Learning Multivariate Distributions by Competitive Assembly of Marginals

Francisco Sánchez-Vega, Jason Eisner, Laurent Younes, and Donald Geman.
Johns Hopkins University, Baltimore, MD, USA

Abstract—We present a new framework for learning high-dimensional multivariate probability distributions from estimated marginals. The approach is motivated by compositional models and Bayesian networks, and designed to adapt to small sample sizes. We start with a large, overlapping set of elementary statistical building blocks, or "primitives", which are low-dimensional marginal distributions learned from data. Each variable may appear in many primitives. Subsets of primitives are combined in a lego-like fashion to construct a probabilistic graphical model; only a small fraction of the primitives will participate in any valid construction. Since primitives can be precomputed, parameter estimation and structure search are separated. Model complexity is controlled by strong biases; we adapt the primitives to the amount of training data and impose rules which restrict the merging of them into allowable compositions. The likelihood of the data decomposes into a sum of local gains, one for each primitive in the final structure. We focus on a specific subclass of networks which are binary forests. Structure optimization corresponds to an integer linear program and the maximizing composition can be computed for reasonably large numbers of variables. Performance is evaluated using both synthetic data and real datasets from natural language processing and computational biology.

Index Terms—graphs and networks, statistical models, machine learning, linear programming

1 Introduction

Probabilistic graphical models provide a powerful tool for discovering and representing the statistical dependency structure of a family of random variables. Generally, these models exploit the duality between conditional independence and separation in a graph in order to describe relatively complex joint distributions using a relatively small number of parameters. In particular, such graded models are potentially well-adapted to small-sample learning, where the bias-variance trade-off makes it necessary to invest in model parameters with the utmost care. Learning models with a very reduced number of samples is no more difficult than with a great many. However, arranging for such models to generalize well to unseen sets of observations, i.e., preventing them from overfitting the training data, remains an open and active area of research in the small-sample domain.

The introduction of carefully chosen topological biases, ideally consistent with prior domain knowledge, can help to guide learning and avoid model overfitting. In practice, this can be accomplished by accepting a restricted set of graph structures as well as by constraining the parameter space to only encode a restricted set of dependence statements. In either case, we are talking about the design of a model class in anticipation of efficient learning.

Our model-building strategy is "compositional" in the sense of a lego-like assembly. We begin with a set of "primitives" — a large pool of low-dimensional, candidate distributions. Each variable may appear in many primitives and only a small fraction of the primitives will participate in any allowable construction. A primitive designates some of its variables as input (α variables) and others as output (ω variables). Primitives can be recursively merged into larger distributions by matching inputs with outputs: in each merge, one primitive is designated the "connector", and the other primitives' α variables must match a subset of the connector's ω variables. Matched variables lose their α and ω designations in the result. The new distribution over the union of variables is motivated by Bayesian networks, being the product of the connector's distribution with the other primitives' distributions conditioned on their α nodes. In fact, each valid construction is uniquely identified with a directed acyclic graph over primitives.

The process is illustrated in Fig. 1 for a set of fourteen simple primitives over twelve variables. This figure shows an example of a valid construction with two connected components using six of the primitives. The form of the corresponding twelve-dimensional probability distribution will be explained in the text. Evidently, many other compositions are possible.

We seek the composition which maximizes the likelihood of the data with respect to the empirical distribution over the training set. Due to the assembly pro-

- *F. Sánchez-Vega, L. Younes and D. Geman are with the Department of Applied Mathematics and Statistics, The Center for Imaging Science and the Institute for Computational Medicine, Johns Hopkins University, 3400 N. Charles St., Baltimore, MD 21218. E-mail: sanchez, laurent.younes, geman@jhu.edu.*
- *Jason Eisner is with the Department of Computer Science and the Center for Language and Speech Processing, Johns Hopkins University, 3400 N. Charles St., Baltimore, MD 21218. E-mail: jason@cs.jhu.edu*

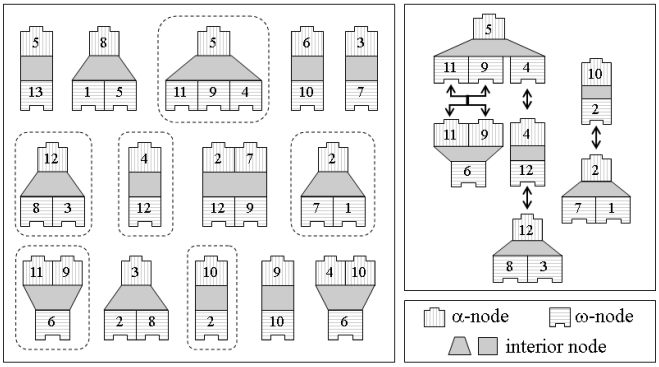

Fig. 1. Simple example of primitives and assemblies.

cess, the global "score" (i.e., expected log-likelihood) of any valid composition decomposes into a sum of local scores, one for each participating primitive; these scores are themselves likelihood ratios corresponding to the gain incurred in fusing the individual variables into the primitive distribution relative to independent variables. This decomposition has several consequences. First, all scores can be precomputed; consequently, parameter estimation (building primitives) and model construction (competitive assembly) are separated. That is, once the primitives are learned the process is data-independent. Second, in many cases, searching over all decompositions for valid compositions can be done by integer linear programming.

The primary intended application is molecular network modeling in systems biology, where it is common to encounter a complex underlying dependency structure among a large number of variables and yet a very small number of samples, at least relative to other fields such as vision and language. DNA microarrays provide simultaneous snap-shot measurements of the levels of expression of thousands of genes inside cells [1], [2], [3]. However, the number of profile measurements per experimental study remains quite small, usually fewer than a few hundreds. Similarly, advances in genotyping microarrays currently make it possible to simultaneously detect single nucleotide polymorphisms (SNPs) over millions of loci practically covering the entire genome of an organism, while the number of individuals in any given study remains orders of magnitude smaller [4], [5], [6]. Thus, any attempt to infer generalizable multivariate distributions from these data, in particular correlation patterns or even higher-dimensional interactions, must deal with well-known trade-offs in computational learning theory between sample size and model complexity [7], and between bias and variance [8].

Our proposals for model-building and complexity control are illustrated with both synthetic and real data. In the former case, experiments include measuring the KL divergence between the optimal composition and the true distribution as a function of the sample size and the number of variables. We compare our graphs with several well-known methods for "reverse engineering" networks, including relevance networks [9], ARACNE [10], CLR [11], which infer graphs from data, and the K2 algorithm [12] for learning Bayesian networks. We present two real-data experiments. One is based on inferring a semantic network from text. The other involves learning dependencies among mutations of the gene *TP53*, which plays a central role in cancer genomics. The substructures in the optimal composition appear biologically reasonable in the sense of aggregating driver mutations and being significantly enriched for certain cell functions. Still, we view our main contribution as methodological, these experiments being largely illustrative.

After discussing related work in Section 2, we will present the general theoretical framework for our models in Section 3, followed by specialization to a specific subclass based on balanced binary trees. In Section 4, we will discuss the choice of a statistically significant set of primitives. These primitives are combined to build the graph structure that maximizes the empirical likelihood of the observed data under a given set of topological constraints. In Section 5 we will show how the corresponding optimization problem can be dealt with using either greedy search or a more efficient integer linear programming formulation. Section 6 discusses the relationship of the foregoing approach to maximum *a posteriori* estimation of graphical model structure and parameters. After this, Section 7 presents some results from synthetic data simulations. In Section 8 we will look at further results obtained using the *20newsgroups* public dataset and the IARC TP53 Database. Finally, we will provide a general discussion and we will sketch some directions for future research.

2 Related Work

Historically, the problem of finding an optimum approximation to a discrete joint probability distribution has been addressed in the literature during the last half century [13]. A seminal paper published by Chow and Liu in the late sixties already proposed the use of information theoretic measures to assess the goodness of the approximation and formulated the structure search as an optimization problem over a weighted graph [14]. Improvements to the original algorithm [15] as well as extensions beyond the original pairwise approach [16] have been proposed. Recently, the popularity of Bayesian networks combined with the need to face small-sample scenarios have led to several works where structural biases are imposed upon the graphs used to approximate the target distribution in order to facilitate learning. Bounded treewidth is an example of such structural constraints. Even though the initial motivation for this approach was to allow for efficient inference [17], [18], [19], there has been work on efficient structure learning [20] and work that uses this kind of bias to avoid model overfitting [21]. Other examples of structural bias aimed

at achieving better generalization properties are the use of L1 regularization to keep the number of edges in the network under control [22], and constraints provided by experts [23]. We will discuss in Section 6 how our method is related to these prior approaches.

Compositional representations of entities as hierarchies of elementary, reusable parts that are combined to form a larger whole constitute an active topic of research in computer vision. Such modeling frameworks are usually built upon a set of composition rules, based on parts and combinations of parts, that progressively define the likelihood of images given the presence of an object at a given pose [24], [25]. A very simple composition rule, based on each part voting for the presence of a relevant object around its location, under the assumption of complex poses, has been proposed in [26]. The hierarchical structures induced by this kind of aggregation procedures provide a convenient tool for hardwiring high-level contextual constraints into the models [27], [28], [29], [30].

Dependency networks, which were proposed in [31] as an alternative to standard Bayesian networks, also provide an interesting example of compositional modeling, since they are built by first learning a set of small graph substructures with their corresponding local conditional probabilities. These "parts" are later combined to define a single joint distribution using the machinery of Gibbs sampling. In any case, the idea of combining compositional modeling and Bayesian networks dates back to the nineties, with the multiply sectioned Bayesian networks (MSBNs) from [32] and the object-oriented Bayesian networks (OOBNs) from [33]. Both approaches, as our work, provide ways to combine a number of elementary Bayesian networks in order to construct a larger model. The final structure can be seen as a hypergraph where hypernodes correspond to those elementary building blocks and hyperlinks are used to represent relations of statistical dependence among them. Hyperlinks are typically associated to so-called "interfaces" between the blocks, which correspond to non-empty pairwise intersections of some of their constituting elements. Even though the actual definition of interface may vary, it usually involves a notion of d-separation of nodes at both of its sides within the network. Later on, the use of a relational structure to guide the learning process [34] and the introduction of structured data types based on hierarchical aggregations [35] (anticipated in [33]) led to novel families of models.

All of the above approaches must confront the structure search problem. That is, given a criterion for scoring graphical models of some kind over the observed variables, how do we *computationally* find the single highest-scoring graph, either exactly or approximately? Structure search is itself an approximation to Bayesian model averaging as in [36], but it is widely used because it has the computational advantage of being a combinatorial optimization problem. In the case of Bayesian networks, Spirtes et al. [37, chapter 5] give a good review of earlier techniques, while Jaakkola et al. [38] review more recent alternatives including exact ones. Like many of these techniques (but unlike the module network search procedure in [39]), ours can be regarded as first selecting and scoring a set of plausible building blocks and only then seeking the structure with the best total score [23]. We formalize this latter search as a problem of integer linear programming (ILP), much as in [38], even if our building blocks have, in general, more internal structure. However, in the particular case that we will present in this paper, we search over more restricted assemblies of building blocks, corresponding to trees (generalizing [14]) rather than DAGs. Thus, our ILP formulation owes more to recent work on finding optimal trees, e.g., non-projective syntax trees in computational linguistics [40].

3 Competitive Assembly of Marginals

In this section, we formulate structure search as a combinatorial optimization problem — *Competitive Assembly of Marginals* (CAM) — that is separated from parameter estimation. The family of models that we will consider is partially motivated by this search problem. We also present a specific subclass of model structures based on balanced binary forests.

3.1 General Construction

Our objective is to define a class of multivariate distributions for a family of random variables $\mathbf{X} = (X_i, i \in D)$ for $D = \{1, \ldots, d\}$, where X_i takes values in a finite set Λ. For simplicity, we will assume that all the X_i have the same domain, although in practice each X_i could have a different domain Λ_i. We shall refer to the elements of Λ^D as configurations.

