Inference for Probabilistic Dependency Graphs

Oliver E. Richardson[1] **Joseph Y. Halpern**[1] **Christopher De Sa**[1]

[1]Deparment of Computer Science, Cornell University, Ithaca NY 14853

Abstract

Probabilistic dependency graphs (PDGs) are a flexible class of probabilistic graphical models, subsuming Bayesian Networks and Factor Graphs. They can also capture inconsistent beliefs, and provide a way of measuring the degree of this inconsistency. We present the first tractable inference algorithm for PDGs with discrete variables, making the asymptotic complexity of PDG inference similar that of the graphical models they generalize. The key components are: (1) the observation that PDG inference can be reduced to convex optimization with exponential cone constraints, (2) a construction that allows us to express these problems compactly for PDGs of boundeed treewidth, for which we needed to further develop the theory of PDGs, and (3) an appeal to interior point methods that can solve such problems in polynomial time. We verify the correctness and time complexity of our approach, and provide an implementation of it. We then evaluate our implementation, and demonstrate that it outperforms baseline approaches. Our code is available at `github.com/orichardson/pdg-infer-uai`.

1 INTRODUCTION

Probabilistic dependency graphs (PDGs) [Richardson and Halpern, 2021], form a very general class of probabilistic graphical models, that includes not only Bayesian Networks (BNs) and Factor Graphs (FGs), but also more recent statistical models built out of neural networks, such as Variational Autoencoders (VAEs) [Kingma and Welling, 2014]. PDGs can also capture inconsistent beliefs, and provide a useful way to measure the degree of this inconsistency; for a VAE, this is the loss function used in training [Richardson, 2022]. PDGs have some significant advantages over other representations of probabilistic information. Their flexibility allows them to model beliefs that BNs cannot, such as information from independent studies of the same variable (perhaps with different controls, yielding probabilistic observations $p(Y|X)$ and $q(Y|Z)$). PDGs can deal gracefully with conflicting information from multiple sources. Every subcomponent of a PDG has probabilistic meaning, independent of the other components; compared to FGs, this makes PDGs more interpretable. But up to now, there has been no practical way to do inference for PDGs—that is, to answer questions of the form "what is the probability of Y given X?" This paper presents the first algorithm to do so.

Before discussing our algorithm, we must discuss what it even means to do inference for a PDG. A BN or FG represents a unique joint distribution. Thus, for example, when we ask "what is the probability of Y given that $X{=}x$?" in a BN, we mean "what is $\mu(Y|X{=}x)$?" for the probability measure μ that the BN represents. But a PDG might, in general, represent more than just one distribution.

Like a BN, a PDG encodes two types of information: "structural" information about the independence of causal mechanisms, and "observational" information about conditional probabilities. Unlike in a BN, the two can conflict in a PDG. Corresponding to these two types of information, a PDG has two loss functions, which quantify how far a distribution μ is from modeling the information of each type. Given a number $\hat\gamma \in [0,1]$ indicating the importance of structure relative to observation, we take the $\hat\gamma$*-semantics* of a PDG to be the set of distributions that minimize the appropriate convex combination of losses. We also consider the 0^+*-semantics*: the limiting case that arises as $\hat\gamma$ goes to zero (which focuses on observation, using structure only to break ties). This set can be shown to contain precisely one distribution for PDGs satisfying a mild regularity condition (required by definition by Richardson and Halpern); we call such PDGs *proper*. Thus, we have a parameterized family of inference notions: to do $\hat\gamma$-inference, for $\hat\gamma \in [0,1] \cup \{0^+\}$, is to answer queries in a way that is true of all distributions in the $\hat\gamma$-semantics.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

If there are distributions fully consistent with both the observational and the structural information in a PDG $\mathcal{M}$, then for $\hat{\gamma} \in (0,1) \cup \{0^+\}$, all notions of $\hat{\gamma}$-inference coincide. If $\mathcal{M}$ is also proper, this means there is a single distribution $\mu_{\mathcal{M}}$ that minimizes both loss functions, in which case we want to answer queries with respect to $\mu_{\mathcal{M}}$ no matter how we weight observational and structural information. Moreover, if $\mathcal{M}$ represents a BN, then $\mu_{\mathcal{M}}$ is the distribution represented by the BN. However, if there is no distribution that is consistent with both types of information, then the choice of $\hat{\gamma}$ matters.

Since PDGs subsume BNs, and inference for BNs is already NP-hard, the same must be true of PDGs. At a high level, the best we could hope for would be tractability on the restricted class of models on which inference has traditionally been tractable—that is, a polynomial algorithm for models whose underlying structure has *bounded treewidth* (see Section 2 for formal definitions). That is indeed what we have. More precisely, we show that 0^+-inference and $\hat{\gamma}$-inference for small $\hat{\gamma}$ can be done for discrete PDGs of bounded treewidth containing N variables in $\tilde{O}(N^4)$ time.

Our algorithm is based on a line of recent work in convex programming that establishes polynomial-time for a class of optimization problems called *exponential conic programs* [Badenbroek and Dahl, 2021, Skajaa and Ye, 2015, Nesterov et al., 1999]. Our contribution is to show that the problem of inference in a PDG of bounded treewidth can be efficiently converted to a (sequence of) exponential conic program(s), at which point it can be solved with a commercial solver (e.g., ApS [2022]) in polynomial time. The direct appeal to a solver allows us to benefit from the speed and reliability of such highly optimized solvers, and also from future improvements in exponential conic optimization. Thus, our result is not only a theoretical one, but practical as well.

Beyond its role as a probabilistic model, a PDG is also of interest for its degree of inconsistency—that is, the minimum value of its loss function. As shown by Richardson [2022], many loss functions and statistical divergences can be viewed as measuring the inconsistency of a PDG that models the context appropriately. This makes calculating this minimum value of interest—but up to now, there has been no way to do so. There is a deep connection between this problem and PDG inference; for now, we remark that this number is a byproduct of our techniques.

Contributions. We provide the first algorithm for inference in a PDG; in addition, it calculates a PDG's degree of inconsistency. We prove that our algorithm is correct, and also fixed-parameter tractable: for PDGs of bounded treewidth, it runs in polynomial time. We also prove that PDG inference and inconsistency calculation are equivalent problems. Our algorithm reduces inference in PDGs to exponential conic programming in a way that can be offloaded to powerful existing solvers. We provide an implementation of this reduction in a standard convex optimization framework, giving users an interface between such solvers and the standard PDG Python library. Finally, we evaluate our implementation. The results suggest our method is faster and significantly more reliable than simple baseline approaches.

2 PRELIMINARIES & RELATED WORK

Vector Notation. For us, a vector is a map from a finite set S, called its *shape*, to the extended reals $\bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. We write $\mathbf{u} := [u_i]_{i \in S}$ to define a vector $\mathbf{u}$ by its components. Vectors of the same shape can be added (+), partially ordered ($\leq$), or multiplied ($\odot$) pointwise as usual. $\mathbf{1}$ denotes an all-ones vector, of a shape implied by context.

Probabilities. We write ΔS to denote the set of probability distributions over a finite set S. Every variable X can take on values from a finite set $\mathcal{V}X$ of possible values. We can regard sets of variables $\mathbf{X}$ as variables themselves, with $\mathcal{V}\mathbf{X} = \Pi_{X \in \mathbf{X}} \mathcal{V}X$. A conditional probability distribution (cpd) $p(Y|X)$ is a map $p : \mathcal{V}X \to \Delta\mathcal{V}Y$ assigning to each $x \in \mathcal{V}X$ a probability distribution $p(Y|x) \in \Delta\mathcal{V}Y$, which is shorthand for $p(Y|X{=}x)$. Given a distribution μ over (the values of) a set of variables including X and Y, we write $\mu(X)$ for its marginal on X, and $\mu(Y|X)$ for the cpd obtained by conditioning on X and marginalizing to Y. We also refer to μ's entropy $\mathrm{H}(\mu) := \mathbb{E}_\mu[\log \frac{1}{\mu}]$ and conditional entropy $\mathrm{H}_\mu(Y|X) := \mathbb{E}_\mu[\log {}^1/_{\mu(Y|X)}]$ of Y given X.

Hypergraphs and Treewidth. A hypergraph $(V, \mathcal{E})$ is a set V of vertices and a set $\mathcal{E}$ of *hyperedges*, which correspond to subsets of V. An ordinary graph may be viewed as the special case in which every hyperedge contains two vertices.

