG whose parameterizations $\mathbb{P}$ are either constant or unconditional and log-concave and $\boldsymbol{\beta} \geq \gamma \boldsymbol{\alpha}$, the map $\boldsymbol{\theta} \mapsto \langle\!\langle \mathcal{M}(\boldsymbol{\theta}), \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle\!\rangle_\gamma$ is convex.*

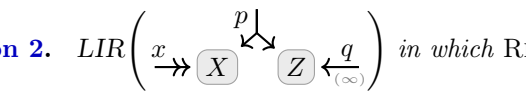
Proof. By definition,

$$\langle\!\langle \varphi \odot \mathcal{M}(\theta) \rangle\!\rangle = \inf_{\mu} \Big\{ \gamma \mathit{SDef}_{\mathcal{M}(\theta)}(\mu) + \mathit{OInc}_{\mathcal{M}(\theta)}(\mu) \Big\}.$$

Only the final term actually depends on θ, though. Let $F(\mu)$ capture the first term. For all of our examples, and indeed, if $\boldsymbol{\beta} \geq \gamma \boldsymbol{\alpha}$, it will be convex in μ (Richardson and Halpern, 2021, from the proof of Proposition 3.2). Then we have

$$\begin{aligned}
\langle\!\langle \varphi \odot \mathcal{M} \rangle\!\rangle &= \inf_{\mu} \left(F(\mu) + \mathop{\mathbb{E}}_{\mu} \left[\sum_{S \xrightarrow{a} T} \beta_a \log \frac{\mu(T|S)}{\mathbb{P}_a(T|S)} \right] \right) \\
&= \inf_{\mu} \left(F(\mu) + \mathop{\mathbb{E}}_{\mu} \left[\sum_{S \xrightarrow{a} T} \beta_a \log \frac{\mu(T|S)}{\lambda(T|S)} \right] + \mathop{\mathbb{E}}_{\mu} \left[\sum_{S \xrightarrow{a} T} \beta_a \log \frac{\lambda(T|S)}{\mathbb{P}_a(T|S)} \right] \right)
\end{aligned}$$

The second term is then entropy (relative to the base distribution), which is convex in μ. The first term, $F(\mu)$, is convex in μ as well, and neither depend on θ. The final term is linear in μ. Since $\mathbb{P}$ is log-concave in θ, each $\log \frac{\lambda(t|s)}{\mathbb{P}_a(t|s)}$ for $a \in \mathcal{A}$ convex in θ, and so that third term is a conic combination of terms that are all convex, and hence itself convex in θ. Thus, the sum of all three terms in the infimum is jointly convex in θ and in μ. Taking an infimum over μ pointwise, the result is still convex in θ. □

Proposition 2. *LIR*() *in which* Refocus *re-samples x and alternates between full control of p and q implements EM, in that $\theta^{(t)}_{EM} = \theta^{(2t)}_{LIR}$.*

Proof. Let $\mathcal{M}_{q,\theta}$ denote the PDG in the statement with observation $X = x$, generative arc labeled by p_θ, and an inference arc q into Z with $\beta = \infty$ (high confidence). By Richardson (2022, Proposition 11) ("negative ELBO equals inconsistency"),

$$\langle\!\langle \mathcal{M}_{q,\theta} \rangle\!\rangle = -\mathrm{ELBO}_{p,q}(x) = -\mathop{\mathbb{E}}_{q(z)} \left[\log \tfrac{p(x,z;\theta)}{q(z)} \right] = \boldsymbol{D}\big(q(z) \,\big\|\, p(z \mid x;\theta)\big) - \log p(x;\theta).$$

Thus, for fixed $p(\cdot;\theta)$, minimizing $\langle\!\langle \mathcal{M}_{q,\theta} \rangle\!\rangle$ over q is equivalent to minimizing $\boldsymbol{D}\big(q \,\big\|\, p(z \mid x;\theta)\big)$.

Full control of q (E-step): Fix θ. Since $\boldsymbol{D}(\cdot \,\|\, \cdot) \geq 0$ with equality iff its two arguments agree,

$$q^* \in \arg\min_{q} \langle\!\langle \mathcal{M}_{q,\theta} \rangle\!\rangle \quad \Longleftrightarrow \quad q^*(z) = p(z \mid x; \theta^{(t)}_{LIR}).$$

Therefore a LIR step that grants full control to q drives q to the exact posterior under the current θ. This is precisely the *E-step* of EM.

Full control of p (M-step): Now fix q (in particular, after the E-step we have $q(z) = p(z \mid x; \theta^{(t)}_{LIR})$). Minimizing $\langle\!\langle \mathcal{M}_{q,\theta} \rangle\!\rangle$ over p (equivalently, maximizing the ELBO over θ) is

$$\theta^{(t+2)}_{LIR} \in \arg\max_{\theta} \mathop{\mathbb{E}}_{q(z)} \big[\log p(x,z;\theta) \big] = \arg\max_{\theta} \mathop{\mathbb{E}}_{z \sim p(z|x;\theta^{(t)}_{LIR})} \big[\log p(x,z;\theta) \big],$$

which is exactly the standard *M-step* update.

Identification with EM iterates: Let $\theta^{(0)}_{\mathrm{LIR}} = \theta^{(0)}_{\mathrm{EM}}$. A LIR cycle consists of two refocus operations: (i) full control of $q \Rightarrow q^{(t+1)} = p(z \mid x; \theta^{(t)}_{\mathrm{LIR}})$ (E-step), (ii) full control of $p \Rightarrow \theta^{(t+2)}_{\mathrm{LIR}} \in \arg\max_\theta \mathbb{E}_{q^{(t+1)}}[\log p(x,z;\theta^{(t)}_{LIR})]$ (M-step). The second update matches the EM M-step with posterior taken at $\theta^{(t)}_{\mathrm{EM}}$. Hence, by induction on t, $\theta^{(t)}_{\mathrm{EM}} = \theta^{(2t)}_{\mathrm{LIR}}$.