First we mention the possibility of global, structural constraints: for each $S \subset D$, we are given a class $\mathcal{M}_S$ of "admissible" probability distributions over the set of subconfigurations Λ^S (or over S, with some abuse). Our construction below will impose $\mathcal{M}_S$ as a constraint on all joint distributions that we build over S. To omit such constraint, one can let $\mathcal{M}_S$ consist of all probability distributions on S. Let $\mathcal{M}^* = \bigcup_{S \subset D} \mathcal{M}_S$. If $\pi \in \mathcal{M}^*$, we will write $J(\pi)$ for its support, i.e., the uniquely defined subset of D such that $\pi \in \mathcal{M}_{J(\pi)}$.

3.1.1 Primitives as Elementary Building Blocks

The main ingredient in our construction is a family $\mathcal{T}_0$ of relatively simple probability distributions that we call *primitives*. A distribution over $\mathbf{X}$ will be in our class only if it factors into a product of conditional distributions each of which is specified by a primitive. The elements of $\mathcal{T}_0$ are triplets $\phi = (\pi, A, O)$ where $\pi \in \mathcal{M}^*$ and A, O are subsets of $J(\pi)$ that serve as "connection nodes." A set of five primitives is shown in Fig. 2. The variables (or "nodes") in A will be called

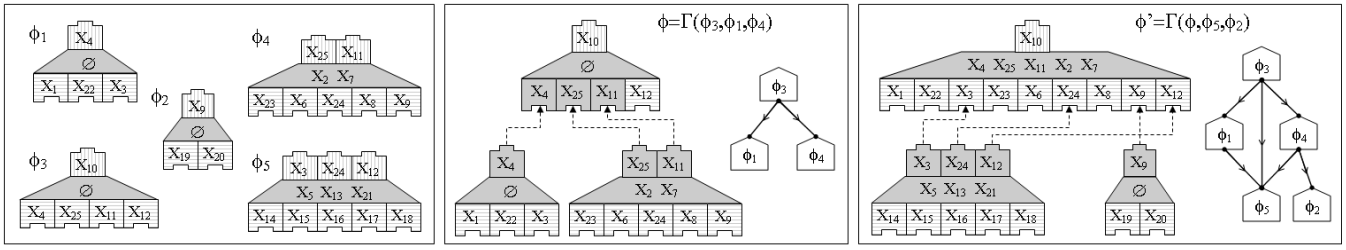

Fig. 2. Example of primitives and merge operations. Left panel shows a set of 5 primitives built from a set of $|D| = 25$ random variables. Center panel illustrates a merge operation where primitives ϕ_1 and ϕ_4 are bound using primitive ϕ_3 as a connector. The resulting new assembly $\phi = \Gamma(\phi_3, \phi_1, \phi_4)$ is shown, as well as its associated primitive DAG (where primitives are drawn as pentagons). Right panel shows the same diagrams for a second merge $\phi' = \Gamma(\phi, \phi_5, \phi_2)$ where the recently created assembly ϕ acts as a connector to bind ϕ_5 and ϕ_2.

α-nodes (A being the *α-set* of primitive ϕ), and the variables in O will be called *ω-nodes* (O being the *ω-set* of ϕ). We require $A, O \neq \emptyset$ and $A \cap O = \emptyset$. Any other nodes, $J(\pi) \setminus (A \cup O)$, are the *interior nodes* of ϕ.

3.1.2 Compositional Modeling by Primitive Merges

One can combine primitives (conditioning on their α-sets as needed) to build more complex distributions, which also have α- and ω-nodes. The merging of three primitives (ϕ_3, ϕ_1, ϕ_4) is illustrated in the second panel of Fig. 2. Here ϕ_3 serves as the binding primitive, or *connector*. The merged assembly ϕ is shown at the top of the third panel and has $|A| = 1, |O| = 9$.

Formally, given a group of primitives $(\phi_0, \phi_1, \ldots, \phi_r)$, where each $\phi_k = (\pi_k, A_k, O_k)$, we define a *merged* distribution π over $S = \bigcup_{k=0}^{r} J(\pi_k)$ as:

$$\pi(x_S) = \pi_0(x_{J(\pi_0)}) \prod_{k=1}^{r} \pi_k(x_{J(\pi_k)} \mid x_{A_k}) \qquad (1)$$

where ϕ_0 serves as the connector. (Here and below, x_I (for $I \subset D$) denotes the restriction of x to I, and we abuse notation by designating joint and conditional probabilities using the same letter as the original probability, the set over which they are defined being implicitly assigned by the variables.)

To ensure that (1) defines a proper distribution, we may only merge $(\phi_0, \phi_1, \ldots, \phi_r)$ when

(M1) $J(\pi_k) \cap J(\pi_0) = A_k \subset O_0$, for all $k = 1, \ldots, r$.
(M2) $J(\pi_k) \cap J(\pi_l) = \emptyset$ for all $k, l = 1, \ldots, r$, $k \neq l$.

(M1) ensures that the α-set of each ϕ_k matches some subset of the connector's ω-set. Together with (M2), it also ensures that, aside from those matchings, primitives $(\phi_0, \phi_1, \ldots, \phi_r)$ are over disjoint sets of variables.

We will say that the group $(\phi_0, \phi_1, \ldots, \phi_r)$ is *mergeable* with ϕ_0 as connector if (M1) and (M2) are satisfied and $\pi \in \mathcal{M}_S$. We then define the resulting merge to be the triplet $\phi = (\pi, A, O)$ with $A = A_0$ and $O = \bigcup_{k=0}^{r} O_k \setminus \bigcup_{k=1}^{r} A_k$. So the α-set of a merge is the α-set of the connector, and its ω-set is the union of the original ω-sets, from which the α-nodes of the non-connector primitives are removed (and become interior nodes in the new structure). This merge or output will be denoted $\phi = \Gamma(\phi_0, \ldots, \phi_r)$.

The merge operation can now be iterated to form probability distributions over increasingly large subsets of variables. This is illustrated in the last panel of Fig. 2, where the merge from the second panel is itself merged with two of the original primitives. For $S \subset D$, we will denote by $\mathcal{T}_S^*$ the set of probability distributions on S that can be obtained by a sequence of merges as defined above. If S is a singleton, we let, by convention, $\mathcal{T}_S^* = \mathcal{M}_S$. Finally, we let $\mathcal{T}^* = \bigcup_{S \subset D} \mathcal{T}_S^*$.

We would define our final model class $\mathcal{F}^*$ (of distributions over D) as $\mathcal{T}_D^*$, except that each distribution in $\mathcal{T}_D^*$ consists of a single connected component. Instead we define $\mathcal{F}^*$ as all product distributions of the form

$$P(x) = \prod_{k=1}^{c} \pi_k(x_{J(\pi_k)}) \qquad (2)$$

where $J(\pi_1), \ldots, J(\pi_c)$ partition D and $\pi_k \in \mathcal{T}_{J(\pi_k)}^*$.

The size and complexity of $\mathcal{F}^*$ are limited by two choices that we made earlier: the set of initial primitives $\mathcal{T}_0$, and the set of admissible distributions $\mathcal{M}^*$. Note that for $S \subsetneq D$, the constraints imposed by $\mathcal{M}_S$ on intermediate merges may be redundant with the final constraints imposed by $\mathcal{M}_D$ (as in Section 3.3 below), or may instead act to further restrict $\mathcal{F}^*$.

The final distribution P is a product of (conditionalized) primitives, whose relationships can be captured by a directed acyclic graph. Indeed, in view of (1), there is an evident connection with Bayesian networks which is amplified in the proposition below.

3.1.3 Atomic Decompositions and Primitive DAGs

Given $\Psi = \{\psi_1 \ldots, \psi_N\} \subset \mathcal{T}_0$, we will let $\mathcal{T}_\Psi$ denote the subset of $\mathcal{T}^*$ obtained by iterations of merge operations involving only elements of Ψ or merges built from them. Equivalently, $\mathcal{T}_\Psi$ is the set of distributions obtained by replacing $\mathcal{T}_0$ by Ψ in the construction above. If $\phi \in \mathcal{T}^*$, we will say that a family of primitives $\Psi = \{\psi_1 \ldots, \psi_N\} \subset \mathcal{T}_0$ is an *atomic decomposition* of ϕ if Ψ is a minimal subset of $\mathcal{T}_0$ such that $\phi \in \mathcal{T}_\Psi$ (i.e., ϕ can be built by merges involving only elements of Ψ and all elements of Ψ are needed in this construction). If $P \in \mathcal{F}^*$ is decomposed as in (2), an atomic decomposition of P is a union of atomic decompositions of each of its independent components. Finally, let $\mathcal{T}_0^*$ (resp. $\mathcal{F}_0^*$) be the set of

atomic decompositions of elements of $\mathcal{T}^*$ (resp. $\mathcal{F}^*$). The set of roots in the atomic decomposition $\Psi \in \mathcal{T}_0^*$ is denoted R_Ψ and defined as the set of indices k such that $A_k \cap J(\pi_l) = \emptyset$ for all $k, l = 1, \ldots, r$, $k \neq l$.

Proposition 1. *Let $\Psi = \{\psi_1, \ldots, \psi_N\} \in \mathcal{T}_0^*$ with $\psi_k = (\pi_k, A_k, O_k)$ and define the directed graph $G(\Psi)$ on $\{1, \ldots, N\}$ by drawing an edge from k to l if and only if $A_l \subset O_k$. Then $G(\Psi)$ is acyclic.*

This proposition is part of a larger one, Proposition S.1, which is stated and proved in Appendix I (see supplemental material). The acyclic graph $G(\Psi)$ is illustrated in Fig. 2 for the merges depicted in the middle and right panels. Notice that the nodes of these graphs are primitives, not individual variables. Consequently, our models are Bayesian networks whose nodes are overlapping *subsets* of our variables $\mathbf{X}$.

3.1.4 Generalization Using Weaker Merging Rules

We remark that the constraints defining merging rules could be relaxed in several ways, resulting in less restricted model families. For example, one could replace (M2) by the weaker condition that supports of non-connectors may intersect over their α-sets, i.e.,

(M2)' $J(\pi_k) \cap J(\pi_l) \subset A_k \cap A_l$.

Similarly, one can remove the constraint that ω-sets do not intersect α-sets within a primitive, allowing for more flexible connection rules, as defined by (M1) (the ω-set after merging would then simply be the union of all previous ω-sets, without removing the α-sets). Such modifications do not affect the well-posedness of (1). An extreme case is when primitives contain all possible pairs of variables, (M2) is replaced by (M2)' and the ω-set constraint is relaxed. Then our model class contains all possible Bayesian networks over the variables (i.e., all probability distributions).

3.2 Likelihood

We now switch to a statistical setting. We wish to approximate a target probability distribution P^* on Λ^D by an element of $\mathcal{F}^*$. This will be done by minimizing the Kullback-Leibler divergence between P^* and the model class. Equivalently, we maximize

$$L(P) = E_{P^*}(\log P) = \sum_{x \in \Lambda^d} P^*(x) \log P(x), \qquad (3)$$

where $P \in \mathcal{F}^*$. Typically, P^* is the empirical distribution obtained from a training set, in which case the procedure is just maximum likelihood.

Let each primitive distribution be parametric, $\phi = (\pi(\cdot;\theta), A, O)$, where θ is a parameter defined on a set Θ_ϕ (which can depend on ϕ). From the definition of merge, it is convenient to restrict the distributions in $\mathcal{F}^*$ by separately modeling the joint distribution of the α-nodes and the conditional distribution of the other nodes given the α-nodes. Therefore, we assume $\theta = (\sigma, \tau)$, where the restriction of π to A only depends on σ and the conditional distribution on $J(\pi)\backslash A$ given x_A only depends on τ, i.e.,

$$\pi(x_{J(\pi)}; \theta) = \pi(x_A; \sigma)\pi(x_{J(\pi)\backslash A} \mid x_A; \tau).$$

We assume that single-variable distributions are unconstrained, i.e., there is a parameter $P_j(\lambda)$ for each $\lambda \in \Lambda, j \in D$.