Definition 1. A *directed hypergraph* $(N, \mathcal{A})$ is a set of nodes N, and a set of *(hyper)arcs* $\mathcal{A}$, each $a \in \mathcal{A}$ of which is associated with a set of source nodes $S_a \subseteq N$, and target nodes $T_a \subseteq N$. We also write $S \xrightarrow{a} T \in \mathcal{A}$ to specify an arc a together with its sources $S = S_a$ and targets $T = T_a$. □

A directed hypergraph can be viewed as a hypergraph by joining each source and target set, thereby "forgetting" the direction of the arrow. Thus, notions defined for undirected hypergraphs (like that of treewidth, which we now review), can be applied to directed hypergraphs as well.

Many problems that are intractable for general graphs are tractable for trees, and some graphs are closer to being trees than others. A tree decomposition of a (hyper)graph $G = (V, \mathcal{E})$ is a tree $(\mathcal{C}, \mathcal{T})$ whose vertices $C \in \mathcal{C}$, called *clusters*, are subsets of V such that:

1. every vertex $v \in V$ and every hyperedge $E \in \mathcal{E}$ is contained in at least one cluster, and
2. every cluster D along the unique path from C_1 to C_2 in $\mathcal{T}$, contains $C_1 \cap C_2$.

The *width* of a tree decomposition is one less than the size of its largest cluster, and the *treewidth* of a (hyper)graph G

is the smallest possible width of any tree decomposition of G. It is NP-hard to determine the tree-width of a graph, but if the tree-width is known to be bounded above, a tree decomposition may be constructed in linear time [Bodlaender, 1993]. For graphs of bounded tree-width, many problems (indeed, any problem expressible in a certain second-order logic [Courcelle, 1990]) can be solved in linear time. This is also true of inference in standard graphical models.

Graphical Models and Inference. A *graphical model structure* is a (directed) (hyper)graph whose vertices $\mathcal{X}$ are variables, and whose (hyper)edges somehow indicate local influences between variables. A *probabilistic graphical model*, or simply "graphical model", is a graphical model structure together with quantitative information about these local influences. Semantically, a graphical model $\mathcal{M}$ typically represents a joint probability distribution $\Pr_{\mathcal{M}} \in \Delta\mathcal{V}\mathcal{X}$ over its variables. Inference for $\mathcal{M}$ is then the ability to calculate cpds $\Pr_{\mathcal{M}}(Y|X{=}x)$, where $X, Y \subset \mathcal{X}$ and $x \in \mathcal{V}X$.

Many inference algorithms (such as belief propagation), when applied to tree-like graphical models, run in linear time and are provably correct. If the same algorithms are naïvely applied to graphs with cycles (as in loopy belief propagation), then they may not converge, and even if they do, may give incorrect (or even inconsistent) answers [Wainwright et al., 2008]. Nearly all exact inference algorithms (including variable elimination, clique calibration, and message-passing with and without division), effectively construct a tree decomposition, and can be viewed as running on a tree [Koller and Friedman, 2009, §9-11]. Indeed, under widely believed assumptions, every class of graphical models for which inference is *not* NP-hard has bounded treewidth [Chandrasekaran et al., 2012]. Given a tree decomposition $(\mathcal{C}, \mathcal{T})$ of the underlying model structure, many of these algorithms use a data structure known as a *clique tree*, which is a collection $\boldsymbol{\mu} {=} \{\mu_C(C)\}_{C \in \mathcal{C}}$ of probabilities over the clusters [Koller and Friedman, 2009, §10].

A clique tree $\boldsymbol{\mu}$ is said to be *calibrated* if neighboring clusters' distributions agree on the variables they share. In this case, it determines a joint distribution by

$$\Pr_{\boldsymbol{\mu}}(\mathcal{X}) = \prod_{C \in \mathcal{C}} \mu_C(C) \Big/ \prod_{(C-D) \in \mathcal{T}} \mu_C(C \cap D), \qquad (1)$$

which has the property that $\Pr_{\boldsymbol{\mu}}(C) = \mu_C$ for all $C \in \mathcal{C}$. A calibrated clique tree summarizes the answers to queries about $\Pr_{\boldsymbol{\mu}}$ [see Koller and Friedman, 2009, §10.3.3]. Therefore, to answer probabilistic queries with respect to a distribution μ, it suffices to find a calibrated clique tree $\boldsymbol{\mu}$ that represents μ, and appeal to standard algorithms.

Probabilistic Dependency Graphs. Our presentation of PDGs is slightly different from (but equivalent to) that of Richardson and Halpern [2021], which the reader is encouraged to consult for more details and intuition. At a high level, a PDG is just a collection of cpds and causal assertions, weighted by confidence. More precisely:

Definition 2. A PDG $\boldsymbol{m} = (\mathcal{X}, \mathcal{A}, \mathbb{P}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is a directed hypergraph $(\mathcal{X}, \mathcal{A})$ whose nodes are variables, together with probabilities $\mathbb{P}$ and confidence vectors $\boldsymbol{\alpha} = [\alpha_a]_{a \in \mathcal{A}}$ and $\boldsymbol{\beta} {=} [\beta_a]_{a \in \mathcal{A}}$, so that each $S \xrightarrow{a} T \in \mathcal{A}$ is associated with:

- a conditional probability distribution $\mathbb{P}_a(T|S)$ on the target variables given values of the source variables,
- a weight $\beta_a \in \overline{\mathbb{R}}$ indicating the modeler's confidence in the cpd $\mathbb{P}_a(T|S)$, and
- a weight $\alpha_a \in \mathbb{R}$ indicating the modeler's confidence in the functional dependence of T on S expressed by a.

If $\boldsymbol{\beta} \geq \mathbf{0}$ and $\alpha_a > 0$ implies $\beta_a > 0$, we write $\boldsymbol{\beta} \gg \boldsymbol{\alpha}$ and call $\boldsymbol{m}$ *proper*. Note that $\boldsymbol{\beta} \gg \boldsymbol{\alpha}$ if $\boldsymbol{\beta} > \mathbf{0}$. □

One significant advantage of PDGs is their modularity: we can combine the information in $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ by taking the union of their variables and the disjoint union of their arcs (and associated data) to get a new PDG, denoted $\boldsymbol{m}_1 + \boldsymbol{m}_2$.

As mentioned in the introduction, a PDG contains two types of information: "structural" information, in the hypergraph $\mathcal{A}$ and weights $\boldsymbol{\alpha}$, and "observational" data, in the cpds $\mathbb{P}$ and weights $\boldsymbol{\beta}$. PDG semantics are based on two scoring functions that quantify discrepancy between each type of information and a distribution $\mu \in \Delta\mathcal{V}\mathcal{X}$ over its variables.

The *observational incompatibility* of μ with $\boldsymbol{m}$, which can be thought of as a "distance" between μ and the cpds of $\boldsymbol{m}$, is given by the weighted sum of relative entropies:

$$\mathit{OInc}_{\boldsymbol{m}}(\mu) := \sum_{S \xrightarrow{a} T \in \mathcal{A}} \beta_a \boldsymbol{D}\Big(\mu(T,S) \,\Big\|\, \mathbb{P}_a(T|S)\mu(S)\Big).$$

Under a standard interpretation of the relative entropy $\boldsymbol{D}(\mu \parallel p) {=} \mathbb{E}_\mu[\log \frac{\mu}{p}]$, $\mathit{OInc}_{\boldsymbol{m}}$ measures the excess cost of using codes optimized for the cpds of $\boldsymbol{m}$ (weighted by their confidences), when reality is distributed according to μ.

The second scoring function measures the extent to which μ is incompatible with a causal picture consisting of independent mechanisms along each hyperarc. This is captured by the *structural incompatibility* (of μ with $\boldsymbol{m}$), and given by

$$\mathit{SInc}_{\boldsymbol{m}}(\mu) := \Big(\sum_{S \xrightarrow{a} T \in \mathcal{A}} \alpha_a \, \mathrm{H}_\mu(T|S)\Big) - \mathrm{H}(\mu).$$

Note that it does not depend on the cpds of $\boldsymbol{m}$, nor the possible values of the variables; it is defined purely in terms of the weighted hypergraph structure $(\mathcal{A}, \boldsymbol{\alpha})$.