The optional Refocus re-sampling of x simply yields the usual (mini-)batched variant without affecting the maximizers of the conditional objectives. This proves that alternating full control between q and p in LIR implements EM as claimed. □

Proposition 3. *LIR($\mathcal{M}$) with foci alternating between (φ_G, χ_G) and (φ_D, χ_D) trains a GAN.*

Proof. Let $C \in \{\text{real}, \text{fake}\}$ denote the coin (real vs. fake) and let X be the displayed image. In the PDG in the text we have high-confidence arcs enforcing

$$C \sim \text{Bernoulli}\big(\tfrac{1}{2}\big), \qquad (X \mid C{=}\text{real}) \sim p_{\text{data}}, \quad (X \mid C{=}\text{fake}) \sim G,$$

so the unique μ^* attaining the infimum in the observational inconsistency is the induced joint $\mu^*(x, c) = \frac{1}{2}\, p_{\text{data}}(x)\, \mathbb{1}\{c{=}\text{real}\} + \frac{1}{2}\, G(x)\, \mathbb{1}\{c{=}\text{fake}\}$. Write $D(c \mid x)$ for the discriminator cpd and $e(c \mid x) \equiv \frac{1}{2}$ for the "equal coin" belief. The inconsistency of the PDG is obtained as follows.

$$\begin{aligned}
&\langle\!\langle \mathcal{M} \rangle\!\rangle = OInc_{\mathcal{M}}(\mu^\star) \\
&= \beta_D \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\Big[\log \mu^\star(C \mid X) - \log D(C \mid X)\Big] + \beta_e \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\Big[\log \mu^\star(C \mid X) - \log e(C \mid X)\Big] \\
&= -\beta_D \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log D(C \mid X)\big] - \beta_e \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log e(C \mid X)\big] + (\beta_D + \beta_e) \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log \mu^\star(C \mid X)\big] \\
&= -\beta_D\Big(\tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim p_{\text{data}}} \log D(x) + \tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim G} \log(1 - D(x))\Big) - \beta_e \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log e(C \mid X)\big] + (\beta_D + \beta_e) \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log \mu^\star(C \mid X)\big] \\
&= -\beta_D \mathcal{L}^{\text{GAN}}(G, D) - \beta_e \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log e(C \mid X)\big] + (\beta_D + \beta_e) \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log \mu^\star(C \mid X)\big] \\
&= -\beta_D \mathcal{L}^{\text{GAN}}(G, D) - \beta_e \log \tfrac{1}{2} + (\beta_D + \beta_e) \mathop{\mathbb{E}}_{(X,C)\sim\mu^\star}\big[\log \mu^\star(C \mid X)\big] \\
&= -\beta_D \mathcal{L}^{\text{GAN}}(G, D) - \beta_e \log \tfrac{1}{2} + (\beta_D + \beta_e)\Big(\tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim p_{\text{data}}} \log \mu^\star(\text{real} \mid x) + \tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim G} \log \mu^\star(\text{fake} \mid x))\Big) \\
&= -\beta_D \mathcal{L}^{\text{GAN}}(G, D) - \beta_e \log \tfrac{1}{2} + (\beta_D + \beta_e)\Big(\tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim p_{\text{data}}} \log \tfrac{p_{\text{data}}(x)}{p_{\text{data}}(x)+G(x)} + \tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim G} \log \tfrac{G(x)}{p_{\text{data}}(x)+G(x)})\Big) \\
&= -\beta_D \mathcal{L}^{\text{GAN}}(G, D) + \beta_e \log 2 + (\beta_D + \beta_e)\left(\left(\tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim p_{\text{data}}} \log \tfrac{p_{\text{data}}(x)}{m_G(x)} + \tfrac{1}{2} \mathop{\mathbb{E}}_{x\sim G} \log \tfrac{G(x)}{m_G(x)}\right) - \log 2\right) \\
&\qquad\qquad (\text{with } m_G(x) = \tfrac{1}{2} p_{\text{data}}(x) + \tfrac{1}{2} G(x)) \\
&= -\beta_D \mathcal{L}^{\text{GAN}}(G, D) \;+\; (\beta_D + \beta_e)\text{JS}(p_{\text{data}}, G) \;-\; \beta_D \log 2,
\end{aligned}$$

where $\text{JS}(p, q)$ is the Jensen-Shannon divergence between p and q.

Discriminator focus (φ_D, χ_D)**.** By construction, χ_D gives control only over D, and φ_D attends to the D arc with weight $\beta_D = 1$ while *disbelieving* the e arc, *i.e.*, $\beta_e = -1$. The inconsistency simplifies to $\langle\!\langle \mathcal{M} \rangle\!\rangle = -\mathcal{L}^{\text{GAN}}(G, D) - \log 2$. Thus, minimizing inconsistency over D is equivalent to the usual $\max_D \mathcal{L}^{\text{GAN}}(G, D)$ step.

Generator focus (φ_G, χ_G)**.** By construction, χ_G gives control only over G, and φ_G *disbelieves* the D arc ($\beta_D = -1$) while attending to the e arc with $\beta_e = 1$. The inconsistency simplifies to $\langle\!\langle \mathcal{M} \rangle\!\rangle = \mathcal{L}^{\text{GAN}}(G, D) + \log 2$. Thus minimizing inconsistency over G is equivalent to the usual $\min_G \mathcal{L}^{\text{GAN}}(G, D)$ step.