In order to maximize L, we first restrict the problem to distributions $P \in \mathcal{F}^*$ which have an atomic decomposition provided by a fixed family $\Psi = \{\psi_1, \ldots, \psi_N\} \in \mathcal{F}_0^*$. Afterwards, we will maximize the result with respect to Ψ. Let $\theta_k = (\sigma_k, \tau_k) \in \Theta_{\psi_k}, k = 1, \ldots, N$ and let $\ell(\theta_1, ..., \theta_N)$ be the expected log-likelihood (3). Rewriting the maximum of ℓ based on likelihood ratios offers an intuitive interpretation for the "score" of each atomic decomposition in terms of individual likelihood gains relative to an independent model. For any primitive $\phi = (\pi(\cdot;\theta), A, O)$, define

$$\begin{aligned} \rho(\phi) \quad = \quad & \max_\theta E_{P^*} \log \pi(X_{J(\pi)}; \theta) \qquad (4) \\ & - \max_\sigma E_{P^*} \log \pi(X_A; \sigma) + \sum_{j \in J(\pi)\backslash A} H(P_j^*), \end{aligned}$$

where $H(P_j^*) = -E_{P_j^*}(\log P_j^*)$. This is the expected log-likelihood ratio of the estimated parametric model for Ψ and an estimated model in which i) all variables in $J \setminus A$ are independent and independent from variables in A, and ii) the model on A is the original one (parametrized with σ). We think of this as an internal binding energy for primitive ϕ. Similarly, define

$$\mu(\phi) = \max_\sigma E_{P^*} \log \pi(X_A; \sigma) + \sum_{j \in A} H(P_j^*), \qquad (5)$$

the expected likelihood ratio between the (estimated) model on A and the (estimated) model which decouples all variables in A. Then it is rather straightforward to show that the maximum log-likelihood of any atomic decomposition decouples into primitive-specific terms. The proof, which resembles that of the Chow-Liu theorem [14], is provided in Appendix II.

Proposition 2. *Let $\ell(\theta_1, \cdots, \theta_N)$ be the expected log-likelihood of the composition generated by $\Psi = (\psi_1, \ldots, \psi_N) \in \mathcal{F}_0^*$. Then*

$$\max_{\theta_1, \ldots, \theta_N} \ell(\theta_1, \cdots, \theta_N) = \ell^*(\Psi) - \sum_{j \in D} H(P_j^*)$$

where

$$\ell^*(\Psi) = \sum_{k \in R_\Psi} \mu(\psi_k) + \sum_{k=1}^{N} \rho(\psi_k). \qquad (6)$$

Since the sum of entropies does not depend on Ψ, finding an optimal approximation of P^* "reduces" to maximizing ℓ^* over all possible Ψ. More precisely, finding an optimal approximation requires computing

$$\hat{\Psi} = \operatorname*{argmax}_{\Psi \in \mathcal{F}_0^*} \ell^*(\Psi) \qquad (7)$$

(with optimal parameters in (4) and (5)).

The important point is that the values of all ρ's and μ's can be precomputed *for all primitives*. Consequently, due to (6), any valid composition can be scored based only on the contributing primitives. In this way, parameter estimation is separated from finding the optimal Ψ. Obviously, the constraint $\Psi \in \mathcal{F}_0^*$ is far from trivial; it requires at least that Ψ satisfy conditions (i) and (ii) in Proposition S.1 (Appendix I). Moreover, computing $\hat{\Psi}$ typically involves a complex combinatorial optimization problem. We will describe it in detail for the special case that is our focus.

3.3 Balanced Compositional Trees

We now specialize to a particular set of constraints and primitives. In everything that follows, we will assume a binary state space, $\Lambda = \{0, 1\}$, and restrict the admissible distributions $\mathcal{M}^*$ to certain models that can be represented as trees (or forests), for which we introduce some notation. We call this subclass of models *balanced compositional trees*.

If T is a tree, let $J(T) \subset D$ denote the set of nodes in T. For $s \in J(T)$, s^+ denotes the set of children of s and s^- the set of its parents. Because T is a tree, s^- has only one element, unless s is the root of the tree, in which case $s^- = \emptyset$. We will say that T is *binary* if no node has more than two children, i.e., $|s^+| \leq 2$ (we allow for nodes with a single child). If $s^+ = \emptyset$ then s is called a terminal node, or leaf, and we let $\mathcal{L}(T)$ denote the set of all leaves in T. If s has two children, we will arbitrarily label them as left and right, with notation $s.l$ and $s.r$. Finally, T_s will denote the subtree of T rooted at s, i.e., T restricted to the descendants of s (including s). We will say that T is *almost-balanced* if for each $s \in J(T)$ such that $|s^+| = 2$, the number of leaves in $T_{s.l}$ and $T_{s.r}$ differ in at most one unit.

Probability distributions on $\Lambda^{J(T)}$ associated with T are assumed to be of the form:

$$\pi(x_{J(T)}) = p_0(x_{s_0}) \prod_{s \in J(T) \setminus \mathcal{L}(T)} p_s(x_{s^+} \mid x_s) \qquad (8)$$

where s_0 is the root of T, p_0 is a probability distribution and $p_s, s \in J(T) \setminus \mathcal{L}(T)$ are conditional distributions. This definition slightly differs from the usual one for tree models in that children do not have to be conditionally independent given parents.

For $S \subset D$, we will let $\mathcal{M}_S$ denote the set of models provided by (8), in which T is an almost-balanced tree such that $J(T) = S$. The balance constraint is introduced as a desirable inductive bias intended to keep the depth of the trees under control. The set $\mathcal{T}_0$ will consist of primitives $\phi = (\pi, A, O)$ where π is a probability distribution over a subset $J \subset D$ with cardinality two or three; A (the α-set) is always a singleton and we let $O = J \setminus A$. Primitives will be selected based on training data, using a selection process that will be described in the next section.

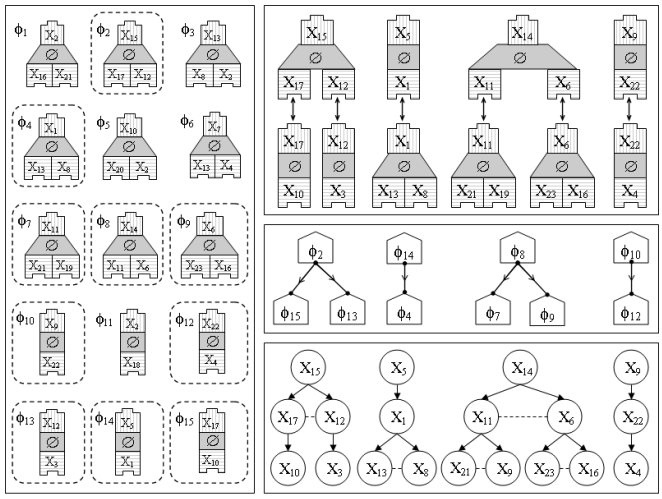

Fig. 3. Illustration of a set of primitives and an atomic decomposition for compositional trees. Left panel shows a pool of 15 primitives built from 23 variables. The encircled ones constitute an atomic decomposition for the four-component model depicted in the top-right panel. The center and bottom right panels show the corresponding DAGs of primitives and variables, respectively. In the last graph, dashed lines are used to link siblings within the same primitive.

Because α-sets have cardinality one, we have $\mu(\phi) = 0$ for all $\phi \in \mathcal{T}_0$ (where μ is defined in (5)), and (6) boils down to maximizing

$$\ell^*(\Psi) = \sum_{k=1}^{N} \rho(\psi_k) \qquad (9)$$

over $\mathcal{F}_0^*$. The description of $\mathcal{F}_0^*$ and of our maximization procedures is given in Section 5.

An example of a set of primitives that can be used to build balanced compositional trees is presented in Fig. 3, together with a sequence of elementary merges.

4 Primitive Selection

From here, we restrict our presentation to the particular case of balanced compositional trees. We discuss selecting an initial set of primitives $\mathcal{T}_0$ and estimating their parameters. We justify the need for a lower bound on the empirical likelihood gain for each accepted primitive and we describe a procedure for choosing this threshold based on controlling the expected number of false positives under the null hypothesis of mutual independence.

4.1 Stepwise Dependency Pursuit

We first specify the allowed representations for primitives, $\phi = (\pi(\cdot; \theta), A, O)$. Primitives are defined over pairs or triplets in D. With pairs, we allow π to be any distribution on $\{0, 1\}^2$. More precisely, letting $A = \{s\}$ and $O = \{t\}$, and using notation from Section 3.2, we let $\sigma = \pi_s(1)$ and $\tau = (\pi_t(1 \mid x_s = 0), \pi_t(1 \mid x_s = 1))$.

For triplets, we introduce several levels of complexity, each adding a single parameter to the joint distribution. Let $A = \{s\}$ and $O = \{u, v\}$. We can make the

joint distribution of (X_s, X_u, X_v) progressively more general with the following steps:

(1)	X_s, X_u, X_v are independent	(3 parameters).
(2)	$X_v \perp (X_s, X_u)$	(4 parameters).
(2′)	$X_u \perp (X_s, X_v)$	(4 parameters).
(3)	$X_u \perp X_v \mid X_s$	(5 parameters).
(4)	$X_u \perp X_v \mid X_s = 0$	(6 parameters).
(4′)	$X_u \perp X_v \mid X_s = 1$	(6 parameters).
(5)	Unconstrained joint	(7 parameters).

Case (1) corresponds to the default singletons, and (2), (2′) involve a pair primitive and a singleton. "True" triplet distributions correspond to (3) through (5). The selection process that we now describe will assign one model to any selected triplet (s, u, v).

If $d = |D|$ is the number of variables, there are $d(d-1)$ possible pairs and $d(d-1)(d-2)/2$ possible triplets. Since we are targeting applications in which d can reach hundreds or more, limiting the pool of primitives is essential to limiting the complexity of both statistical estimation and combinatorial optimization. The selection will be based on a very simple principle: only accept a primitive at a given complexity level when the previous level has been accepted and the expected likelihood increment in passing to the higher-dimensional model is sufficiently large. So, when building a primitive $\phi = (\pi(\cdot;\theta), A, O)$ supported by a set J, with $\theta \in \Theta_\phi$, we will assume a sequence of submodels $\Theta_1 \subset \Theta_2 \subset \cdots \subset \Theta_q$ and let Θ_ϕ be indexed by the largest k such that, for all $l = 1, \ldots, k-1$

$$\max_{\theta \in \Theta_{l+1}} E_{P^*} \log \pi(X_J;\theta) - \max_{\theta \in \Theta_l} E_{P^*} \log \pi(X_J;\theta) \geq \eta$$

where η is a positive constant and P^* is the empirical distribution computed from observations. For example, to form a pair primitive over $J = \{u, v\}$, we compare the joint empirical distribution over J (which estimates three parameters) to the one for which X_u and X_v are independent (which estimates two parameters), and we accept the pair primitive if

$$E_{P^*} \log \frac{P^*(X_u, X_v)}{P^*(X_u)P^*(X_v)} \geq \eta_{(2)}. \tag{10}$$

(For simplicity, we are just writing $P^*(X_u, X_v)$ for the empirical joint distribution of X_u, X_v; in each case the meaning should be clear from context.) In fact, we accept two primitives if this inequality is true: one for which u is the α-node and v the ω-node, and one for which the roles are reversed. Note that selection for pairs reduces to applying a lower bound on the mutual information, the same selection rule as in relevance networks [9].

For triplets, we will apply the analysis to the sequence of models (1), (2)/(2′), (3), … above. For example, to accept a triplet that corresponds to model (3), we first require that model (2) (or (2′)) is preferred to model (1), which implies that either the pair (s, u) or the pair (s, v) is accepted as a primitive using (10). We then demand that model (3) is significantly better than, say, model (2), meaning that

$$E_{P^*} \log \frac{P^*(X_u \mid X_s)P^*(X_v \mid X_s)}{P^*(X_u \mid X_s)P^*(X_v)} \geq \eta_{(3)}. \tag{11}$$

To select model (4), we need model (3) to have been selected first, and then, defining

$$\hat{P}(x_u, x_v \mid x_s) = \begin{cases} P^*(x_u \mid x_s)P^*(x_u \mid x_s) \text{ if } x_s = 0 \\ P^*(x_u, x_v \mid x_s) \text{ if } x_s = 1, \end{cases}$$

we will require

$$E_{P^*} \log \frac{\hat{P}(X_u, X_v \mid X_s)}{P^*(X_u \mid X_s)P^*(X_v \mid X_s)} \geq \eta_{(4)}. \tag{12}$$

Selecting model (5) is done similarly, assuming that either model (4) or (4)′ is selected.