When the observational and structural information conflict, then the distribution(s) that best represent a PDG will depend on the relative importance of observation and structure. This is captured by a trade-off paramter $\hat{\gamma} \in [0,1]$, which we can use to form the convex combination $(1-\hat{\gamma})\,\mathit{OInc} + \hat{\gamma}\,\mathit{SInc}$. So as to simplify the math and match the notation in previous work (2021, 2022), we instead use a rescaled variant with a different parameterization. Let $\gamma := \hat{\gamma}/(1-\hat{\gamma}) \in [0,\infty]$, which is almost identical to $\hat{\gamma}$ when $\hat{\gamma} \approx 0$. With it, define the combined scoring function:

$$\begin{aligned} [\![\boldsymbol{m}]\!]_\gamma(\mu) &:= \mathit{OInc}_{\boldsymbol{m}}(\mu) + \gamma\, \mathit{SInc}_{\boldsymbol{m}}(\mu) \qquad (2) \\ &= \frac{1}{1-\hat{\gamma}}\Big((1-\hat{\gamma})\, \mathit{OInc}_{\boldsymbol{m}}(\mu) + \hat{\gamma}\, \mathit{SInc}_{\boldsymbol{m}}(\mu) \Big) \\ &= \mathbb{E}_\mu \Bigg[\sum_{S \xrightarrow{a} T \in \mathcal{A}} \log \frac{\mu(T|S)^{\beta_a - \gamma\alpha_a}}{\mathbb{P}_a(T|S)^{\beta_a}} \Bigg] - \gamma\, \mathrm{H}(\mu). \end{aligned}$$

Let $[\![\boldsymbol{m}]\!]^*_\gamma := \arg\min_\mu [\![\boldsymbol{m}]\!]_\gamma(\mu)$ denote the set of optimal distributions at a particular value γ. One natural conception of inference in PDGs is then parameterized by $\hat{\gamma}$: to do $\hat{\gamma}$-inference in $\boldsymbol{m}$ is to respond to probabilistic queries in a way that is sound with respect to every $\mu \in [\![\boldsymbol{m}]\!]^*_\gamma$. It is not too difficult to see that when $\boldsymbol{\beta} \geq \gamma\boldsymbol{\alpha}$, (2) is strictly convex, which ensures that $[\![\boldsymbol{m}]\!]^*_\gamma$ is a singleton. This paper demonstrates that $\hat{\gamma}$-inference is tractable for such PDGs.

The limiting behavior of the $\hat{\gamma}$-semantics as $\hat{\gamma} \to 0$, which we denote $[\![\boldsymbol{m}]\!]^*_{0+}$ and call the 0^+-*semantics*, has some special properties. If $\boldsymbol{m}$ is proper, then $[\![\boldsymbol{m}]\!]^*_{0+}$ contains precisely one distribution. This distribution intuitively reflects an extreme empirical view: observational data trumps causal structure. Note that in the absence of a causal picture ($\boldsymbol{\alpha} = \mathbf{0}$), this corresponds to the well-established practice of selecting the maximum entropy distribution consistent with some observational constraints [Jaynes, 1957]. One should be careful to distinguish $[\![\boldsymbol{m}]\!]^*_{0+}$ from $[\![\boldsymbol{m}]\!]^*_0$, the set of distributions that minimize $\mathit{OInc}_{\boldsymbol{m}}$; the latter set includes $[\![\boldsymbol{m}]\!]^*_{0+}$ [Richardson and Halpern, 2021, Prop 3.4], but may also contain other distributions. This paper also shows how to efficiently answer queries with respect to the unique distribution in $[\![\boldsymbol{m}]\!]^*_{0+}$, which we call 0^+-*inference*.

Given a PDG $\boldsymbol{m}$, the smallest possible value of its scoring function, $\langle\!\langle \boldsymbol{m} \rangle\!\rangle_\gamma := \inf_\mu [\![\boldsymbol{m}]\!]_\gamma(\mu)$, is known as its γ-inconsistency and is interesting in its own right: $\langle\!\langle \cdot \rangle\!\rangle_\gamma$ seems to be a "universal" loss function [Richardson, 2022].

Interior-Point Methods and Convex Optimization. Interior-point methods provide an iterative way of approximately solving linear programs in polynomial time [Karmarkar, 1984]. With the theory of "symmetric cones", these methods were extended in the 1990s to handle second-order cone programs (SOCPs) and semidefinite programs (SDPs), which allow more expressive constraints. But the constraints that these methods can handle are insufficient for our purposes. We need what have been called *exponential cone constraints*. The *exponential cone* is the convex set

$$\begin{aligned} K_{\text{exp}} := &\big\{(x_1,x_2,x_3) : x_1 \geq x_2 e^{x_3/x_2},\, x_2 > 0\big\} \\ &\cup \big\{(x_1,0,x_3) : x_1 \geq 0,\, x_3 \leq 0\big\} \qquad \subset \overline{\mathbb{R}}^3. \end{aligned}$$

Let $K := K^p_{\text{exp}} \times [0,\infty]^q \subset \overline{\mathbb{R}}^n$ be a product of p exponential cones and $q = n - 3k$ non-negative orthants. An *exponential conic program* is then an optimization problem of the form

$$\underset{\mathbf{x}}{\mathsf{minimize}} \quad \mathbf{c}^{\mathsf{T}}\mathbf{x} \quad \mathsf{subject\ to} \quad A\mathbf{x} = \mathbf{b}, \;\; \mathbf{x} \in K, \qquad (3)$$

where $\mathbf{c} \in \overline{\mathbb{R}}^n$ is some cost vector, the function $\mathbf{x} \mapsto \mathbf{c}^{\mathsf{T}}\mathbf{x}$ is called the *objective*, and $\mathbf{b} \in \overline{\mathbb{R}}^m$, $\mathbf{A} \in \overline{\mathbb{R}}^{m\times n}$ encode linear constraints. Nesterov, Todd, and Ye [1999] first established that such problems can be solved in polynomial time, but incur double the memory and eight times the time, compared to the symmetric counterparts. These drawbacks were eliminated in Skajaa and Ye [2015]. The algorithm that seems to display the best empirical performance [Dahl and Andersen, 2022], however, was only recently shown to run in polynomial time [Badenbroek and Dahl, 2021].

Disciplined Convex Programming [Grant, 2004] is a compositional approach to convex optimization that imposes certain restrictions on how problems can be formed; a program conforming to those rules is said to be dcp. The reason to do so is that a dcp program can be efficiently compiled to a standard form [Agrawal et al., 2018], which in our case is an exponential conic program. Only two rules are relevant to us: a constraint of the form $(x,y,z) \in K_{\text{exp}}$ is dcp iff x, y, and z are affine transformations of the optimization variables, and a linear program augmented with dcp exponential conic constraints is dcp. Because all the optimization problems that we give are of this form, we can easily compile them to exponential conic programs even if they do not exactly conform to (3).

3 INFERENCE AS A CONVEX PROGRAM

Here is an obvious, if inefficient, way of calculating $\Pr_{\mathcal{M}}(Y|X{=}x)$ in a probabilistic model $\mathcal{M}$. First compute an explicit representation of the joint distribution $\Pr_{\mathcal{M}} \in \Delta\mathcal{V}\mathcal{X}$, then marginalize to $\Pr_{\mathcal{M}}(X,Y)$ and condition on $X{=}x$. For a factor graph or BN, each step is straightforward; the problem is the exponential time and space required to represent $\Pr_{\mathcal{M}}(\mathcal{X})$ explicitly. A key feature of inference algorithms for BNs and FGs is that they do not represent joint distributions in this way. For PDGs, though, it is not obvious that we can calculate the $\hat{\gamma}$-semantics, even if we know it is unique, and we ignore the space required to represent it (as we do in this section). Note that $\hat{\gamma}$-inference is already an optimization problem by definition:

$$\underset{\mu}{\mathsf{minimize}} \quad [\![\boldsymbol{m}]\!]_\gamma(\mu) \quad \mathsf{subject\ to} \quad \mu \in \Delta\mathcal{V}\mathcal{X}.$$

For small enough γ, it is even convex. But can we solve it efficiently? With exponential cone constraints, the answer is yes, as we show in Section 3.2. Moreover, we can compute the 0^+-semantics with a sequence of two exponential conic programs (Section 3.3). To give a flavor of our constructions and ease into the more complicated ones, we begin by minimizing *OInc*, the simpler of the two scoring functions.

3.1 MINIMIZING INCOMPATIBILTY ($\gamma = 0$)

When $\gamma = 0$, we want to find minimizers of *OInc*, which is a weighted sum of conditional relative entropies. There is a straightforward connection between the exponential cone and relative entropy: if $\mathbf{m}, \mathbf{p} \in \Delta\{1, \dots, n\} \subset \mathbb{R}^n$ are points on a probability simplex, then $(-\mathbf{u}, \mathbf{m}, \mathbf{p}) \in K^n_{\text{exp}}$ if and only if $\mathbf{u}$ is an upper bound on $\mathbf{m} \log \frac{\mathbf{m}}{\mathbf{p}}$, the pointwise contribution to relative entropy at each outcome. Thus, perhaps unsurprisingly, we can use an exponential conic program to find minimizers of *OInc*. If all beliefs are unconditional and over the same space, the construction is standard; we review it here, so that we can build upon it.