Therefore, alternating LIR steps that refocus between (φ_D, χ_D) and (φ_G, χ_G) performs exactly the standard min-max optimization for GANs. □

Proposition 4. *If* Refocus *selects a focus non-deterministically from $\{a{\to}X, X{\to}a, X\}_{X\in\mathcal{X}, a\in\partial X}$ (see illustration above; details in Appendix B), then the possible runs of* LIR($\mathcal{M}_\Phi, \mathcal{M}_{\mathcal{BG}} + \mathcal{B}$) *are precisely those of BP for different message schedules.*

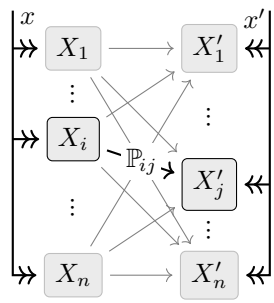
Proof. When $\gamma = 1$, and $\alpha, \beta = 1$ for all of the input factors, then the optimal distribution μ^* that realizes the infimum is just the product of factors. It follows that any distribution that has those marginals will minimize the observational inconsistency.

Different interleavings of the update equations (6) and (7) for different adjacent pairs (a, X), which correspond to the different possible message passing schedules, also correspond immediately to different selections from this fixed set of foci in the LIR procedure. □

Proposition 5. *LIR($\mathcal{M}$) in which attention is set to $\varphi_{ij} = \exp\langle k_i, q_j\rangle$, when controlling x', leads to inference in the transformer layer,* i.e., $x'_j = \sum_i \alpha_{i|j} v_i$.

Proof. Recall the parametric PDG in question:

$\mathcal{M}(W_V, W_Q, W_K, x, x') :=$



,

where $\mathbb{P}_{ij}(X'_j \mid X_i) = \mathcal{N}(X'_j \mid W_V X_i, I)$ is a unit normal distribution. For completeness, and to aid the discussion, we will start with an analysis of the more general case where $\mathbb{P}_{ij}(X'_j \mid X_i) = \mathcal{N}(X'_j \mid W_V X_i, \Sigma_{ij})$ has an arbitrary covariance.

Because of the deterministic arrows x and x', the infimum defining the inconsistency must be attained at the degenerate distribution $\mu^*(X, X')$ where all mass is supported on the joint assignment (x, x').

We can therefore calculate

$$
\begin{aligned}
&\Big\langle\!\Big\langle \mathcal{M}(W_Q, W_K, W_V, x, x') \Big\rangle\!\Big\rangle \\
&= \sum_{i,j\in[n]} \varphi_{ij} \boldsymbol{D}\Big(\delta_{x'_j}(X'_j) \,\Big\|\, \mathcal{N}(X'_j \mid W_V x_i,\, \Sigma_{ij})\Big) \\
&= \sum_{i,j\in[n]} -\varphi_{ij} \log\left[(2\pi)^{-d/2} \det(\Sigma_{ij})^{-1/2} \exp\Big(-\frac{1}{2}(x'_j - v_i)^\intercal \Sigma_{ij}^{-1}(x'_j - v_i)\Big)\right] \qquad \text{(see Appendix C)} \\
&= \sum_{i,j\in[n]} \frac{\varphi_{ij}}{2}\Big[\log((2\pi)^d \det \Sigma_{ij}) + (x'_j - v_i)^\intercal \Sigma_{ij}^{-1}(x'_j - v_i)\Big],
\end{aligned}
$$

where we have defined $v_i := W_V x_i$, as in the main text.

What happens when we control x'?

The equation above is (strongly) convex in x', and so gradient descent moves to the unique local minimum. The

directional derivative of $\langle\!\langle \mathcal{M} \rangle\!\rangle$ with respect to x' in direction $u = (u_j)_{j=1}^n$ is given by

$$
\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\epsilon} \sum_{i,j\in[n]} \frac{\varphi_{ij}}{2} (u_j\epsilon + x'_j - v_i)^\top \Sigma_{ij}^{-1} (u_j\epsilon + x'_j - v_i) \Big|_{\epsilon=0} \\
&= \sum_{i,j\in[n]} \frac{\varphi_{ij}}{2} \Bigg(\frac{\mathrm{d}}{\mathrm{d}\epsilon} \Big[\epsilon^2 u_j^\top \Sigma_{ij}^{-1} u_j\Big]_{\epsilon=0} + \frac{\mathrm{d}}{\mathrm{d}\epsilon} \Big[\epsilon u_j^\top \Sigma_{ij}^{-1}(x'_j - v_i) + \epsilon (x'_j - v_i)^\top \Sigma_{ij}^{-1} u_j\Big]_{\epsilon=0} \\
&\qquad\qquad + \frac{\mathrm{d}}{\mathrm{d}\epsilon} \Big[(x'_j - v_i)^\top \Sigma_{ij}^{-1} (x'_j - v_i)\Big] \Bigg) \\
&= \sum_{i,j\in[n]} \frac{\varphi_{ij}}{2} u_j^\top (\Sigma_{ij}^{-1} + \Sigma_{ij}^{-\top})(x'_j - v_i) \\
&= \sum_{i,j\in[n]} \varphi_{ij} u_j^\top \Sigma_{ij}^{-1} (x'_j - v_i) && \text{since } \Sigma_{ij}^{-1} \text{ is symmetric} \\
&= \sum_{j\in[n]} u_j^\top \sum_{i\in[n]} \varphi_{ij} \Sigma_{ij}^{-1} (x'_j - v_i).
\end{aligned}
$$

The unique attractor x'_* of the ODE defined by these gradients is characterized by having zero directional gradients for all $\vec{u}$, which is the case iff

$$
\begin{aligned}
& \forall j.\ \sum_i \varphi_{ij} \Sigma_{ij}^{-1} (x'_j - v_i) = 0 \\
\iff\quad & \forall j.\ \Big(\sum_{i\in[n]} \varphi_{ij} \Sigma_{ij}^{-1}\Big) x'_j = \sum_{i\in[n]} \varphi_{ij} \Sigma_{ij}^{-1} v_i \\
\iff\quad & \forall j.\ x'_j = \Big(\sum_{i\in[n]} \varphi_{ij} \Sigma_{ij}^{-1}\Big)^{-1} \sum_{i\in[n]} \varphi_{ij} \Sigma_{ij}^{-1} v_i. \qquad \text{provided the matrix on the left is invertible.}
\end{aligned}
$$