4.2 Determination of the Selection Threshold

The threshold η determines the number of selected primitives and will be based on an estimation of the expected number of false detections. At each step of the primitive selection process, which correspond to the five numbered steps from above, we will assume a null hypothesis consistent with the dependencies accepted up to the previous level and according to which no new dependencies are added to the model. We will define η to ensure that the expected number of detections under this null is smaller than some $\epsilon > 0$, which will be referred to as the *selection threshold*.

We will fix $\eta_{(2)}$ such that the expected number of selected pairs under the assumption that all variables are pairwise independent is smaller than ϵ. Assuming that $m_{(2)}$ pairs have been selected, we will define $\eta_{(3)}$ to ensure that the expected number of selected triplets of type (3) is smaller than ϵ, under the assumption that any triplet of variables must be such that at least one of the three is independent from the others. Similarly, assuming that $m_{(3)}$ triplets are selected, $\eta_{(4)}$ will be chosen to control the number of false alarms under the hypothesis of all candidates following model (3). In some sense, selection at each level is done conditionally to the results obtained at the previous one.

At each step, the expected number of false alarms from model $(k-1)$ to (k) can be estimated by $\hat{\epsilon}$, which is defined as the product of the number of trials and the probability that model (k) is accepted given that model $(k-1)$ is true. Since model (k) is preferred to model $(k-1)$ when the likelihood ratio between the optimal models in each case is larger than $\eta_{(k)}$, $\hat{\epsilon}$ will depend on the distribution of this ratio when the smaller model is true. If the number of observations, n, is large enough, this distribution can be estimated via Wilks' theorem [41] which states that two times the likelihood ratio asymptotically follows a χ^2 distribution, with degrees of freedom given by the number of additional parameters (which is equal to one for each transition).

The number of trials for passing from level (3) to (4) and from level (4) to (5) is the number of selected triplets of the simplest type, i.e., $t_{(3)} = m_{(3)}$ and $t_{(4)} = m_{(4)}$ respectively. Between levels (2) and (3), we make $t_{(2)} = (d-2)m_{(2)}$ trials, and $t_{(1)} = d(d-1)/2$ trials between levels (1) and (2). With this notation, and the fact that each new level involves one additional parameter, we use the following selection process: let $\eta_{(k)}, k = 2, \ldots, 5$ be defined by

$$\eta_{(k)} = \frac{1}{2n} F_{step}^{-1}\left(1 - \frac{\epsilon}{4t_{(k-1)}}\right) \tag{13}$$

where F_{step} is the cumulative distribution function of a χ^2 with 1 d.f. (a standard normal squared) and the factor 4 ensures that the total number of expected false alarms across all levels is no more than ϵ.

For small values of n, the approximation based on Wilks' theorem is in principle not valid. Based on Monte-Carlo simulations, however, we observed that it can be considered as reasonably accurate for $n \geq 20$, which covers most practical cases. When $n < 20$, we propose to choose $\eta_{(k)}$ using Monte-Carlo (for very large values of d, the number of required Monte Carlo replicates may become prohibitively large, but learning distributions for extremely large d and $n < 20$ may be a hopeless task to begin with).

5 Structure Search Algorithm

The procedure defined in the previous section yields the collection, $\mathcal{T}_0$, of building blocks that will be composed in the final model. Each of these blocks, say ψ, comes with their internal binding energy, $\rho(\psi)$, which can be precomputed. The structure search problem, as described in (9), consists in maximizing

$$\ell^*(\Psi) = \sum_{k=1}^{N} \rho(\psi_k)$$

over all groups $\Psi = \{\psi_1, \ldots, \psi_N\} \in \mathcal{F}_0^*$, i.e., all groups of primitives that lead to a distribution on D that can be obtained as a sequence of legal merges on Ψ.

We start by describing $\mathcal{F}_0^*$. Recall that each family $\Psi = \{\psi_1, \ldots, \psi_N\} \subset \mathcal{T}_0$ defines an oriented graph $G(\Psi)$ on D, by inheriting the edges associated to each of the ψ_k's. We have the following fact (the proof is provided in Appendix III, as supplemental material).

Proposition 3. *A family of primitives $\Psi \subset \mathcal{T}_0$ belongs to $\mathcal{F}_0^*$ if and only if*

(i) *The α-nodes of the primitives are distinct.*
(ii) *The primitives do not share edges*
(iii) *$G(\Psi)$ is an almost-balanced binary forest.*

These conditions can be checked without seeking a particular sequence of admissible merges that yields $G(\Psi)$. That is, the structure search problem reduces to maximizing $\ell^*(\Psi)$ over all $\Psi = \{\psi_1, \ldots, \psi_N\}$ such that $G(\Psi)$ is an almost-balanced forest. This is still hard: when the true underlying distribution is rich in dependencies (yielding a large set $\mathcal{T}_0$), the number of possible Ψ's explodes combinatorially as the number of variables increases. Because of this, the exhaustive enumeration of all possible forests is not feasible. We propose two alternatives: a greedy search heuristic and a reformulation of the search as an ILP optimization problem, which can be solved using publicly available software (we worked with the Gurobi optimizer).

5.1 Greedy Search Solution

We begin with an edgeless graph where all variables are treated as singletons, i.e. $\Psi_0 = \emptyset$. The search operates by progressively adding new elements to Ψ until no such option exists. At step k of the procedure, with a current solution denoted Ψ_k, we define the next solution to be $\Psi^{(k+1)} = \Psi_k \cup \{\psi_{k+1}\}$ where ψ_{k+1} is chosen as the primitive for which ρ is maximized over all primitives that complete Ψ_k into a legal structure (and the procedure stops if no such primitive exists). At the end of each step, the set $\mathcal{T}_0$ can be progressively pruned out from all primitives that will never be used to complete the current Ψ_k, i.e., primitives that share an edge, or an α-node, with already selected ψ_j's, or primitives with ω-nodes coinciding with already allocated α-nodes. Of course, this strategy risks getting trapped in local maxima and is not guaranteed to find the optimal global solution.

5.2 Integer Linear Programming Solution

Exact maximization of $\ell^*(\Psi)$ is an ILP problem. Let V be the set of vertices and let $\mathcal{E}$ be the set of (oriented) edges present in $\mathcal{T}_0$. Here, whenever we speak of an edge we refer to edges in the graph structure associated to each primitive, where each node corresponds to a variable (as opposed to hyperedges in the higher level hypergraph where each node corresponds to a different primitive). The graph structure for pair primitives consists of an oriented edge from the α-node to the ω-node. The graph for triplet primitives consists of two oriented edges from the α-node to each of the ω-nodes (as shown in Fig. 3).

Introduce binary selector variables $x_\psi, \psi \in \mathcal{T}_0$ and $y_e, e \in \mathcal{E}$. For $e \in \mathcal{E}$, let $\mathcal{T}_e$ be the set of $\psi \in \mathcal{T}_0$ that contain e. We want to rephrase the conditions in Proposition 3 using linear equalities on the x's, y's and other auxiliary variables. (The meaning of the notation x, y, is different, in this section only, of what it is in the rest of the paper, in which it is used to denote realizations of random variables.)

We formulate the ILP here only in the specific setting of balanced compositional trees (Section 3.3), although the approach generalizes to other cases where $G(\Psi)$ is restricted to be a forest. If we wished to allow $G(\Psi)$ to be any DAG, we would modify the ILP problem to rule out only *directed* cycles [38], [42].

The first constraint is, for all $e \in \mathcal{E}$,

$$\sum_{t \in \mathcal{T}_e} x_t = y_e,$$

which ensures that every selected edge belongs to one and only one selected primitive.

We also need all edges in each selected primitive to be accounted for, which gives, for all $\psi \in \mathcal{T}_0$,

$$(|\psi| - 1)x_\psi \leq \sum_{e \in \psi} y_e$$

where $|\psi|$ is the number of vertices in ψ (two or three).

For every directed edge $e = (v, v')$ with $v, v' \in V$, let its reversal be $\bar{e} = (v', v)$. Our next constraint imposes $y_e + y_{\bar{e}} \leq 1$. Note that this constraint is made redundant by the acyclicity constraint. Still, it may be useful to speed up the solver.

Vertices must have no more than one parent and no more than two children, which gives, for all $v \in V$,

$$\sum_{(v',v) \in \mathcal{E}} y_{(v',v)} \leq 1 \quad \text{and} \quad \sum_{(v,v') \in \mathcal{E}} y_{(v,v')} \leq 2.$$

We also ensure that no vertex is the α-node of two distinct selected binary primitives. For $v \in V$, let Ψ_v denote the subset of $\mathcal{T}_0$ containing binary primitives with $\{v\}$ as an α-node. Then we want, for all $v \in V$

$$\sum_{\psi \in \Psi_v} x_\psi \leq 1.$$

The remaining conditions are more complex and require auxiliary variables. We first ensure that the graph associated to the selected ψ's is acyclic. This can be done by introducing auxiliary flow variables $f_e, e \in \mathcal{E}$ with the constraint

$$\begin{cases} -C(1 - y_e) + y_e + \sum_{e' \to e} f_{e'} \leq f_e \leq y_e + \sum_{e' \to e} f_{e'} \\ 0 \leq f_e \leq C y_e \end{cases}$$

where C is large enough (e.g., $C = |\mathcal{E}|$) and $e' \to e$ means that the child in edge e' coincides with the parent in edge e. (If $y_e = 1$, this condition implies $f_e = 1 + \sum_{e' \to e} f_{e'}$ which is impossible unless $f_e = \infty$ if the graph has a loop.)

The last condition is for balance. Introduce variables $g_e, e \in \mathcal{E}$ and $h_e, e \in \mathcal{E}$ with constraints

$$\begin{cases} 0 \leq h_e \leq y_e \\ h_e \leq 1 - y_{e'} \text{ if } e \to e' \\ h_e \geq 1 - \sum_{e \to e'} y_{e'} - C(1 - y_e) \\ -C(1 - y_e) + \sum_{e \to e'} g_{e'} \leq g_e \leq h_e + \sum_{e \to e'} g_{e'} \\ y_e \leq g_e \leq C y_e \\ \text{for all triplets } \psi, |g_{e(\psi)} - g_{e'(\psi)}| \leq 1 + C(1 - x_\psi) \end{cases}$$

where $e(\psi)$ and $e'(\psi)$ denote the two edges in triplet ψ. The variable h_e equals 1 if and only if e is a terminal edge. The variable g_e counts the number of leaves (or terminal edges). We have $g_e = 0$ if $y_e = 0$. If e is terminal and $y_e = 1$, then the sum over descendants vanishes and the constraints imply $g_e = 1$. Otherwise ($h_e = 0$ and $y_e = 1$), we have $g_e = \sum_{e \to e'} g_{e'}$. The last inequality ensures that the trees are almost-balanced.

The original problem can now be solved by maximizing $\sum_{\psi \in \mathcal{T}_0} \rho(\psi) x_\psi$ subject to these constraints, the resulting solution being $\hat{\Psi} = \{\psi : x_\psi = 1\}$.

ILP is a general language for formulating NP-complete problems. Although the worst-case runtime of ILP solvers grows exponentially with the problem size, some problem instances are much easier than others, and modern solvers are reasonably effective at solving many practical instances of moderate size. We show empirical runtimes in Section II of the supplemental material, together with an analysis of the size of the ILP encoding. Note that one can improve upon greedy search even without running the ILP solver to convergence, since the solver produces a series of increasingly good suboptimal solutions en route to the global optimum. Also, when the number of variables and constraints in the ILP problem becomes computationally prohibitive, we can adopt a hybrid search strategy: start by running a greedy search (which typically leads to a forest with several independent components) and then solve multiple ILP problems as the one described above, each restricted to $\psi's$ that are supported by the set of variables involved in each of those components. Even though the solution may still not be globally optimal, this coarse-to-fine approach may lead to improved performances over the use of greedy search and ILP alone.