Warm-up. Consider a PDG with only one variable X with $\mathcal{V}X = \{1, \dots, n\}$. Suppose further that every arc $j \in \mathcal{A} = \{1, \dots, k\}$ has $T_j = \{X\}$ and $S_j = \emptyset$. Then each $\mathbb{P}_j(X)$ can be identified with a vector $\mathbf{p}_j \in [0,1]^n$, and all k of them can conjoined to form a matrix $\mathbf{P} = [\,p_{ij}\,] \in [0,1]^{n \times k}$. Similarly, a candidate distribution μ can be identified with $\mathbf{m} \in [0,1]^n$. Now consider a matrix $\mathbf{U} = [u_{i,j}] \in \overline{\mathbb{R}}^{n \times k}$ that, intuitively, gives an upper bound on the contribution to *OInc* due to each edge and value of X. Observe that

$$\begin{aligned}
& (-\mathbf{U},\ [\mathbf{m}, \dots, \mathbf{m}],\ \mathbf{P}) \in K^{n \times k}_{\text{exp}} \\
\iff \quad & \forall i,j.\ \ u_{ij} \geq m_i \log(m_i/p_{ij}) \\
\implies \quad & \forall j.\ \textstyle\sum_i u_{ij} \geq \boldsymbol{D}(\mu \parallel p_j) \\
\implies \quad & \textstyle\sum_{i,j} \beta_j u_{ij} \geq \sum_j \beta_j \boldsymbol{D}(\mu \parallel p_j) \\
\iff \quad & \mathbf{1}^{\mathsf{T}} \mathbf{U} \boldsymbol{\beta} \geq \mathit{OInc}(\mu).
\end{aligned} \tag{4}$$

So now, if $(\mathbf{U}, \mathbf{m})$ is a solution to the convex program

$$\underset{\mathbf{m},\mathbf{U}}{\text{minimize}} \quad \mathbf{1}^{\mathsf{T}} \mathbf{U} \boldsymbol{\beta} \quad \text{subject to} \quad \mathbf{1}^{\mathsf{T}} \mathbf{m} = 1, \quad (-\mathbf{U}, [\mathbf{m}, \dots, \mathbf{m}],\ \mathbf{P}) \in K^{n \times k}_{\text{exp}},$$

then (a) the objective value $\mathbf{1}^{\mathsf{T}} \mathbf{U} \boldsymbol{\beta}$ equals the inconsistency $\langle\!\langle \mathcal{M} \rangle\!\rangle_0$, and (b) $\mu \in [\![\mathcal{M}]\!]^*_0$, meaning μ minimizes $\mathit{OInc}_{\mathcal{M}}$.

The General Case. We now show how the same construction can be used to find a distribution $\mu \in [\![\mathcal{M}]\!]^*_0$ for an arbitrary PDG $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P}, \boldsymbol{\alpha}, \boldsymbol{\beta})$. To further simplify the presentation, for each arc $a \in \mathcal{A}$, let $\mathcal{V}a := \mathcal{V}(S_a, T_a)$ denote all joint settings of a's source and target variables, and write $\mathcal{VA} := \sqcup_{a \in A} \mathcal{V}a = \{(a, s, t) : a \in \mathcal{A},\ (s,t) \in \mathcal{V}(S_a, T_a)\}$ for the set of all choices of an arc together with values of its source and target. For each $a \in \mathcal{A}$, we can regard $\mu(T_a, S_a)$ and $\mu(S_a)\mathbb{P}_a(T_a|S_a)$, both distributions over $\{S_a, T_a\}$, as vectors of shape $\mathcal{V}a$. As before, we introduce an optimization variable $\mathbf{u}$ that packages together all of the relevant pointwise upper bounds. To that end, consider a vector $\mathbf{u} = [u_{a,s,t}] \in \overline{\mathbb{R}}^{\mathcal{VA}}$ in the optimization problem

$$\begin{aligned}
& \underset{\mu,\mathbf{u}}{\text{minimize}} \quad \sum_{(a,s,t) \in \mathcal{VA}} \beta_a\, u_{a,s,t} \\
& \text{subject to} \quad \mu \in \Delta\mathcal{VX}, \\
& \forall a \in \mathcal{A}.\ \big(-\mathbf{u}_a,\ \mu(T_a, S_a),\ \mathbb{P}_a(T_a|S_a)\mu(S_a) \big) \in K^{\mathcal{V}a}_{\text{exp}}.
\end{aligned} \tag{5}$$

where $\mathbf{u}_a = [u_{a,s,t}]_{(s,t) \in \mathcal{V}a}$ consists of those components of $\mathbf{u}$ associated with arc a. Note that the marginals $\mu(S_a, T_a)$ and $\mu(S_a)$ are affine transformations of μ, so (5) is dcp. A straightforward generalization of the logic in (4) gives us:

Proposition 1. *If $(\mu, \mathbf{u})$ is a solution to (5), then $\mu \in [\![\mathcal{M}]\!]^*_0$, and $\sum_{(a,s,t) \in \mathcal{VA}} \beta_a u_{a,s,t} = \langle\!\langle \mathcal{M} \rangle\!\rangle_0$.*

Thus, a solution to (5) encodes a distribution that minimizes *OInc*, and the (0-)inconsistency of $\mathcal{M}$. This is a start, but to do 0^+-inference, among the minimizers of *OInc* we must find the unique distribution in $[\![\mathcal{M}]\!]^*_{0^+}$, while for $\hat{\gamma}$-inference ($\hat{\gamma} > 0$), we need to find the optimizers of $[\![\mathcal{M}]\!]^*_\gamma$. Either way, we must consider *SInc* in addition to *OInc*.

3.2 γ-INFERENCE FOR SMALL $\gamma > 0$

When $\gamma > 0$ is small enough, the scoring function (2) is not only convex, but admits a straightforward representation as an exponential conic program. To see this, note that (2) can be rewritten [Richardson and Halpern, 2021, Prop 4.6] as:

$$\begin{aligned}
[\![\mathcal{M}]\!]_\gamma(\mu) = &-\gamma\, \mathrm{H}(\mu) - \sum_{a \in \mathcal{A}} \beta_a \mathop{\mathbb{E}}_{\mu} \log \mathbb{P}_a(T_a|S_a) \\
&+ \sum_{a \in \mathcal{A}} (\gamma\alpha_a - \beta_a)\, \mathrm{H}_\mu(T_a|S_a).
\end{aligned} \tag{6}$$

The first term, $-\gamma\,\mathrm{H}(\mu)$, is strictly convex and has a well-known translation into an exponential cone constraint; the second one linear in μ. Now, if $0 < \gamma \leq \min_a \frac{\beta_a}{\alpha_a}$, then every summand of the last term is a negative conditional entropy, and can be captured by an exponential cone constraint. The only wrinkle is that it is possible for a user to specify that some $\mathbb{P}_a(t \mid s) = 0$, in which case the linear term is undefined. The result is a requirement that $\mu(s,t) = 0$ at such points, which we can instead encode directly with linear constraints. To do this formally, divide $\mathcal{VA}$ into two parts: $\mathcal{VA}^+ := \{(a,s,t) \in \mathcal{VA} : \mathbb{P}_a(t|s) > 0\}$ and $\mathcal{VA}^0 := \{(a,s,t) \in \mathcal{VA} : \mathbb{P}_a(t|s) = 0\}$. Armed with this notation, consider upper bound vectors $\mathbf{u} = [u_{a,s,t}]_{(a,s,t) \in \mathcal{VA}}$ and $\mathbf{v} = [v_w]_{w \in \mathcal{VX}}$, in the following optimization problem:

$$\begin{aligned}
\underset{\mu,\mathbf{u},\mathbf{v}}{\text{minimize}} \quad & \sum_{(a,s,t) \in \mathcal{VA}} (\beta_a - \alpha_a\gamma) u_{a,s,t} + \gamma \sum_{w \in \mathcal{VX}} v_w \\
& - \sum_{(a,s,t) \in \mathcal{VA}^+} \alpha_a \gamma\, \mu(S_a{=}s, T_a{=}t) \log \mathbb{P}_a(t|s) \\
\text{subject to} \quad & \mu \in \Delta\mathcal{VX}, \quad (-\mathbf{v}, \mu, \mathbf{1}) \in K^{\mathcal{VX}}_{\text{exp}}, \\
& \forall a \in \mathcal{A}.\ \big(-\mathbf{u}_a, \mu(T_a, S_a), \mathbb{P}_a(T_a|S_a)\mu(S_a) \big) \in K^{\mathcal{V}a}_{\text{exp}}, \\
& \forall (a,s,t) \in \mathcal{VA}^0.\ \mu(S_a{=}s, T_a{=}t) = 0.
\end{aligned} \tag{7}$$

This optimization problem may look complex, but it falls out of (6) fairly directly, and gives us what we wanted.