In the special case where $\Sigma_{ij} = \sigma_{ij}^2 I$, and let $\psi_{ij} := \varphi_{ij}/\sigma_{ij}^2$, the optimum is

$$
x'_j = \sum_{i\in[n]} \Big(\frac{\psi_{ij}}{\sum_{i'=1}^n \psi_{i'j}}\Big) v_i.
$$

Therefore, whenever it is the case that $\psi_{ij} = b \exp\langle k_i, q_j\rangle$ for some constant b, it is also the case that the unique fixed point of LIR when controlling x' leads to $x'_j = \sum_i \alpha_{i|j} v_i$ where $\alpha_{i|j} = \mathrm{softmax}_i[\langle k_i, q_j\rangle]_{i,j} = \Big(\frac{\exp\langle k_i, q_j\rangle}{\sum_{i'=1}^n \exp\langle k_{i'}, q_j\rangle}\Big)$.

In particular, this is the case when $\varphi_{ij} = \exp\langle k_i, q_j\rangle$ and $\sigma_{ij} = 1$, as assumed in the theorem statement. □

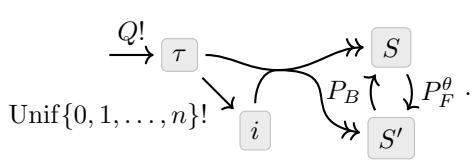
Proposition 6. *The inconsistency of the PDG $\mathcal{M}$ from (8) with centered surprisal-based attention φ is*

$$
\langle\!\langle \varphi \odot \mathcal{M} \rangle\!\rangle = \mathop{\mathbb{E}}_{\tau\sim Q} \left[\frac{1}{|\tau|} \log^2 \frac{P_F(\tau) Z}{R(x) P_B(\tau \mid x)} \right] =: \mathcal{L}_{\mathrm{ModTB}}.
$$

Furthermore, the gradients of $\mathcal{L}_{\mathrm{ModTB}}$ differ only by a factor of two if φ is regarded as fixed and not a function of θ. Therefore, LIR on $\mathcal{M}$ with attention φ and control of P_F (and optionally P_B) amounts to training a GFlowNet with "re-weighted" TB loss.

Proof. Recall the PDG in question:

$\mathcal{M}(\theta) =$



Let $\hat{Q}(\tau, i) := Q(\tau)\,\mathrm{Unif}(i \mid \tau)$ denote the distribution over trajectories and indices obtained by first drawing a trajectory $\tau \sim Q$ and then a uniformly random index $i \in \{0, \ldots, n\}$ along the (random) trajectory $\tau = (s_0, \ldots, s_n{=}x, s_{n+1}{=}s_f)$. Because the labels on $Q(\tau)$ and $\mathrm{Unif}(i \mid \tau)$ are high confidence, the infimum in the observational inconsistency is attained by the induced joint

$$\mu^*(\tau, i, S, S') \;=\; Q(\tau)\,\mathrm{Unif}(i \mid \tau)\;\delta(S = \tau[i])\;\delta(S' = \tau[i{+}1]).$$

Subclaim. For all $s \in \mathcal{S} \setminus \{s_0, s_f\}$ we have $\mu^*(S{=}s) = \mu^*(S'{=}s)$; also, $\mu^*(S{=}s_0) = \mu^*(S'{=}s_f)$.

Proof. Since the zeroth element $\tau[0]$ of a trajectory τ is always s_0, the assumption that $S = \tau[i] = s \notin \{s_0, s_f\}$ necessitates $i > 0$. Therefore,

$$\begin{aligned}
\mu^*(S = s) &= \hat{Q}(\{(\tau, i) : \tau[i] = s\}) \\
&= \hat{Q}(\{(\tau, i-1) : \tau[i] = s\}) && \text{by uniformity of } i\text{, as } i > 0 \\
&= \hat{Q}(\{(\tau, j) : \tau[j+1] = s\}) && \text{by taking } j := i - 1 \\
&= \mu^*(S' = s).
\end{aligned}$$

Summing over s and using $\mu^*(S{=}s_f) = \mu^*(S'{=}s_0) = 0$ yields $\mu^*(S{=}s_0) = \mu^*(S'{=}s_f)$. □

Only the two cpds $P_F(\cdot \mid \cdot)$ and $P_B(\cdot \mid \cdot)$ contribute non-constant terms (all copy/selection hyperarcs are satisfied by μ^*). Denote

$$a(\tau) := \log P_F(\tau) = \sum_{i=0}^{n} \log P_F(s_{i+1} \mid s_i), \qquad b(\tau) := \log P_B(\tau) = \sum_{i=0}^{n} \log P_B(s_i \mid s_{i+1}).$$

By construction at the sink,

$$P_B(\tau) \;=\; P_B(\tau \mid x)\,\frac{R(x)}{Z} \quad \Longrightarrow \quad b(\tau) = \log P_B(\tau \mid x) + \log R(x) - \log Z.$$

The observational incompatibility (with centered surprisal attention) is

$$\begin{aligned}
\mathit{OInc}_{\mathcal{M}}(\mu^*) \;=\; \mathop{\mathbb{E}}_{(\tau,i)\sim\hat{Q}} \Big[& \overbrace{\varphi(P_F)}^{=\,b(\tau)-a(\tau)} \cdot \overbrace{\boldsymbol{D}(\delta_{S'=\tau[i+1]} \parallel P_F(\,\cdot \mid S = \tau[i]))}^{=\,-\log P_F(\tau[i+1]|\tau[i])} \\
& + \underbrace{\varphi(P_B)}_{=\,a(\tau)-b(\tau)} \cdot \underbrace{\boldsymbol{D}(\delta_{S=\tau[i]} \parallel P_B(\,\cdot \mid S' = \tau[i+1]))}_{=\,-\log P_B(\tau[i]|\tau[i+1])} \Big].
\end{aligned}$$