6 Connection with MAP Learning of Bayesian Network Structure

To situate our statistical approach with respect to the prior work of Section 2, we now discuss how it relates to MAP estimation of Bayesian networks.

6.1 Primitives as Bayesian Networks

The approach that we propose in this paper consists in constructing small-dimensional primitives, possibly having complex parametrizations (if allowed by the data-driven selection process), and assembling them globally into a model covering all variables subject to complexity contraints. Note that, even though the global relationship among primitives is organized as a Bayesian network, as described in Section 3.1, the distribution specified by each primitive can be modeled in an arbitrary way. We conceive of these primitives as small modeling units, and the parametric representation introduced in Section 3.2 can be based on any appropriate model (one can use, for example, a Markov random field built by progressive maximum entropy extension [43], selected similarly to Section 4).

In the case in which these primitives are also modeled as Bayesian networks, the global distribution of our model is obviously also represented as such.

This case includes the compositional trees introduced in Section 3.3, for which deriving a Bayesian network representation in cases (2)–(5) is straightforward.

In such a case, an alternative characterization of our method is that we perform a structure search over Bayesian networks that can be partitioned into previously selected primitives. In this regard, it can be compared with other inductive biases, including the well-studied case of restricting the tree-width of the graph, which leads to a maximum-likelihood structure search problem that is equivalent to finding a maximum-weight *hyperforest* [44], [45]. Our global constraints $\mathcal{M}^*$ could be used to impose such a tree-width restriction on the graph *over primitives*, during greedy or global search. In particular, the compositional trees of Section 3.3 restrict this graph to tree-width 1, yielding our simpler combinatorial problem of finding a maximum-weight *forest* whose nodes are (possibly complex) primitives. Relaxing (M2) to (M2)′ in Section 3.1, we remark that if our primitives consist of all Bayesian networks on subsets of $\leq w+1$ variables, then assembling them under the global compositional-tree constraint gives a subset of Bayesian networks of tree-width w, while dropping the global constraint gives the superset $CPCP^w$ [46].

6.2 Primitive Selection vs. MAP Estimation

Suppose we omit the initial step of primitive filtering. Then purely maximum-likelihood estimation could be done using our global structure search algorithms from Section 5. Naturally, however, maximum-likelihood estimation will tend to overfit the data by choosing models with many parameters. (Indeed, this is the motivation for our approach.) A common remedy is instead to maximize some form of penalized log-likelihood. For many penalization techniques, this can be accomplished by the same global structure search algorithm over assemblies of primitives. One modifies the definition of each binding energy $\rho(\psi_k)$ in our maximization objective (9) to add a constant penalty that is more negative for more complex primitives ψ_k [23]. Before the search, it is safe to discard any primitive ψ such that the penalized binding energy $\rho(\psi) < \rho(\psi')$ for some ψ' with $(J(\psi), A(\psi)), O(\psi)) = (J(\psi'), A(\psi'), O(\psi'))$, since then ψ cannot appear in the globally optimal structure [23]. The filtering strategy in 4.1 can be regarded as a slightly more aggressive version of this, if the penalties are set to 0 for triplets of type (1), $-\eta_{(2)}$ for those of type (2)/(2′), $-(\eta_{(2)}+\eta_{(3)})$ for those of type (3), and so on.

One can regard the total penalty of a structure as the log of its prior probability (up to a constant). The resulting maximization problem can be seen as MAP estimation under a structural prior. To interpret our η penalties in this way would be to say that a random model, *a priori*, is $\exp\eta_{(3)}$ times less likely to use a given triplet of type (3) than one of type (2). However, our actual approach differs from the above MAP story in two ways. First, it is not fully Bayesian, since in Section 4.2, we set the η parameters of the prior from our training data (cf. empirical Bayes or jackknifing). Second, a MAP estimator would include the η penalties in the global optimization problem—but we use these penalties only for primitive selection.

Why the difference? While our approach is indeed somewhat similar to MAP estimation with the above prior, that is not our motivation. We do not actually subscribe to a prior belief that simple structures are more common than complex structures in our application domains (Section 8). Furthermore, our goal for structure estimation is not to minimize the posterior risk under 0-1 loss, which is what MAP estimation does. Rather, we seek an estimator of structure that bounds the risk of a model under a loss function defined as the number of locally useless correlational parameters. Our structure selection procedure conservatively enforces such a bound ϵ (by keeping the false discovery rate low even within $\mathcal{T}_0$ as a whole, and *a fortiori* within any model built from a subset of $\mathcal{T}_0$). Subject to this procedure, we optimize likelihood, which minimizes the posterior risk under 0-1 loss for a *uniform* prior over structures and parameters.

We caution that ϵ does not bound the number of incorrect edges relative to a true model. $\mathcal{T}_0$ includes all correlations that are valid within a primitive, even if they would vanish when conditioning also on variables outside the primitive (cf. [47]). To distinguish direct from indirect correlations, our method uses only global likelihood as in [14]. In the small-sample regime, the resulting models (as with MAP) can have structural errors but at least remain predictive without overfitting—as we now show. Bounding the number of incorrect edges would have to *underfit*, rejecting all edges, even if true or useful, that might be indirect.

7 Simulation Study

We assessed the performance of our learned models using synthetic data. Here we present an overview of our results. The full description of our simulations is provided in Section I of the supplemental material.

We first run several experiments to measure the effect of sample size, number of variables, selection threshold and search strategy upon learning performance when the true model belongs to our model class $\mathcal{F}^*$. We evaluated the quality of the estimation by computing the KL divergence between the known, ground truth distribution and the distribution learned using our models. We evaluated network reconstruction accuracy by building ROC and precision-recall (PR) curves in terms of true-positive and false-positive edges. The actual curves and further details can be found in Section I.A of the supplemental material. Besides the obvious fact that both quality of estimation and network reconstruction accuracy improved with

increasing sample sizes, our results showed that i) the choice of an excessively large selection threshold leads to model overfitting and ii) for very small samples, the distribution learned using CAM can be better (in terms of KL divergence to the ground truth) than the distribution learned by using the true generating graph and estimating parameters from data.

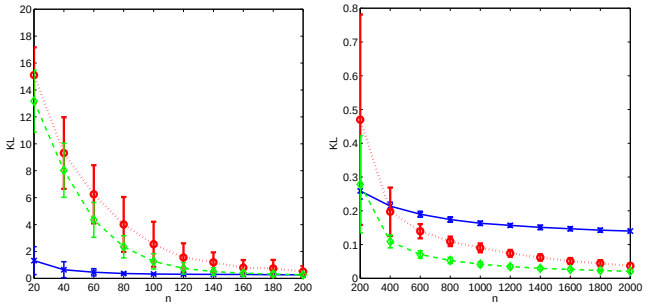

Fig. 4. KL divergence between the Bayesian network ground truth distribution ($d = 14$) and the distributions learned using CAM (solid, blue, cross), K2 (dotted, red, circle) and the true generating graph with MLE parameters (dashed, green, diamond). Results are shown for a fixed choice of parameters and averaged over 100 random replicates (see Section I.B of the supplemental material for details).

We compared CAM to other methods from the literature, namely Bayesian networks [48] (represented by the K2 algorithm [12]), relevance networks (RN) [9], ARACNE [10] and CLR [11]. Full details are included as Section I.B of the supplemental material. First, we sampled from a balanced binary forest like the ones described in previous sections. We found that CAM outperformed all the alternatives, both in terms of KL to the true distribution and network reconstruction accuracy. This was particularly evident for small samples. Of course, these results were not surprising because we were in the favorable case where the ground truth belonged to our constrained family of models. Next, we considered the unfavorable case where we sampled from a generic Bayesian network with a more complex dependency structure than those allowed within our model class. Fig. 4 shows an example of the type of curves that we observed (curves for other choices of parameters are shown in Fig. S.7 of supplemental material). CAM offered the best performance for small samples by favoring bias over variance. In fact, for small enough sample sizes CAM performed better than using the true graph and only estimating parameters, as we had remarked before. For larger sample sizes, CAM performed worse than the alternatives. This was expected: when data are plentiful and the dependency structure is rich, learning models more complex than ours becomes feasible. Precision-recall curves (Fig. S.9) showed that, for same levels of recall, K2 achieves the least precision. In general, RN, CLR and particularly ARACNE offer the best performances, although CAM is comparable for small samples. For larger sample sizes (over 100) CAM has lower precision than the others (except K2).

8 Real-Data Experiments

Our first experiment involves a semantic network learning task for which the results are reasonably easy to interpret. In the second one, we learn a network of statistical dependencies among somatic mutations in gene *TP53*, which plays an important role in cancer.

8.1 Learning Semantic Networks from Text

The *20newsgroups* dataset [49] is a collection of approximately 20,000 documents that were obtained from 20 different Usenet newsgroups. We worked with a reduced version made publicly available by the late Sam Roweis through his website at the New York University. The data are summarized in a matrix with binary occurrence values for $d = 100$ words across $n = 16,242$ newsgroup postings, these values indicating presence or absence of the word in a specific document. We discarded some words with general or ambiguous meanings, leaving 66 words that were clearly associated to six well differentiated semantic categories (*computers*, *space*, *health*, *religion*, *sports* and *cars*). Intuitively, we would expect the occurrences of words such as *dog* and *boat* to be approximately independent, whereas not so for say *hockey* and *nhl*.

We first measured the effect of the observed sample size. We evaluated edgewise network reconstruction accuracy using a hand-crafted ground-truth network where words that are semantically related were linked by an undirected edge. We chose random subsets of documents of different sizes and we learned a network for each of them using CAM, RN, CLR and ARACNE (K2 was not used because the causal ordering of the variables is unknown). The net result (not shown) is that all methods provide roughly comparable ROC and PR curves. We then compared the predictive performance of CAM versus K2 and a baseline edgeless Bayesian network as a function of sample size, by computing average log-likelihood on different sets of hold-out samples. We arbitrarily chose an entropy ordering for K2 and, in order to provide a fair comparison, we enforced the same ordering constraint upon the set of candidate CAM structures. Our results show that CAM outperforms the alternatives when sample sizes are small (full details can be found in Section III.A of the supplemental material).

Next, we learned the network shown in Fig. 5 using the full dataset. The network is componentwise optimal. It is not guaranteed to be globally optimal because the dual gap for the global optimization problem was non-negligible; still, the result appeared to be stable after extensive computation (see Section II.C for details). We observe a very good correspondence between network components and semantic categories. In fact, there is only one merge between components that may seem questionable: the *space* and *religion* components end up being connected by the word *earth*. This is a consequence of the local

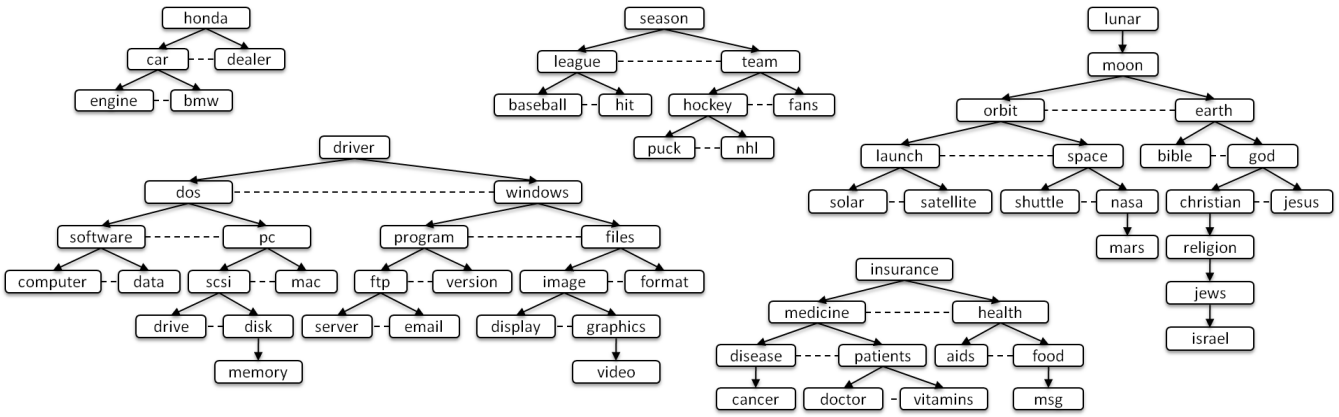

Fig. 5. Final network learned using a subset of words from the *20newsgroups* dataset and the full sample size. This result was obtained using the ILP search procedure. Dashed lines are used to identify siblings that belong to the same primitive within our model. Words belonging to the same semantic categories (*computers*, *space*, *health*, *religion*, *sports* and *cars*) tend to form proximity clusters within the network.

scope of our primitives and the fact that this word is frequently used within both semantic contexts. For comparison purposes, we learned two graphs using RN (see Section III.B of supplemental material). In the first case, the threshold for mutual information was the same as the one used for the CAM binary primitives in Fig. 5; in the RN case all the variables wound up in the same connected component and the large number of learned interactions made it somewhat difficult to interpret the result. In the second case, we equalized the number of learned edges, leading to the isolation of twenty variables as singletons (and thus to the failure to learn some important connections).