Proposition 2. *If $(\mu, \mathbf{u}, \mathbf{v})$ is a solution to (7), and $\boldsymbol{\beta} \geq \gamma\boldsymbol{\alpha}$, then μ is the unique element of $[\![\mathcal{M}]\!]^*_\gamma$, and $\langle\!\langle \mathcal{M} \rangle\!\rangle_\gamma$ equals the objective of (7) evaluated at $(\mu, \mathbf{u}, \mathbf{v})$.*

3.3 CALCULATING THE 0^+-SEMANTICS ($\gamma \to 0$)

Section 3.1 shows how to find a distribution ν that minimizes *OInc*—but to do 0^+-inference, we need to find the minimizer that, uniquely among them, best minimizes *SInc*. It turns out this can be done by using ν to construct a second optimization problem. The justification requires two more results; we start by characterizing the minimizers of *OInc*.

Proposition 3. *If $\mathcal{M}$ has arcs $\mathcal{A}$ and $\boldsymbol{\beta} \geq 0$, the minimizers of $\mathit{OInc}_{\mathcal{M}}$ all have the same conditional marginals along $\mathcal{A}$. That is, for all $\mu_1, \mu_2 \in \llbracket \mathcal{M} \rrbracket_0^*$ and all $S \xrightarrow{a} T \in \mathcal{A}$ with $\beta_a > 0$, we have $\mu_1(T,S)\mu_2(S) = \mu_2(T,S)\mu_1(S)$.*[1]

As a result, once we find one minimizer ν of $\mathit{OInc}_{\mathcal{M}}$ (e.g., via (5)), it then suffices to constrain distributions that have the same conditional marginals along the edges, and optimize *SInc*. But in attepting to do so, we run into a second issue: *SInc* is typically not convex. Fortunately, it is if we constrain to distributions that minimize *OInc*. Moreover, on this restricted domain, it can be represented with dcp exponential cone constraints.

Proposition 4. *If $\mu \in \llbracket \mathcal{M} \rrbracket_0^*$, then*

$$\underset{\mathcal{M}}{\mathit{SInc}}(\mu) = \sum_{w \in \mathcal{V}\mathcal{X}} \mu(w) \log \left({\mu(w)} \Big/ {\prod_{a \in \mathcal{A}} \nu\big(T_a(w) \big| S_a(w)\big)^{\alpha_a}} \right), \qquad (8)$$

where $\{\nu(T_a|S_a)\}_{a \in \mathcal{A}}$ are the marginals along the arcs $\mathcal{A}$ shared by all distributions in $\llbracket \mathcal{M} \rrbracket_0^*$ (per Proposition 3), and $S_a(w), T_a(w)$ are the values of variables S_a and T_a in w.

If we already know a distribution $\nu \in \llbracket \mathcal{M} \rrbracket_0^*$, perhaps by solving (5), then the denominator of (8) does not depend on μ and so is constant in our search for minimizers of *SInc*. For ease of exposition, aggregate these values into a vector

$$\mathbf{k} := \Big[\prod_{a \in \mathcal{A}} \nu(T_a(w)|S_a(w))^{\alpha_a} \Big]_{w \in \mathcal{V}\mathcal{X}} . \qquad (9)$$

We can now capture $\llbracket \mathcal{M} \rrbracket_{0+}^*$ with a convex program.

Proposition 5. *If $\nu \in \llbracket \mathcal{M} \rrbracket_0^*$ and $(\mu, \mathbf{u})$ solves the problem*

$$\begin{aligned} &\underset{\mu,\mathbf{u}}{\text{minimize}} \quad \mathbf{1}^{\mathsf{T}}\mathbf{u} && (10)\\ &\text{subject to} \quad (-\mathbf{u}, \mu, \mathbf{k}) \in K_{\exp}^{\mathcal{V}\mathcal{X}}, \qquad \mu \in \Delta\mathcal{V}\mathcal{X}, \\ &\qquad \forall S \xrightarrow{a} T \in \mathcal{A}.\ \ \mu(S,T)\,\nu(S) = \mu(S)\,\nu(S,T), \end{aligned}$$

then $\llbracket \mathcal{M} \rrbracket_{0+}^* = \{\mu\}$ and $\mathbf{1}^{\mathsf{T}}\mathbf{u} = \mathit{SInc}_{\mathcal{M}}(\mu)$.

Running (10) through a convex solver gives rise to the first algorithm that can reliably find $\llbracket \mathcal{M} \rrbracket_{0+}^*$.

4 POLYNOMIAL-TIME INFERENCE UNDER BOUNDED TREEWIDTH

We have now seen how $\hat{\gamma}$-inference (for small $\hat{\gamma}$) can be reduced to convex optimization over joint distributions μ—but μ grows exponentially with the number of variables in the PDG, so we do not yet have a tractable inference algorithm. We now show how μ can be replaced with a clique tree over the PDG's structure. What makes this possible is a key independence property of traditional graphical models, which we now prove holds for PDGs as well.

Theorem 6 (Markov Property for PDGs)**.** *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are PDGs over the sets of variables $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively, then $\mathcal{X}_1$ and $\mathcal{X}_2$ are conditionally independent given $\mathcal{X}_1 \cap \mathcal{X}_2$ in every $\mu \in \llbracket \mathcal{M}_1 + \mathcal{M}_2 \rrbracket_\gamma^*$, for all $\gamma > 0$ and $\gamma = 0^+$.*

For the remainder of this section, fix a PDG $\mathcal{M}$ and a tree decomposition $(\mathcal{C}, \mathcal{T})$ of $\mathcal{M}$'s hypergraph. One significant consequence of Theorem 6 is that, in the search for optimizers of (2), we need consider only distributions that satisfy those independencies, all of which can be represented as a clique tree $\boldsymbol{\mu} = \{\mu_C \in \Delta\mathcal{V}(C)\}_{C \in \mathcal{C}}$ over $(\mathcal{C}, \mathcal{T})$.

Corollary 6.1. *If $\mathcal{M}$ is a PDG with arcs $\mathcal{A}$, $(\mathcal{C}, \mathcal{T})$ is a tree decomposition of $\mathcal{A}$, $\gamma > 0$, and $\mu \in \llbracket \mathcal{M} \rrbracket_\gamma^*$, then there exists a clique tree $\boldsymbol{\mu}$ over $(\mathcal{C}, \mathcal{T})$ such that $\Pr_{\boldsymbol{\mu}} = \mu$.*

For convenience, let $\mathcal{V}\mathcal{C} := \{(C, c) : C \in \mathcal{C}, c \in \mathcal{V}(C)\}$ be the set of all choices of a cluster together with a setting of its variables. Like before, we start by optimizing *OInc*, this time over calibrated clique trees $\boldsymbol{\mu}$, which we identify with vectors $\boldsymbol{\mu} \cong [\mu_C(C{=}c)]_{(C,c) \in \mathcal{V}\mathcal{C}}$. We need the conditional marginals $\Pr_{\boldsymbol{\mu}}(T_a|S_a)$ of $\boldsymbol{\mu}$ along every arc a in order to calculate $\mathit{OInc}_{\mathcal{M}}(\Pr_{\boldsymbol{\mu}})$; fortunately, they are readily available. Since $(\mathcal{C}, \mathcal{T})$ is a tree decomposition, we know S_a and T_a lie entirely within some cluster $C_a \in \mathcal{C}$, and $\Pr_{\boldsymbol{\mu}}(T_a|S_a) = \mu_{C_a}(T_a|S_a)$ if $\boldsymbol{\mu}$ is calibrated. For $\mathbf{u} \in \bar{\mathbb{R}}^{\mathcal{V}\mathcal{A}}$, consider the problem

$$\begin{aligned} &\underset{\boldsymbol{\mu},\mathbf{u}}{\text{minimize}} \quad \sum_{(a,s,t) \in \mathcal{V}\mathcal{A}} \beta_a\, u_{a,s,t} && (11)\\ &\text{subject to} \quad \forall C \in \mathcal{C}.\ \mu_C \in \Delta\mathcal{V}(C), \\ &\forall a \in \mathcal{A}.\ \big(-\mathbf{u}_a,\ \mu_{C_a}(S_a,T_a),\ \mu_{C_a}(S_a)\mathbb{P}_a(T_a|S_a)\big) \in K_{\exp}^{\mathcal{V}a} \\ &\quad \forall (C,D) \in \mathcal{T}.\ \ \mu_C(C \cap D) = \mu_D(C \cap D), \end{aligned}$$

where again $\mathbf{u}_a$ is the restriction of $\mathbf{u}$ to components associated with a. Problem (11) is similar to (5), except that it requires local marginal constraints to restrict our search to calibrated clique trees. It is analogous to problem CTree-Optimize-KL of Koller and Friedman [2009, pg. 384].