Since the attention φ depends only on τ, we can factor it out of the expectation over i, which is uniform over the $|\tau|$ transition sin the trajectory, and expectation amounts to their sum divided by $|\tau|$: just sums the local logs back to the whole-trajectory logs:

$$\mathit{OInc}_{\mathcal{M}}(\mu^*) \;=\; \mathop{\mathbb{E}}_{\tau\sim Q}\Big[\big(b(\tau) - a(\tau)\big)\,\big(\frac{-a(\tau)}{|\tau|}\big) \;+\; \big(a(\tau) - b(\tau)\big)\,\big(\frac{-b(\tau)}{|\tau|}\big)\Big].$$

Expanding,

$$\mathit{OInc}_{\mathcal{M}}(\mu^*) \;=\; \mathop{\mathbb{E}}_{\tau\sim Q}\Big[\frac{a(\tau)^2 + b(\tau)^2 - 2\,a(\tau)b(\tau)}{|\tau|}\Big] \;=\; \mathop{\mathbb{E}}_{\tau\sim Q}\Big[\frac{1}{|\tau|}\big(a(\tau) - b(\tau)\big)^2\Big].$$

Using the sink identity for $b(\tau)$, we obtain

$$a(\tau) - b(\tau) \;=\; \log \frac{P_F(\tau)\,Z}{R(x)\,P_B(\tau \mid x)}.$$

Therefore

$$\mathit{OInc}_{\mathcal{M}}(\mu^*) \;=\; \mathop{\mathbb{E}}_{\tau\sim Q}\left[\frac{1}{|\tau|}\log^2 \frac{P_F(\tau)\,Z}{R(x)\,P_B(\tau \mid x)}\right] \;=\; \mathcal{L}_{\mathrm{ModTB}}(Q),$$

which proves that the (centered-surprisal) attention yields inconsistency equal to the modified TB objective.

Gradients. Treat φ as a fixed (stop-gradient) attention. If only P_F depends on θ,

$$\nabla_\theta OInc_{\mathcal{M}} \;=\; 2 \mathop{\mathbb{E}}_{\tau\sim Q}\left[\log \frac{P_F(\tau)\,Z}{R(x)\,P_B(\tau \mid x)}\,\nabla_\theta \log P_F(\tau)\right],$$

which is exactly $\nabla_\theta \mathcal{L}_{\text{TB}}(Q)$. If, in addition, P_B shares the same parameters (or we also place control on P_B), an equal-magnitude term appears with opposite sign in $\nabla_\theta \log P_B(\tau \mid x)$, yielding the same direction and a proportionality factor (a constant multiple), as claimed. Hence LIR on $\mathcal{M}$ with the stated attention and control over P_F (and optionally P_B) trains a GFlowNet with the TB loss. □

Proposition 7. *Let $\mathcal{M}$ be the PDG in (9) together with the cpd $b(\texttt{T}{=}\texttt{t} \mid U{=}u) = k \cdot \exp(u)$ for some $k > 0$.*

(a) If $\beta_p = \infty$ and $\beta_b < \infty$, then the action(s) $a \in \mathcal{V}A$ that minimize the inconsistency are those that maximize expected utility. Formally, for all $\gamma \geq 0$,

$$\arg\min_{a\in\mathcal{V}A} \langle\!\langle \mathcal{M} + A{=}a \rangle\!\rangle_\gamma = \arg\max_{a\in\mathcal{V}A} \mathop{\mathbb{E}}_{\substack{s\sim p\\ o\sim\tau|s,a}} \big[u(o)\big].$$

(b) If $\beta_p < \infty$ and $\beta_b = \infty$, then the action(s) $a \in \mathcal{V}A$ that minimize overall inconsistency are those that can lead to the best possible outcome, i.e.*,*

$$\arg\min_{a\in\mathcal{V}A} \langle\!\langle \mathcal{M} + A{=}a \rangle\!\rangle_\gamma = \arg\max_{a\in\mathcal{V}A} \max_{s\in\mathcal{V}S} \mathop{\mathbb{E}}_{o\sim\tau|s,a} [u(o)].$$

Proof. Since the choice a is deterministic, the value of A is determined; likewise the value of $\texttt{T}$ is also fixed. Similarly, as u is a deterministic function, the value of U is determined according to u. Thus we need only consider distributions $\mu(S, O)$ in our minimization; the other variables can be found as a function of these. However, we also know that $\mu(O \mid S) = \tau(O|S, A{=}a)$ since it is given with high confidence. Therefore it suffices to restrict our search to distributions over the variable S.

To simplify notation, let $EU(s, a) := \mathbb{E}_{o\sim\tau(O|s,a)}[u(o)] + \log k$ be the expected utility of an action, shifted by the constant $\log k$. Note that

$$\log \frac{1}{b(\texttt{T}{=}\texttt{t} \mid U{=}u(o))} = -\log(k \cdot \exp(u(o))) = -u(o) - \log k,$$

which in expectation over $\tau(O|s, a)$, is $-EU(s, a)$. With this in mind, we calculate:

$$\begin{aligned}
\langle\!\langle \mathcal{M}_{p,\tau,u,b} + A{=}a \rangle\!\rangle_\gamma &= \inf_{\mu(S)} \beta_p \mathop{\mathbb{E}}_{s\sim\mu}\Big[\log\frac{\mu(s)}{p(s)} + \frac{\beta_b}{\beta_p} \mathop{\mathbb{E}}_{o\sim\tau|a,s}\Big[\log\frac{1}{b(\texttt{T}{=}\texttt{t} \mid U{=}u(o))}\Big]\\
&= \inf_{\mu(S)} \beta_p \mathop{\mathbb{E}}_{s\sim\mu}\Big[\log\frac{\mu(s)}{p(s)} + \log\circ\exp\Big(-\frac{\beta_b}{\beta_p}EU(s,a)\Big)\\
&= \inf_{\mu(S)} \beta_p \mathop{\mathbb{E}}_{s\sim\mu}\Big[\log\frac{\mu(s)}{1}\cdot\frac{\exp(-\frac{\beta_b}{\beta_p}EU(s,a))}{p(s)}\cdot\frac{Z}{Z}\Big]\\
&= \inf_{\mu(S)} \beta_p \mathop{\mathbb{E}}_{s\sim\mu}\Big[\log\frac{\mu(s)}{1}\cdot\frac{Z}{p(s)\cdot\exp(\frac{\beta_b}{\beta_p}EU(s,a))}\cdot\frac{1}{Z}\Big]
\end{aligned} \tag{19}$$

At this point, we can use the same trick used repeatedly in the proving results of Richardson (2022): take Z to be the normalization constant needed to regard the middle fraction as the inverse of a probability distribution ν. Once we do so, we are left with an infimum over a KL divergence $\boldsymbol{D}(\mu \parallel \nu)$ plus the expectation of a constant:

$$\langle\!\langle \mathcal{M}_{p,\tau,u,b} + A{=}a \rangle\!\rangle_\gamma = \beta_p \log\frac{1}{Z} = -\log\sum_s p(s)\exp\Big(+\frac{\beta_b}{\beta_p}EU(s,a)\Big).$$

We now look at the two extreme cases.

When $\beta_p \to \infty$, then the ratio $\frac{\beta_b}{\beta_p}$ becomes small, and we can use the fact that $\exp(\epsilon) \approx 1 + \epsilon$ for small ϵ to find that the inconsistency of interest is approximately

$$\approx -\beta_p \log \sum_s p(s)\Big[1 + \frac{\beta_b}{\beta_p} EU(s,a)\Big] = -\beta_p \log(1 + \frac{\beta_b}{\beta_p} EU(s,a)) \approx -\beta_b EU(s,a)$$

Alternatively, more directly, when $\beta_p = \infty$, the optimal distribution must be $\mu = p$, and so the inconsistency is immediately $-\mathbb{E}_{s\sim p}[\beta_b EU(s,a)]$. Either way, minimizing this quantity amounts to maximizing expected utility, as the two differ by a negative affine transformation.

At the other extreme, when $\beta_b \to \infty$, we can write our expression in terms of $\mathrm{LSE}\{x_1, \ldots, x_n\} := \log \sum_{i=1}^n \exp(x_i)$ (LogSumExp) which is a smooth approximation to a max. (In a moment, we will negate its arguments and the final output, using it as an approximation to a min, instead.) Picking back up from (19) and letting $t := \frac{\beta_b}{\beta_p}$, we find

$$\langle\!\langle \mathcal{M}_{p,\tau,u,b} + A{=}a \rangle\!\rangle_\gamma = -\beta_b \cdot \frac{-1}{t} \operatorname*{LSE}_{s\in\mathcal{V}S}\Big[-t \cdot \Big(\frac{1}{t}\log\frac{1}{p(s)} - EU(s,a)\Big)\Big].$$

Using the standard fact[4] that

$$\min_{i\in[n]} x_i - \frac{1}{t}\log n \le \frac{-1}{t}\operatorname*{LSE}_{i\in[n]}(-tx_i) < \min_{i\in[n]} x_i,$$

we find that, in our case,

$$M - \frac{1}{t}\log|\mathcal{V}S| \quad\le\quad \frac{1}{\beta_b}\langle\!\langle \mathcal{M}_{p,\tau,u,b} + A{=}a \rangle\!\rangle_\gamma \quad\le\quad M \tag{20}$$

where $M := \min_{s\in\mathcal{V}S}(-EU(s,a) + \frac{1}{t}\log\frac{1}{p(s)})$. In particular, when $\beta_b \to \infty$, meaning $t \to \infty$, the gap between the upper and lower bounds shrinks to zero, and the resulting inconsistency becomes $-\min_{s\in\mathcal{V}S}(-EU(s,a)) = \max_{s\in\mathcal{V}S)} EU(s,a)$, proving the result. □

E Generative Flow Network Experiments

E.1 Trajectory-Length Normalization for the Trajectory Balance and Log-Partition Variance Losses

In Trajectory Balance (TB) and Log-Partition Variance (LPV, also called VarGrad) objectives, longer trajectories accumulate larger log-probability sums, causing per-trajectory squared losses to scale with trajectory length. This can starve gradients on shorter trajectories and bias learning toward long-trajectory corrections. Their losses (reproduced below) were found to be missing a normalization constant n_i, denoting the length of the trajectory τ_i.

The standard trajectory balance (TB) condition states that for all complete trajectories $\tau = (s_0 \to s_1 \to \cdots \to x)$, $Z\prod_t P_F(s_{t+1} \mid s_t) = R(x)\prod_t P_B(s_t \mid s_{t+1})$, or equivalently,

$$\log Z + \log P_F(\tau) \;=\; \log R(x) + \log P_B(\tau \mid x).$$

which gives the loss

$$\mathcal{L}_{\mathrm{TB}}(Q) = \mathop{\mathbb{E}}_{\tau\sim Q}\left[\log^2 \frac{P_F(\tau)Z}{R(x)P_B(\tau \mid x)}\right].$$

When trained as a GFlowNet, the forward policy P_F and log partition function estimate $\log Z$ are parameterized separately, optionally alongside the backward policy P_B (which may not be parameterized as it is possible to leave it fixed during training), and $R(x)$ is a reward function providing some scalar reward defining a distribution over the state space $\mathcal{X}$.