8.2 Learning Statistical Dependencies among Somatic Mutations in Gene *TP53*

The *TP53* gene is located on the short arm of chromosome 17 and encodes the p53 tumor suppressor protein, which plays a fundamental role in many human cancers [50], [51]. The p53 protein is activated when cells are stressed or damaged and it blocks their multiplication, providing an important mechanism to prevent tumor proliferation. Mutations in the *TP53* gene are primarily of the missense type. They are known to cause direct inactivation of the p53 protein in about half of the cancers where this protein fails to function correctly and indirect inactivation in many other cases [50]. Because of this, understanding the effect of these mutations can provide very valuable insight into the mechanisms of cancer [52].

Most of the somatic mutations reported in the literature are compiled in the International Agency for Research on Cancer (IARC) TP53 Database [53]. We used version R15 (updated in Nov. 2010) and worked with the somatic mutations dataset, which mainly consists of missense mutations detected by DNA sequencing in tumor samples and mutations within exons 5-8. The original dataset contains measurements for $d = 4{,}356$ different mutations and $n = 25,101$ different tissue samples. (Discarding mutations appearing in fewer than two samples leaves $d = 2{,}000$ and $n = 23{,}141$.) We can represent each sample as a d-dimensional binary vector indicating which mutations were observed in a tissue extracted from a patient. Since patients can have more than one cancer, multiple samples may come from the same individual. Still, we treat the vectors of observations as independent samples from an underlying multivariate distribution which characterizes the dependency structure among mutations in a cancer population.

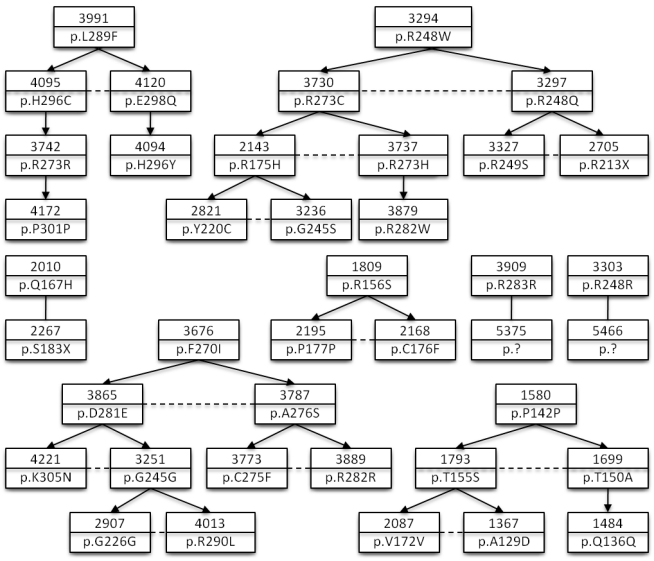

Fig. 6. Some components of the TP53 somatic mutations network learned using CAM (full network is provided in Fig. S.12). Dashed lines link siblings within the same primitive. For each node, we provide the unique mutation identifier in the IARC Database and the (standard) mutation description at the protein level, where *p.XzY* means substitution of amino acid X by amino acid Y at codon z; e.g., $p.R248W$ represents substitution of *Arg* by *Trp* at codon 248.

Selecting $\epsilon = 1$ leads to 760 candidate primitives. Using CAM, we learned a network that contained 68 different mutations (out of the original set of $2{,}000$ candidates). Fig. 6 shows some of the components in this network (the full network graph is provided in Fig. S.12, Section IV of supplemental material).

Each somatic mutation in IARC is annotated with biochemical details about the actual nucleotide variation, as well as clinical data for the patient and tissue where the mutation was observed. Enrichment of shared annotations is statistically significant for several subsets of mutations within our network, both at the pairwise and the component levels, which suggests that these mutations might be functionally related. Furthermore, based on a hypergeometric null, the mutations in our network were significantly enriched for several biological indicators such as presence in CpG sites and associated gain of function (GoF) at the protein level. The same type of test shows that our network is significantly enriched for "deleterious" or "driver" mutations (which are known to have a negative impact on the phenotype, as opposed to "neutral" or "passenger" ones). A detailed explanation of our statistical analysis, including some comments on the biological interpretation of our results, is provided in Section IV of the supplemental material.

9 Conclusions and Future Work

We have introduced a new modeling framework that combines local model selection (designing and estimating primitives) with a compositional step that merges primitives into valid graphical arrangements and multivariate distributions. This construction makes it possible to adjust model complexity to sample size by controlling the dimension of the parameter space. The introduction of structural biases can be used to decrease variance and avoid model overfitting, which is critical in small sample regimes.

Our approach has been validated using both synthetic and real data. For simulated data, our method outperforms general Bayesian networks in approximating the true generating distribution for small sample sizes. Our approach is also comparable with methods designed for recovering graphs rather than distributions. Finally, experiments with real data, particularly genetic mutations in the *TP53* gene, demonstrate that the CAM algorithm can cluster mutations into biologically plausible groups.

Even though in this paper we have only discussed the case of discrete random variables, the CAM framework generalizes to the continuous case where discrete distributions are replaced by probability density functions. Also, we have focused on balanced binary trees, which simplifies both primitive learning and the combinatorial optimization problem for competitive assembling. These constraints yield networks which are easy to interpret, since the limits on topological complexity often lead to a final graph with several components of moderate size. These limits include strong restrictions to the set of allowable values for incoming and outgoing vertex degrees. However, our entire framework applies more generally to any family of primitives, such as those depicted in Fig. 1, allowing us to move beyond the strong structural constraints imposed by trees in the context of moderate to large sample sizes. In particular, such extensions might allow for learning networks with "hubs" and scale-free properties provided near-optimal assemblies can be identified.

Acknowledgment

Francisco Sánchez-Vega was partially supported by graduate fellowships from La Caixa Foundation and Cajamadrid Foundation in Spain. The work of Donald Geman is partially supported by NIH-NCRR Grant UL1 RR 025005 and by the US National Science Foundation (NSF) CCF-0625687.

References

[1] D. J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J. M. Trent, "Expression profiling using cDNA microarrays," *Nat. Gen. Suppl.*, vol. 21, pp. 10–14, 1999.

[2] R. J. Lipshutz, S. P. Fodor, T. R. Gingeras, and D. J. Lockhart, "High-density synthetic oligonucleotide arrays," *Nat. Gen.*, vol. 21, pp. 20–24, 1999.

[3] Y. Wang, D. J. Miller, and R. Clarke, "Approaches to working in high-dimensional data spaces: Gene expression microarrays," *British J. Cancer*, February 2008.

[4] J. H. Moore and M. D. Ritchie, "The challenges of whole-genome approaches to common diseases," *J. Amer. Med. Assoc.*, vol. 291, no. 13, pp. 1642–1643, 2004.

[5] M. Morley, C. Molony, T. Weber, J. Devlin, K. Ewens, R. Spielman, and V. Cheung, "Genetic analysis of genome-wide variation in human gene expression," *Nature*, 2004.

[6] M. I. McCarthy, G. R. Abecasis, L. R. Cardon, D. B. Goldstein, J. Little, J. P. A. Ioannidis, and J. N. Hirschhorn, "Genome-wide association studies for complex traits: Consensus, uncertainty and challenges," *Nat. Rev. Gen.*, vol. 9, no. 5, pp. 356–369, 2004.

[7] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[8] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, 1992.

[9] A. J. Butte and I. S. Kohane, "Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements," *Pac. Symp. Biocomput.*, pp. 418–29, 2000.

[10] A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Favera, and A. Califano, "ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, vol. 7, p. S7, 2006.

[11] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner, "Large-scale mapping and validation of E. coli transcriptional regulation from a compendium of expression profiles," *PLoS Biol.*, vol. 5, no. 1, p. e8, Jan 2007.

[12] G. F. Cooper and T. Dietterich, "A Bayesian method for the induction of probabilistic networks from data," in *Machine Learning*, 1992, pp. 309–347.

[13] D. T. Brown, "A note on approximations to discrete probability distributions," *Info. and Control*, vol. 2, no. 4, pp. 386–392, 1959.

[14] C. I. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Trans. Inf. Theory*, vol. 14, pp. 462–467, 1968.

[15] M. Meila, "An accelerated Chow and Liu algorithm: Fitting tree distributions to high-dimensional sparse data," in *Proc. ICML*. San Francisco: Morgan Kaufmann, 1999, pp. 249–257.

[16] T. Szántai and E. Kovács, "Hypergraphs as a means of discovering the dependence structure of a discrete multivariate probability distribution," *Ann. of Operat. Res.*, pp. 1–20, 2010.

[17] F. R. Bach and M. I. Jordan, "Thin junction trees," in *NIPS 14*, 2001, pp. 569–576.