Proposition 7. *If $(\boldsymbol{\mu}, \mathbf{u})$ is a solution to (11), then*

(a) $\boldsymbol{\mu}$ is a calibrated, with $\Pr_{\boldsymbol{\mu}} \in \llbracket \mathcal{M} \rrbracket_0^*$, and
(b) the objective of (11) evaluated at $\mathbf{u}$ equals $\langle\!\langle \mathcal{M} \rangle\!\rangle_0$.

We can now find a minimizer of *OInc* and compute $\langle\!\langle \mathcal{M} \rangle\!\rangle_0$ without storing a joint distribution. But to do anything else, we must deviate from the template laid out in Section 3.

Dealing with Joint Entropy. In the construction of (11), we rely heavily on the fact that each term of $\mathit{OInc}_{\mathcal{M}}$ depends only on local marginal distributions $\mu_{C_a}(T_a, S_a)$ and

[1] Intuitively, this assserts $\mu_1(T_a|S_a) = \mu_2(T_a|S_a)$, but also handles cases where some $\mu_1(S_a{=}s)$ or $\mu_2(S_a{=}s)$ equals zero.

$\mu_{C_a}(S_a)$. The same is not true of *SInc*, which depends on the joint entropy $\mathrm{H}(\Pr_{\boldsymbol{\mu}})$ of the entire distribution. At this point we should point out an important reason to restrict our focus to trees: it allows the joint entropy to be expressed in terms of the cluster marginals [Wainwright et al., 2008], by

$$-\mathrm{H}(\Pr_{\boldsymbol{\mu}}) = -\sum_{C \in \mathcal{C}} \mathrm{H}(\mu_C) \;+ \sum_{(C,D) \in \mathcal{T}} \mathrm{H}_{\boldsymbol{\mu}}(C \cap D). \tag{12}$$

Even so, it is not obvious that (12) can be captured with dcp exponential cone constraints. (Exponential conic programs can minimize negative entropy, but not positive entropy, which is concave.) We now describe how this can be done.

Choose a root node C_0 of the tree decomposition, and orient each edge of $\mathcal{T}$ so that it points away from C_0. Now each cluster $C \in \mathcal{C}$, except for C_0, has a parent cluster $\mathrm{Par}(C)$; define $\mathrm{Par}(C_0) := \emptyset$ to be an empty cluster, since C_0 has no parent. Finally, for each $C \in \mathcal{C}$, let $\mathit{VCP}_C := C \cap \mathrm{Par}(C)$ denote the the set of **v**ariables that cluster C has in **c**ommon with its **p**arent cluster.[2]As $\mathcal{T}$ is now a directed tree, this definition allow us to express (12) in a more useful form:

$$\begin{aligned} -\mathrm{H}(\,\Pr_{\boldsymbol{\mu}}) &= -\,\mathrm{H}(\mu_{C_0}) - \sum_{(C \to D) \in \mathcal{T}} \mathrm{H}_{\Pr_{\boldsymbol{\mu}}}(D \mid C) \\ &= \sum_{C \in \mathcal{C}} \sum_{c \in \mathcal{V}(C)} \mu_C(C{=}c) \log \frac{\mu_C(C{=}c)}{\mu_C(\,\mathit{VCP}_C(c))}, \end{aligned} \tag{13}$$

where $\mathit{VCP}_C(c)$ is the restriction of the joint value $c \in \mathcal{V}(C)$ to the variables $\mathit{VCP}_C \subseteq C$. Crucially, the denominator of (13) is an affine transformation of μ_C. The upshot: we have now rewritten the joint entropy as a sum of functions of the clusters, each of which can be captured with a dcp exponential cone constraint. This gives us analogues of the problems in Sections 3.2 and 3.3 that operate on clique trees.

Finding clique trees for $\hat{\gamma}$-inference. The ability to decompose the joint entropy as in (13) allows us to adapt (7) to operate on calibrated clique trees, rather than joint distributions. Beyond the changes already present in (11), the key is to replace the exponential cone constraint $(-\mathbf{v}, \mu, \mathbf{1}) \in K_{\exp}^{\mathcal{V}\mathcal{X}}$, which captures the entropy of μ, with

$$\big(-\mathbf{v},\; \boldsymbol{\mu},\; [\,\mu_C(\,\mathit{VCP}_C(c))\,]_{(C,c)\in\mathcal{V}\mathcal{C}}\big) \in K_{\exp}^{\mathcal{V}\mathcal{C}},$$

which captures the entropy of $\boldsymbol{\mu}$, by (13). Over vectors $\mathbf{v}, \boldsymbol{\mu} \in \overline{\mathbb{R}}^{\mathcal{V}\mathcal{C}}$ and $\mathbf{u} \in \overline{\mathbb{R}}^{\mathcal{V}\mathcal{A}}$, the problem becomes:

$$\begin{aligned} \underset{\boldsymbol{\mu},\mathbf{u},\mathbf{v}}{\textsf{minimize}}\quad & \sum_{(a,s,t)\in\mathcal{V}\mathcal{A}} (\beta_a - \alpha_a\gamma) u_{a,s,t} + \gamma \sum_{(C,c)\in\mathcal{V}\mathcal{C}} v_{C,c} \\ & - \sum_{(a,s,t)\in\mathcal{V}\mathcal{A}^+} \alpha_a \gamma\, \mu_{C_a}(S_a{=}s, T_a{=}t) \log \mathbb{P}_a(T_a{=}t \mid s) \\ \textsf{subject to}\quad & \forall C \in \mathcal{C}.\ \mu_C \in \Delta\mathcal{V}(C), \\ & \forall a \in \mathcal{A}.\ \big(-\mathbf{u}_a,\, \mu_{C_a}(S_a, T_a),\, \mu_{C_a}(S_a)\mathbb{P}_a(T_a|S_a)\big) \in K_{\exp}^{\mathcal{V}a}, \\ & \forall (a,s,t) \in \mathcal{V}\mathcal{A}^0.\ \mu_{C_a}(S_a{=}s, T_a{=}t) = 0, \\ & \forall (C,D) \in \mathcal{T}.\ \ \mu_C(C \cap D) = \mu_D(C \cap D), \\ & \big(-\mathbf{v},\; \boldsymbol{\mu},\; [\,\mu_C(\,\mathit{VCP}_C(c))\,]_{(C,c)\in\mathcal{V}\mathcal{C}}\big) \in K_{\exp}^{\mathcal{V}\mathcal{C}}. \end{aligned} \tag{14}$$

Proposition 8. *If $(\boldsymbol{\mu}, \mathbf{u}, \mathbf{v})$ is a solution to (14), and $\boldsymbol{\beta} \geq \gamma\boldsymbol{\alpha}$, then $\Pr_{\boldsymbol{\mu}}$ is the unique element of $[\![\mathcal{M}]\!]_{\gamma}^{*}$, and the objective of (14) at $(\boldsymbol{\mu}, \mathbf{u}, \mathbf{v})$ equals $\langle\!\langle \mathcal{M} \rangle\!\rangle_{\gamma}$.*

A related use of (13) is to enable an analogue of (10) that operates on clique trees (rather than joint distributions), to find a compact representation of $[\![\mathcal{M}]\!]_{0+}^{*}$. We begin with a straightforward adaptation of the relevant machinery in Section 3.3. Suppose that $\boldsymbol{\nu}{=}\{\nu_C : C \in \mathcal{C}\}$ is a calibrated clique tree over the tree decomposition $(\mathcal{C}, \mathcal{T})$ representing a distribution $\Pr_{\boldsymbol{\nu}} \in [\![\mathcal{M}]\!]_{0}^{*}$, say obtained by solving (11). For $C \in \mathcal{C}$, let $\mathcal{A}_C := \{a \in \mathcal{A} : C_a = C\}$ be the set of arcs assigned to cluster C, and let

$$\mathbf{k} := \Big[\prod_{a \in \mathcal{A}_C} \nu_C(T_a(c) | S_a(c))^{\alpha_a} \Big]_{(C,c)\in\mathcal{V}\mathcal{C}} \in \overline{\mathbb{R}}^{\mathcal{V}\mathcal{C}}$$

be the analogue of (9) for a cluster tree. Once again, consider $\mathbf{u} := [u_{(C,c)}]_{(C,c)\in\mathcal{V}\mathcal{C}}$ in the optimization problem

$$\begin{aligned} \underset{\boldsymbol{\mu},\mathbf{u}}{\textsf{minimize}}\quad & \mathbf{1}^{\mathsf{T}}\mathbf{u} \\ \textsf{subject to}\quad & \forall C \in \mathcal{C}.\ \mu_C \in \Delta\mathcal{V}(C), \\ & \big(-\mathbf{u},\, \boldsymbol{\mu},\, \mathbf{k} \odot [\ \mu_C(\,\mathit{VCP}_C(c))\ ]_{(C,c)\in\mathcal{V}\mathcal{C}}\big) \in K_{\exp}^{\mathcal{V}\mathcal{C}}, \\ & \forall a \in \mathcal{A}.\ \ \mu_{C_a}(S_a, T_a)\nu_{C_a}(S_a) = \mu_{C_a}(S_a)\nu_{C_a}(S_a, T_a) \\ & \forall (C,D) \in \mathcal{T}.\ \ \mu_C(C \cap D) = \mu_D(C \cap D). \end{aligned} \tag{15}$$

The biggest change is in the second constraint: the upper bounds $[u_{(C,c)}]_{c\in\mathcal{V}C}$ for cluster C now account only for the additional entropy not already modeled by C's ancestors.