[4] Letting $m := \min_i x_i$, observe that, for all $t > 0$, we have $\exp(-tm) \le \sum_i \exp(-tx_i) \le n\exp(-tm)$. Apply a logarithm and multiply by $-\frac{1}{t}$ to get the promised result.

In contrast, the Log Partition Variance loss avoids explicit estimation of $\log Z$ by noting that it is constant with respect to τ. Given the per-trajectory residual,

$$\Delta(\tau) \;=\; \log P_F(\tau) - \log R(x) - \log P_B(\tau \mid x).$$

$\Delta(\tau) = -\log Z$ when the TB condition holds exactly, so $\log Z$ can treated as a constant shift and dropped from the variance:

$$\mathcal{L}_{\text{LPV}}(Q) \;=\; \text{Var}_{\tau\sim Q}\left[\log \frac{P_F(\tau)}{R(x)\,P_B(\tau \mid x)}\right]$$

Expanding the variance via $\text{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$:

$$\mathcal{L}_{\text{LPV}}(Q) \;=\; \mathop{\mathbb{E}}_{\tau\sim Q}\left[\left(\log \frac{P_F(\tau)}{R(x)\,P_B(\tau \mid x)}\right)^2\right] \;-\; \left(\mathop{\mathbb{E}}_{\tau\sim Q}\left[\log \frac{P_F(\tau)}{R(x)\,P_B(\tau \mid x)}\right]\right)^2.$$

In practice, all expectations are estimated by the sample mean over a minibatch $\{\tau_i\}_{i=1}^m$ of trajectories sampled from the forward policy P_F. We normalize the per-trajectory squared loss by dividing by per-trajectory length n_i after squaring, preserving the fixed point ($s_i + \log Z = 0$) and preventing an attenuation of gradients that would arise from normalizing the score before squaring. For TB, the modified loss is simply:

$$\mathcal{L}_{\text{ModTB}} := \frac{1}{m}\sum_{i=1}^{m} \frac{s_i^2}{n_i}, \qquad \text{where} \;\; s_i = \log \frac{P_F(\tau_i)\,Z}{R(x_i)\,P_B(\tau_i \mid x_i)}, \tag{21}$$

and for LogPV, the modification is analogous. Let $s_i = \log \frac{P_F(\tau_i)}{R(x_i)\,P_B(\tau_i|x_i)}$ for trajectories $\tau_i \sim Q$,

$$\mathcal{L}_{\text{ModLPV}} = \frac{1}{m}\sum_{i=1}^{m} \frac{(s_i - \bar{s})^2}{n_i}, \quad \text{where} \quad \bar{s} = \frac{1}{m}\sum_{i=1}^{n} s_i \quad \text{and} \quad s_i = \log \frac{P_F(\tau_i)}{R(x_i)\,P_B(\tau_i \mid x_i)} \tag{22}$$

where $T_i = |\tau_i|$ is the length of trajectory i. This is no longer $\text{Var}_Q[s]$; the $\frac{1}{T_i}$ weighting downweights long trajectories, analogous to SubTB-style per-step normalization (Madan et al., 2023).

E.2 Experimental Setup

E.2.1 Environment

All losses were tested on four variants of the discrete HyperGrid task: a d-dimensional grid of height H where trajectories can terminate in any state and all trajectories start at $\mathbf{0}_d$. For all experiments, we evaluate on a 4-dimensional HyperGrid with height $H = 24$, giving a state space of $24^4 = 331{,}776$ terminal states (states are vectors $s \in \{0, 1, \ldots, 23\}^4$). We define a normalized coordinate $a_i = |s_i/(H-1) - 0.5|$ for each dimension i.

Original: The original hypergrid environments follows (Bengio et al., 2021):

$$R(s) = R_0 + R_1 \prod_i \mathbf{1}[a_i > 0.25] + R_2 \prod_i \mathbf{1}[0.3 < a_i < 0.4]$$

with $R_0 = 0.1$, $R_1 = 0.5$, $R_2 = 2.0$. Intuitively, modes occupy the 4 corners of each dimension, giving $4^4 = \mathbf{256}$ mode states. Modes are states satisfying the inner band condition in all dimensions simultaneously (reward $\geq R_0 + R_1 + R_2 = 2.6$). At $H = 24$, the band indices per dimension are $\{3, 4, 19, 20\}$.

Cosine: A smooth, oscillatory reward with per-dimension factors:

$$R(s) = R_0 + R_1 \prod_i f(a_i), \quad f(a) = (\cos(50a) + 1) \cdot \mathcal{N}(0,1)(5a)$$

where $\mathcal{N}(0,1)(\cdot)$ is the standard normal PDF. With $R_0 = 0.1$, $R_1 = 0.5$, and closeness parameter $\gamma = 0.8$, mode states are those with reward $\geq R_0 + (\gamma \cdot f_{\max})^4 \cdot R_1$, where $f_{\max} \approx 0.581$ is the discrete per-dimension maximum at $H = 24$. The product structure yields **1,280** mode states with a regular pattern across dimensions.

Bitwise XOR: A tiered compositional reward based on GF(2) parity constraints:

$$R(s) = \sum_{t=0}^{2} w_t \cdot \mathbf{1}\left[\forall\, b \in [l_t, h_t],\ \bigoplus_{i=0}^{3} \mathrm{bit}_b(s_i) = 0\right]$$

with tier weights $w = (1, 10, 100)$ and bit ranges $(l_t, h_t) \in \{(0,5), (0,7), (0,9)\}$. Each tier requires even parity (XOR $= 0$) across all 4 dimensions at every bit plane in its range. Rewards are cumulative: tier t is awarded only if all constraints through tier t are satisfied. Modes achieve the highest tier (reward $= 111$). Since values ≤ 23 have at most 5 nonzero bits, the higher bit-plane constraints are automatically satisfied, and mode membership reduces to $s_0 \oplus s_1 \oplus s_2 \oplus s_3 = 0$ (bitwise). The fourth coordinate is determined ($s_3 = s_0 \oplus s_1 \oplus s_2$) subject to $s_3 < 24$, giving **10,752** mode states.