[18] A. Chechetka and C. Guestrin, "Efficient principled learning of thin junction trees," in *NIPS*, Vancouver, Canada, 2007.
[19] D. Shahaf and C. Guestrin, "Learning thin junction trees via graph cuts," *J. Mach. Learning Res.— Proc. Track*, vol. 5, pp. 113–120, 2009.
[20] M. Narasimhan and J. Bilmes, "PAC-learning bounded tree-width graphical models," in *Proc. UAI*, 2004, pp. 410–417.
[21] G. Elidan and S. Gould, "Learning bounded treewidth Bayesian networks," in *NIPS*, 2008, pp. 417–424.
[22] S. I. Lee, V. Ganapathi, and D. Koller, "Efficient structure learning of Markov networks using L1-regularization," in *Proc. NIPS*, Cambridge, MA, 2007, pp. 817–824.
[23] C. P. de Campos, Z. Zeng, and Q. Ji, "Structure learning of Bayesian networks using constraints," in *Proc. ICML*, 2009.
[24] S. Geman, D. F. Potter, and Z. Chi, "Composition systems," *Quart. Appl. Math.*, vol. 60, no. 4, pp. 707–736, 2002.
[25] S. C. Zhu and D. Mumford, "A stochastic grammar of images," *Found. Trends Comp. Graph. Vision*, vol. 2, no. 4, pp. 259–362, 2006.
[26] Y. Amit and A. Trouvé, "POP: Patchwork of parts models for object recognition," *Int. J. of Comput. Vision*, vol. 75, no. 2, pp. 267–282, Nov. 2007.
[27] A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West, "Sparse graphical models for exploring gene expression data," *J. Multivar. Anal.*, vol. 90, pp. 196–212, July 2004.
[28] J. Utans, "Learning in compositional hierarchies: Inducing the structure of objects from data," in *NIPS 6*, 1994, pp. 285–292.
[29] R. D. Rimey and C. M. Brown, "Control of selective perception using Bayes nets and decision theory," *Int. J. Comput. Vision*, vol. 12, pp. 173–207, April 1994.
[30] B. Neumann and K. Terzic, "Context-based probabilistic scene interpretation," in *Artificial Intell. in Theory and Practice III*, ser. IFIP Adv. in Inform. and Commun. Tech. Springer Boston, 2010, vol. 331, pp. 155–164.
[31] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency networks for inference, collaborative filtering, and data visualization," *J. Mach. Learning Res.*, pp. 49–75, 2000.
[32] Y. Xiang, F. V. Jensen, and X. Chen, "Multiply sectioned Bayesian networks and junction forests for large knowledge-based systems," *Comp. Intell.*, vol. 9, pp. 680–687, 1993.
[33] D. Koller and A. Pfeffer, "Object-oriented Bayesian networks," in *Proc. UAI*, 1997, pp. 302–313.
[34] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," in *Proc. IJCAI*, 1999, pp. 1300–1309.
[35] E. Gyftodimos and P. A. Flach, "Hierarchical Bayesian networks: An approach to classification and learning for structured data," in *Methods and Applications of Artificial Intelligence*, ser. Lecture Notes in Computer Science, G. A. Vouros and T. Panayiotopoulos, Eds., 2004, vol. 3025, pp. 291–300.
[36] D. Pe'er, "Bayesian network analysis of signaling networks: A primer," *Sci. STKE*, vol. 2005, no. 281, p. pl4, 2005.
[37] P. Spirtes, C. Glymour, and R. Scheins, *Causation, Prediction, and Search*, 2nd ed. MIT Press, 2001.
[38] T. Jaakkola, D. Sontag, A. Globerson, and M. Meila, "Learning Bayesian network structure using LP relaxations," in *Proc. AISTATS*, vol. 9, 2010, pp. 358–365.
[39] E. Segal, D. Koller, N. Friedman, and T. Jaakkola, "Learning module networks," in *J. Mach. Learning Res.*, 2005, pp. 525–534.
[40] A. Martins, N. Smith, and E. Xing, "Concise integer linear programming formulations for dependency parsing," in *Proc. ACL-IJCNLP*, 2009, pp. 342–350.
[41] S. S. Wilks, "The large-sample distribution of the likelihood ratio for testing composite hypotheses," *Ann. Math. Statist.*, no. 9, pp. 60–62, 1938.
[42] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation and traveling salesman problems," *J. Assoc. for Computing Machinery*, vol. 7, pp. 326–329, 1960.
[43] S. Della Pietra, V. Della Pietra, and J. Lafferty, "Inducing features of random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 380–393, Apr. 1997.
[44] D. Karger and N. Srebro, "Learning Markov networks: Maximum bounded tree-width graphs," in *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, 2001.
[45] N. Srebro, "Maximum likelihood bounded tree-width Markov networks," *Artificial Intell.*, vol. 143, pp. 123–138, 2003.
[46] K.-U. Höffgen, "Learning and robust learning of product distributions," in *Proc. COLT*, 1993, pp. 77–83.
[47] J. Schäfer and K. Strimmer, "An empirical Bayes approach to inferring large-scale gene association networks," *Bioinformatics*, vol. 21, no. 6, pp. 754–764, 2005.
[48] D. Heckerman, "A tutorial on learning Bayesian networks," Microsoft Research, Tech. Rep. MSR-TR-95-06, March 1995.
[49] K. Lang, "Newsweeder: Learning to filter netnews," in *Proc. ICML*, 1995, pp. 331–339.
[50] B. Vogelstein, D. Lane, and A. J. Levine, "Surfing the p53 network," *Nature*, no. 408, 2000.
[51] A. J. Levine, C. A. Finlay, and P. W. Hinds, "P53 is a tumor suppressor gene," *Cell*, vol. 116, pp. S67—S70, 2004.
[52] M. S. Greenblatt, W. P. Bennett, M. Hollstein, and C. C. Harris, "Mutations in the p53 tumor suppressor gene: Clues to cancer etiology and molecular pathogenesis," *Cancer Res.*, vol. 54, no. 18, pp. 4855–4878, 1994.
[53] A. Petitjean, E. Mathe, S. Kato, C. Ishioka, S. V. Tavtigian, P. Hainaut, and M. Olivier, "Impact of mutant p53 functional properties on TP53 mutation patterns and tumor phenotype: Lessons from recent developments in the IARC TP53 database," *Human Mut.*, vol. 28, no. 6, pp. 622–629, 2007.

Francisco Sánchez-Vega graduated in Telecommunications Engineering in 2005 from ETSIT Madrid and ENST Paris. Also in 2005, he was awarded a M.Res. in Applied Mathematics for Computer Vision and Machine Learning from ENS Cachan. He arrived at Johns Hopkins University in 2006 and received a M.Sc.Eng. in Applied Mathematics and Statistics in 2008. He is currently a Ph.D. candidate at this same department and a member of the Center for Imaging Science and the Institute for Computational Medicine at JHU.

Jason Eisner holds an A.B. in Psychology from Harvard University, a B.A./M.A. in Mathematics from the University of Cambridge, and a Ph.D. in Computer Science from the University of Pennsylvania. Since his Ph.D. in 2001, he has been at Johns Hopkins University, where he is Associate Professor of Computer Science and a core member of the Center for Language and Speech Processing. Much of his research concerns the prediction and induction of complex latent structure in human language.

Laurent Younes Former student of the Ecole Normale Superieure in Paris, Laurent Younes was awarded the Ph.D. from the University Paris Sud in 1989, and the thesis advisor certification from the same university in 1995. He was a junior, then senior researcher at CNRS (French National Research Center) from 1991 to 2003. He is now professor in the Department of Applied Mathematics and Statistics at Johns Hopkins University (that he joined in 2003). He is a core faculty member of the Center for Imaging Science and of the Institute for Computational Medicine at JHU.

Donald Geman received the B.A. degree in Literature from the University of Illinois and the Ph.D. degree in Mathematics from Northwestern University. He was a Distinguished Professor at the University of Massachusetts until 2001, when he joined the Department of Applied Mathematics and Statistics at Johns Hopkins University, where he is a member of the Center for Imaging Science and the Institute for Computational Medicine. His main areas of research are statistical learning, computer vision and computational biology. He is a fellow of the IMS and SIAM.

Learning Multivariate Distributions by Competitive Assembly of Marginals: Supplemental Material

Francisco Sánchez-Vega, Jason Eisner, Laurent Younes,
and Donald Geman.
Johns Hopkins University, Baltimore, MD, USA

1 Full Description of Simulation Study

We assess the performance of our learned models using synthetic data. We sample from a known model and attempt to recover both the joint probability distribution and the underlying graph structure. We compare our method with several alternatives from the literature, some designed only for learning graphs.

1.1 Analysis of the Learning Procedure

The first set of experiments is designed to measure the effect of sample size, number of variables and search strategy in the ideal case in which the true model, Q, belongs to our model class $\mathcal{F}^*$. The steps are:

(1) Generate at random an almost-balanced binary forest, F, on the set $D = \{1, \ldots, d\}$.

(2) Randomly select parameters, θ_0, to build a ground truth distribution Q. These parameters are chosen in such a way that all the primitives in the graph have associated ρ-values (computed analytically using θ_0) above a threshold η_0 (which is obtained using Eq. (13) from the main paper and a reference sample size $n_0 = 500$).

(3) Sample n d-dimensional binary vectors from Q.

(4) Using our CAM approach, induce a distribution $\hat{P} \in \mathcal{F}^*$ from the training data and compute its KL divergence from the ground truth

$$KL := E_Q \left[\log \frac{Q}{\hat{P}} \right]$$

The divergence can be computed exactly since the ground truth distribution is known.

This process is repeated 1,000 times for each experimental setting (choice of d, n, ϵ in the selection procedure), and the results are averaged to provide the curves in Figs. S.1 and S.2.

Fig. S.1 shows the average KL divergence for fixed $\epsilon = 1$ as a function of the sample size, n, ranging from $n = 16$ to $n = 2,048$. This, and all the other plots shown in our paper, show error bars that extend for a distance equal to one standard deviation above and below each average value. Different curves correspond to different choices of model dimension, d, and search strategy. Naturally, the divergence between $\hat{P}$ and Q decreases with sample size. For a fixed number of samples, the KL divergence increases with d. For this kind of relatively simple models, with small values of d, we observe that the ILP solution and the greedy search alternative provide very similar results, with a very slight improvement from ILP. From here on we shall only show results obtained with the ILP solution.

In Fig. S.2, we fix $d = 10$, and consider the effect of varying the selection threshold, ϵ, on the quality of estimation, again using KL divergence. We also evaluate the impact of knowing the true structure F, but still estimating the parameters, θ. Top left panel shows how choosing a very large selection threshold results in model overfitting. In cases like this, where the number of variables is relatively small, the ILP search method always finds the optimal solution (i.e. the one that maximizes the global sum of ρ-values), hence minimizes the KL divergence to the empirical distribution P^*. However, when the number of observed samples is small and ϵ is too large, this is not necessarily the same as finding the best approximation to the true distribution Q that generated the data. Similarly, when the number of observed samples is small, the true structure that generated the data may not be the one that leads to the best approximation to the

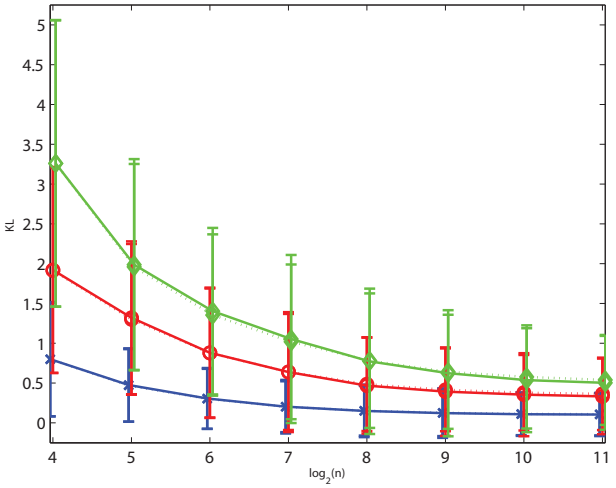

Fig. S.1. Evolution of KL divergence as a function of the sample size (in $\log_2$ scale) for fixed $\epsilon = 1$ and different choices of the number of variables: $d = 5$ (blue, cross); $d = 10$ (red, circle); $d = 15$ (green, diamond). Solid lines correspond to the ILP solution and dotted lines correspond to greedy search. The average value of the entropy $H(Q)$ is equal to 2.97 bits, 4.90 bits and 9.00 bits for $d = 5$,$d = 10$ and $d = 15$ respectively. Each curve represents an average over 1,000 random choices of Q and error bars show one standard deviation below and above the mean.

true generating distribution (hence the negative values of the curves in the top right panel for sufficiently small values of ϵ and n).

Fig. S.2 also shows how larger sample sizes lead to improved edgewise network reconstruction accuracy. Our edge comparison is based on the extended undirected graph in which an edge is added between sibling nodes (since our model does not assume that they are conditionally independent given their parent). We measure the recall or true positive rate (TPR), which is the fraction of edges in the ground truth networks that appear in the (undirected) learned graph; the false positive rate (FPR), which is the fraction of non-edges in the ground truth that appear in the learned graph; and the precision (fraction of recovered edges which are true). We provide both ROC and precision-recall (PR) curves.