Proposition 9. *If $(\boldsymbol{\mu}, \mathbf{u})$ is a solution to (15), then $\boldsymbol{\mu}$ is a calibrated clique tree and $[\![\mathcal{M}]\!]_{0+}^{*} = \{\Pr_{\boldsymbol{\mu}}\}$.*

At this point, standard algorithms can use $\boldsymbol{\mu}$ to answer probabilistic queries about $\Pr_{\boldsymbol{\mu}}$ in polynomial time [Koller and Friedman, 2009, §10.3.3]. From Propositions 8 and 9, it follows that $\hat{\gamma}$-inference (for small $\hat{\gamma}$, and for 0^+) can be reduced to a (pair of) convex optimization problem(s) with a polynomial number of variables and constraints. All that remains is to show that such a problem can be solved in polynomial time. For this, we turn to interior-point methods. As (14) and (15) are dcp, they can be transformed via established methods [Agrawal et al., 2018] into a standard form that can be solved in polynomial time by commercial solvers [ApS, 2022, Domahidi et al., 2013]. Threading the details of our constructions through the analyses of Dahl and Andersen [2022] and Nesterov et al. [1999] results in our main theorem.

[2] Different choices of C_0 yield different definitions of *VCP*, and ultimately optimization problems of different sizes; the optimal choice can be found with Edmund's Algorithm [Chu, 1965], which computes a directed analogue of the minimum spanning tree.

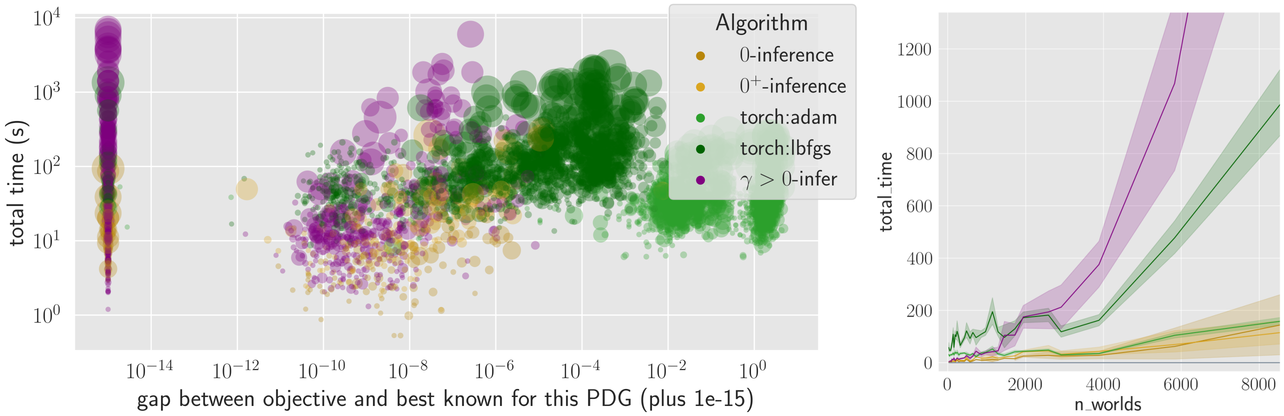

Figure 1: Accuracy and resource costs for the methods in Section 3. Left: a scatter plot of several algorithms on random PDGs of ≈ 10 variables. The x-axis is the difference in scores $\llbracket \mathcal{M} \rrbracket_\gamma(\mu) - \llbracket \mathcal{M} \rrbracket_\gamma(\mu^*) + 10^{-15}$, where μ is the method's output, and μ^* achieves best (smallest) known value of $\llbracket \mathcal{M} \rrbracket_\gamma$. (Thus, the best solutions lie on the far left.) The y axis is the time required to compute μ. Our methods are in gold (0^+-inference) and violet ($\hat\gamma$-inference, for $\hat\gamma > 0$); the baselines (black-box optimizers applied directly to (2)) are in green. The area of each circle is proportional to the size of the optimization problem, as measured by `n_worlds`$:= |\mathcal{VX}|$. Right: how the same methods scale in run time, as $|\mathcal{VX}|$ increases.

Theorem 10. *Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathbb{P}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ be a proper discrete PDG with $N = |\mathcal{X}|$ variables each taking at most V values, and $A = |\mathcal{A}|$ arcs forming a hypergraph of treewidth T. Then for all $\gamma \in \{0^+\} \cup (0, \min_{a\in\mathcal{A}} \frac{\beta_a}{\alpha_a}]$ and $\epsilon > 0$, we can do $\hat\gamma$-inference to precision ϵ in*

$$O\bigg((N{+}A)^4 V^{4T} \Big(T \log V + \log \frac{N{+}A}{\epsilon} + \log \frac{\beta^{\max}}{\beta^{\min}} \Big) \bigg)$$

time,[†] *where* $\beta^{\max} := \max_{a\in\mathcal{A}} \beta_a$ *and*

$$\beta^{\min} := \begin{cases} \min\limits_{a\in\mathcal{A}} \{\beta_a : \beta_a > 0\} & \text{if } \gamma = 0^+ \\ \gamma & \text{if } \gamma > 0 \ . \end{cases}$$

Our approach to $\hat\gamma$-inference computes $\langle\!\langle \mathcal{M} \rangle\!\rangle_\gamma$ as a side effect. But suppose that we were interested in calculating only this inconsistency. Might there be a more direct, asymptotically easier way to do so? In general, the answer is no.

Theorem 11. (a) *Determining whether or not there is a distribution that shares all cpds with $\mathcal{M}$ is NP-hard.*
(b) *Computing $\langle\!\langle \mathcal{M} \rangle\!\rangle_\gamma$ is #P-hard, for all $\gamma \geq 0$.*
(c) *For $\gamma \in \{0^+\} \cup (0, \min_a \frac{\beta_a}{\alpha_a})$, there is an $O($size of query result$)$ reduction from $\hat\gamma$-inference to the problem of calculating γ-inconsistency. Under bounded treewidth, there is also an $O(|\mathcal{VC}|)$ reduction in the other direction, making the problems essentially equivalent.*

Part (b) undermines Richardson and Halpern's original idea of inferring the probability of Y given X by adding a hypothesis $h(Y|X)$ to $\mathcal{M}$, and adjusting h to minimize the overall inconsistency $\langle\!\langle \mathcal{M} + h \rangle\!\rangle_\gamma$. If we had oracle access to $\langle\!\langle \mathcal{M} + h \rangle\!\rangle_\gamma$, however, then this procedure would not only give the right answer, but also its running time would not depend on the size of $\mathcal{M}$. Indeed, that is how we prove (c).

5 EXPERIMENTS

We have given the first algorithm to provably do inference in polynomial time, but that does not mean that it is the best way of answering queries in practice; it also makes sense to use black-box optimization tools such as Adam [Kingma and Ba, 2014] or L-BFGS [Fletcher, 2013] to find minimizers of $\llbracket \mathcal{M} \rrbracket_\gamma$. Indeed, this scoring function has several properties that make it highly amenable to such methods: it is infinitely differentiable, γ-strongly convex, and its derivatives have simple closed-form expressions. So it may seem surprising that $\llbracket \mathcal{M} \rrbracket_\gamma$ poses a challege to standard optimization tools—but it does, even when we optimize directly over joint distributions.

Synthetic Experiment 1 (over joint distributions)**.** Repeatedly do the following. First, randomly generate a small PDG $\mathcal{M}$ containing at most 10 variables and 15 arcs. Then for various values of $\gamma \in \{0, 0^+, 10^{-8}, \ldots, \min_a \frac{\beta_a}{\alpha_a}\}$, optimize $\llbracket \mathcal{M} \rrbracket_\gamma(\mu)$ over joint distributions μ, in one of two ways.