Multiplicative Coprime: A tiered reward based on prime factorization constraints:

$$R(s) = \sum_{t=0}^{2} w_t \cdot \mathbf{1}\left[\forall\, i,\ s_i = \prod_{p \in \mathcal{P}} p^{e_{i,p}} \text{ with } e_{i,p} \leq c_t\right]$$

with $w = (1, 10, 100)$, prime set $\mathcal{P} = \{2, 3, 5\}$, and uniform exponent cap $c_t = 2$ for all tiers. Each dimension's value must factor completely over $\mathcal{P}$ with exponents ≤ 2 (zero values are excluded; $s_i = 1$ is always valid). The valid values per dimension are $\{1, 2, 3, 4, 5, 6, 9, 10, 12, 15, 18, 20\}$ (12 values), yielding $12^4 =$ **20,736** mode states.

E.3 Training

Hyperparameter Tuning: To fairly compare each loss, we first used the Optuna (Akiba et al., 2019) TPE sampler to tune:

- Learning rate: $[10^{-5}, 10^{-1}]$ (log-scale)
- Adam $\beta_2 \in \{0.99, 0.999, 0.9999\}$
- Gradient clipping norm: $[0.01, 10.0]$ (log-scale)
- LogZ learning-rate multiplier (TB variants only): $[10, 1000]$ (log-scale)

The LogZ learning-rate multiplier is commonly used to assist in the convergence of TB-based algorithms which cannot converge until the $logZ$ estimate is close to the true partition function, which may be very large for some distributions. To accomplish this, practitioners often train $logZ$ with an independent optimizer and/or learning rate. In our case the multiplier is applied to whatever base learning rate is proposed by Optuna for that run. For all experiments, batch size=1024, we employed a replay buffer with capacity=10,000 and 50% replay fraction, a per-trajectory loss clamp of 100, linear LR schedule down to 1% of the initial learning rate, and the AdamW optimizer with weight decay of 0.01. We ran 50 trials for each of the 16 algorithm-environment pairs, each run for 2,000 iterations with a single seed, optimizing final L1 distance estimated between the true reward distribution and the learned posterior using 10^6 samples from the trained P_F.

Model Training: The top thee hyperparameter configurations found were brought forward, and trained across 5 seeds for 4,000 iterations each (240 runs total). The following metrics were collected:

- **L1 distance** between the learned terminal-state distribution and the true reward-proportional distribution: $\sum_s |p(s) - p^*(s)|$
- **Jensen–Shannon divergence (JSD)**: $\frac{1}{2}\mathrm{KL}(p\|m) + \frac{1}{2}\mathrm{KL}(p^*\|m)$, where $m = \frac{1}{2}(p + p^*)$
- **Mode coverage**: fraction of reward modes discovered during training

E.4 Results

See Table 3 for the median ± interquartile range metrics for all losses on all environments using the single best hyperparameters identified across 5 seeds. All models (with some single-seed exceptions) cover all environment modes. However, two methods that have 100% mode coverage may have substantially different L1 / JSD distances. L1 & JSD measures are highly correlated ($r > 0.95\%$). On the Bitwise XOR environment, the modified losses show clear benefits as measured using the distribution distance measures. ModLPV also shows faster convergence at the beginning of training relative to the LPV variant, as measured by the fraction of iterations required to reach 2× the final loss. Differences appear to be environment dependent, and the advantages of the improved training dynamics of the modified losses may be even more apparent on more sophisticated and challenging reward distributions.

Table 3: Final metrics for rank-1 configurations (median ± IQR across 5 seeds).

Environment	**TB**	**ModTB**	**LPV**	**ModLPV**
		Final L1 Distance		
Bitwise XOR	1.74 ± 0.28	1.23 ± 0.01	1.49 ± 0.51	1.24 ± 0.00
Cosine	0.449 ± 0.000	0.448 ± 0.000	0.448 ± 0.000	0.448 ± 0.000
Multiplicative Coprime	0.137 ± 0.003	0.137 ± 0.003	0.134 ± 0.001	0.138 ± 0.002
Original	0.438 ± 0.000	0.438 ± 0.000	0.44 ± 1.54	0.439 ± 0.001
		Final JSD		
Bitwise XOR	0.487 ± 0.111	0.289 ± 0.006	0.388 ± 0.182	0.288 ± 0.001
Cosine	0.050 ± 0.000	0.050 ± 0.000	0.050 ± 0.000	0.050 ± 0.000
Multiplicative Coprime	0.010 ± 0.001	0.010 ± 0.000	0.008 ± 0.000	0.010 ± 0.001
Original	0.052 ± 0.000	0.052 ± 0.000	0.052 ± 0.620	0.052 ± 0.000
		Mode Coverage (%)		
Bitwise XOR	100.00 ± 0.00	99.27 ± 0.06	100.00 ± 0.00	99.45 ± 0.29
Cosine	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Multiplicative Coprime	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Original	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 96.09	100.00 ± 0.00
		Iterations to 2× final loss (fraction, ↓)		
Bitwise XOR	0.006 ± 0.003	0.030 ± 0.023	0.004 ± 0.002	0.045 ± 0.009
Cosine	0.835 ± 0.054	0.857 ± 0.058	0.794 ± 0.011	0.521 ± 0.055
Multiplicative Coprime	0.199 ± 0.356	0.455 ± 0.099	0.574 ± 0.044	0.248 ± 0.109
Original	0.838 ± 0.007	0.895 ± 0.013	0.857 ± 0.858	0.586 ± 0.023