(a) Use `cvxpy` [Diamond and Boyd, 2016] to feed one of problems (5,7,10) to the MOSEK solver [ApS, 2022], or
(b) Choose a learning rate and a representation of μ in terms of optimization variables $\theta \in \mathbb{R}^n$. Then run a standard optimizer (Adam or L-BFGS) built into `pytorch` [Paszke et al., 2019] to optimize θ until μ_θ converges to a minimizer of $\llbracket \mathcal{M} \rrbracket_\gamma$ (or a time limit is reached). Keep only the best result across all learning rates.

[†] At the cost of substantial overhead and engineering effort, the exponent 4 can be reduced to 2.872, by appeal to Skajaa and Ye [2015] and the current best matrix multiplication algorithm [Duan et al., 2022, $O(n^{2.372})$] to invert $n{\times}n$ linear systems.

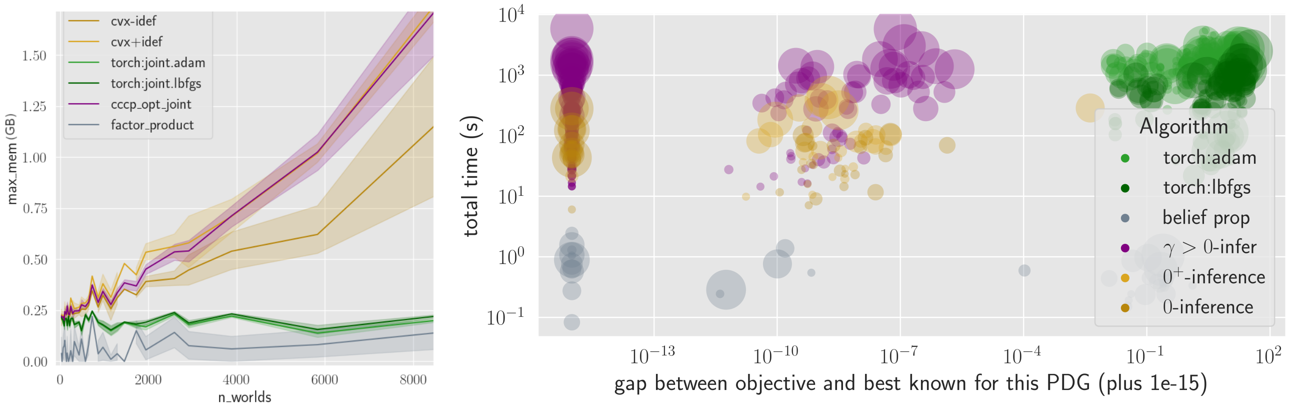

Figure 2: Left: Memory footprint. The convex solver (violet, gold) requires more memory than baselines (green). Right: Analogue of Figure 1 for the cluster setting. Here there is even more separation between exponential conic optimization (gold, violet) and black-box optimization (greens). The grey points represent belief propagation, which is fastest and most accurate—but only applies in the special case when $\boldsymbol{\beta} = \gamma\boldsymbol{\alpha}$.

The results are shown in Figure 1. Observe that the convex solver (gold, violet) is significantly more accurate than the baselines, and also much faster for small PDGs. Our implementation of 0^+-inference (gold) also appears to scale better than L-BFGS in this regime, although that of $\hat{\gamma}$-inference (purple) seems to scale much worse. We suspect that the difference comes from `cvxpy`'s complilation process, because the two use similar amounts of memory (Figure 2), and so are problems of similar sizes.

Synthetic Experiment 2 (over clique trees). For PDGs of bounded treewidth, Corollary 6.1 allows us to express these optimzation problems compactly not just for the convex solver, but for the black-box baseline approaches as well. We adapt the previous experiment for clique trees as follows. First randomly sample a maximal graph G of tree-width k, called a k-tree [Patil, 1986]; then generate a PDG $\mathcal{m}$ whose hyperarcs lie within cliques of G. This ensures that the maximal cliques of G form a tree-decomposition $(\mathcal{C}, \mathcal{T})$ of $\mathcal{m}$'s underlying hypergraph. We can now proceed as before: either encode (11,14,15) as disciplined convex programs in `cvxpy`, or use `torch` to directly minimize $[\![\mathcal{m}]\!]_\gamma(\Pr_{\boldsymbol{\mu}})$ amongst clique trees $\boldsymbol{\mu}$ over $(\mathcal{C}, \mathcal{T})$.

In the latter case, however, there is now an additional difficulty: it is not easy to strictly enforce the calibration constraints with the black-box methods. Common practice is to instead add extra loss terms to "encourage" calibration—but it can still be worthwile for the optimizer to simply incur that loss in order to violate the constraints. Thus, for fairness, we must recalibrate the the clique trees returned by all methods before evaluation. The result is an even more significant advantage for the convex solver; see Figure 2.

Evaluation on BNs. We also applied the procedure of the Synthetic Experiment 2 to the smaller BNs in the `bnlearn` repository, and found similar results (but with fewer examples; see Appendix C.3). But for a PDG that happens to also be a BN, it is possible to use belief propagation, which is much faster and at least as accurate.

Explicit details about all of our experiments, and many more figures, can be found in Appendix C.

6 DISCUSSION AND CONCLUSION

In this paper, we have provided the first practical algorithm for inference in PDGs. In more detail, we have defined a parametric family of PDG inference notions, given a fixed-parameter tractable inference algorithm for a subset of these parameters, proven our algorithm correct, implemented it, and shown our code to empirically outperform baselines. Yet many questions about PDG inference remain open.

Asymptotically, there may be a lot of room for improvement. Our implementation runs in time $\tilde{O}(N^4)$, and our analysis suggests one of time $\tilde{O}(N^{2.872})$. But assuming bounded tree-width, most graph problems, including inference inference for BNs and FGs, can be solved in time $O(N)$.

Moreover, we have shown how to do inference for only a subset of possible paramer values, specifically, when either $\boldsymbol{\beta} \geq \gamma\boldsymbol{\alpha}$ or $\boldsymbol{\beta} \gg \boldsymbol{\alpha}$. The remaining cases are also of interest, and likely require different techniques. When $\boldsymbol{\beta} = 0$ and $(\mathcal{A}, \boldsymbol{\alpha})$ encodes the structure of a BN, for instance, inference is about characterizing the BN's independencies. While we do not know how to tackle this problem in general, our methods can be augmented with the convex-concave procedure [Yuille and Rangarajan, 2003] to obtain an inference algorithm that applies slightly more broadly; see Appendix B.

Given the long history of improvements to Bayesian network inference algorithms, we are optimistic that further improvements on these problems are possible.

Acknowledgements

Halpern and Richardson were supported in part by MURI grant W911NF-19-1-0217 and ARO grant W911NF-17-1-0592. De Sa was supported by NSF RI-CAREER award 2046760.

References

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1): 42–60, 2018.

MOSEK ApS. *MOSEK Optimizer API for Python 10.0.25*, 2022. URL https://docs.mosek.com/10.0/pythonapi/index.html.

Riley Badenbroek and Joachim Dahl. An algorithm for nonsymmetric conic optimization inspired by mosek. *Optimization Methods and Software*, pages 1–38, 2021.

Hans L Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 226–234, 1993.

Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. Complexity of inference in graphical models. *arXiv preprint arXiv:1206.3240*, 2012.

Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. ISSN 0890-5401. doi: https://doi.org/10.1016/0890-5401(90)90043-H. URL https://www.sciencedirect.com/science/article/pii/089054019090043H.

Joachim Dahl and Erling D Andersen. A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization. *Mathematical Programming*, 194(1): 341–370, 2022.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076, 2013. doi: 10.23919/ECC.2013.6669541.

Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. *arXiv preprint*, 2022. doi: 10.48550/ARXIV.2210.10173. URL https://arxiv.org/abs/2210.10173.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

Michael Charles Grant. *Disciplined Convex Programming*. PhD thesis, Stanfor University, December 2004. URL https://web.stanford.edu/~boyd/papers/pdf/mcg_thesis.pdf.

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Yu E Nesterov, Michael J Todd, and Yinyu Ye. Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems. Technical report, 1999.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

HP Patil. On the structure of k-trees. *Journal of Combinatorics, Information and System Sciences*, 11(2-4):57–64, 1986.

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

Oliver E Richardson and Joseph Y Halpern. Probabilistic dependency graphs. *AAAI '21*, 2021.

Anders Skajaa and Yinyu Ye. A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. *Mathematical Programming*, 150(2):391–422, 2015.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2): 1–305, 2008.

Alan L Yuille and Anand Rangarajan. The concave-convex procedure. *Neural computation*, 15(4):915–936, 2003.