1.2 Comparison with Other Methods

We now compare the performance of our CAM algorithm to other methods from the literature on reverse engineering networks using the two criteria considered above. We will do this both in the favorable case in which $Q \in \mathcal{F}^*$ and the unfavorable case in

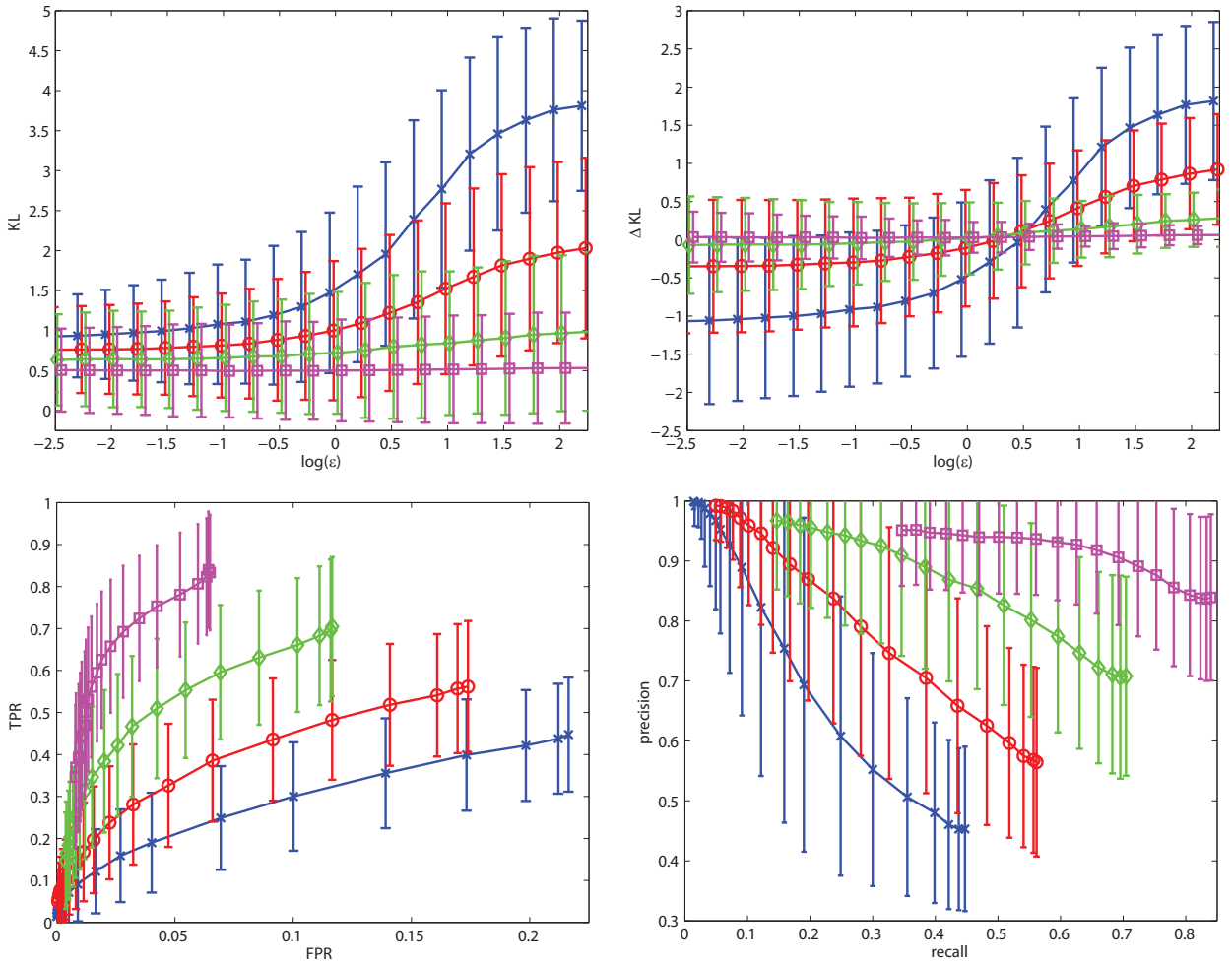

Fig. S.2. Top left: evolution of the KL divergence for $d = 10$ as a function of $\log(\epsilon)$ for different sample sizes ($n = 25$ in blue and cross marker, $n = 50$ in red and circular marker, $n = 100$ in green and diamond marker, $n = 200$ in magenta and square marker). Top right: difference between the estimated KL divergence using full model estimation and the KL divergence obtained assuming that the true structure F is known. Bottom: ROC (left) and PR (right) curves for structure estimation. Each curve represents an average over 1,000 random choices of Q.

which it does not, and our method cannot learn a graph structure as rich as F. We will make comparisons with Bayesian networks [1], relevance networks (RN) [2], ARACNE [3] and CLR [4].

RN, ARACNE and CLR learn network topologies only, i.e., they do not induce a probability distribution over the variables of interest. Consequently, our comparison is limited to the accuracy of edge reconstruction. Moreover, these methods do not

learn directed edges, so all comparisons will be made using the underlying undirected graphs for all the approaches. From the structure learning point of view, RN can be seen as a simplified version of our primitive selection method, where only binary primitives are considered and all the topological constraints on the graph structure are ignored. ARACNE goes one step further by inspecting all the three-cliques and using the triangular information inequality to prune spurious edges. CLR also prunes relevance networks by scoring each edge based on its z-score relative to the mutual information of edges with which it shares at least one node. Experiments with CLR used the code from [4].

The K2 algorithm [5] learns a Bayesian network. It is a heuristic structure-search method based on the use of the Bayesian score. Its main drawback is that it assumes that an ordering of the variables is known, $\{X_{(1)}, ..., X_{(d)}\}$, such that $X_{(i)}$ cannot be a parent of $X_{(j)}$ if $i > j$. Of course, in practice the causal ordering of the variables is typically unknown, which is a very important handicap. On the other hand, this method is relatively easy to implement and scales well to a reasonably large number of variables. Here, we used it just for benchmarking purposes as a generic representative of the Bayesian networks general family of models, and we used the ordering of the ground truth. We worked with the Matlab implementation by Guangdi Li (update of August 2009), available through the Mathworks File Exchange website.

1.2.1 Comparison for Binary Tree Models

First, the ground truth is generated as in the previous simulations. In order to provide a fair comparison, we assume that the true causal ordering of the variables is known, both for K2 and our algorithm; this information is not applicable to RN, CLR and ARACNE because they learn undirected graphs. We avoid the need to fine-tune the thresholds used by these last three approaches by ensuring that they always learn roughly the same number of edges as our tree models. For example, if the graph we learn for a given choice of our selection threshold, ϵ, contains k edges, we will learn the corresponding RN by keeping the top k edges in terms of pairwise mutual information (and similarly for CLR and ARACNE).

Fig. S.3 shows the average KL divergence between the learned joint probability distribution and the ground truth distribution for both K2 and CAM. The approximation obtained using CAM is clearly better. This is particularly evident for small sample sizes (in fact, when sample sizes are small enough, our approximation is better than the one learned using the true structure F).

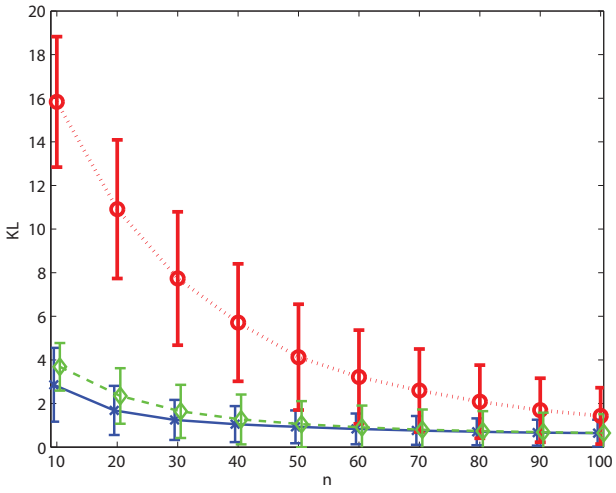

Fig. S.3. Evolution of KL divergence for small sample sizes (linear scale) between the true distribution Q and the distributions learned using CAM (solid, blue, cross), using K2 (dotted, red, circle) and using the true generating graph and simply estimating parameters (dashed, green, diamond). Each curve represents an average over 1,000 random choices of Q.

Fig. S.4 compares network reconstruction accuracy for the same simulation. In this case, we fixed $d = 10$ and $n = 100$ and the curves were drawn by choosing different values for the statistical threshold used by each approach. Results for K2 were averaged and shown as a single point (with vertical and horizontal error bars showing one standard deviation at each side of the mean in each dimension), since there were not any parameters to tune. The performance of CAM is comparable to that of RN, CLR and ARACNE. For small enough values of the threshold, CAM actually outperforms these alternatives (which is not surprising, since we are in the favorable case where the ground truth belongs to our constrained family of models). K2 always learns a relatively large number of edges, which results in a high average level of recall, but also in an average level of precision which is below the range of values observed for CAM.

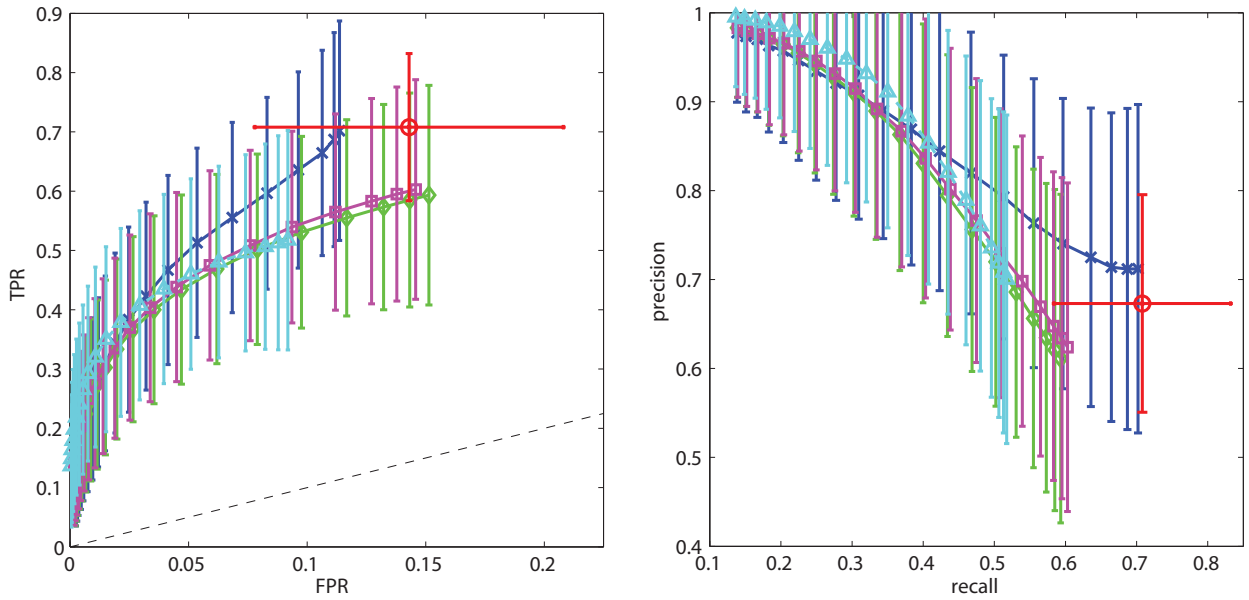

Fig. S.4. Comparison of (undirected) edge recovery accuracy. Curves correspond to CAM (blue,cross), relevance networks (green, diamond), CLR (magenta, square) and ARACNE (cyan, triangle). Results for K2 are shown as a single encircled red dot. The black dashed line on the left panel corresponds to random guessing. Results are shown for $d = 10$, $n = 100$ and were averaged over 1,000 replicates.

1.2.2 Comparison for a Generic Bayesian Network

We now assess performance when the data are generated from a more general Bayesian network. The ground truth network has $d = 14$ variables and is shown in Fig. S.5. The parametrization of the corresponding distribution is as follows.

- For the two root nodes, we fix $P(X_1 = 1) = 0.4$ and $P(X_2 = 1) = 0.6$.
- For each non-root node s with parent set s^-,

$$P(X_s = 1 | X_t = x_t, t \in s^-) = \frac{\exp\left(\alpha \cdot \sum_{t \in s^-} x_t\right)}{1 + \exp\left(\alpha \cdot \sum_{t \in s^-} x_t\right)}$$

Evidently, small values of α correspond to weak dependence (with complete decoupling for $\alpha = 0$) whereas as α becomes large, nodes with any "on" parent are likely to be "on" themselves.

We considered a range of sample sizes from 20 to 2,000 and we fixed $\epsilon = 1$. Fig. S.6 shows the number of edges in the final learned structures (averaged over 100 replicates). As expected, the structures learned by CAM contain fewer edges than their K2 counterparts. Indeed, CAM cannot learn more edges than variables (the slightly

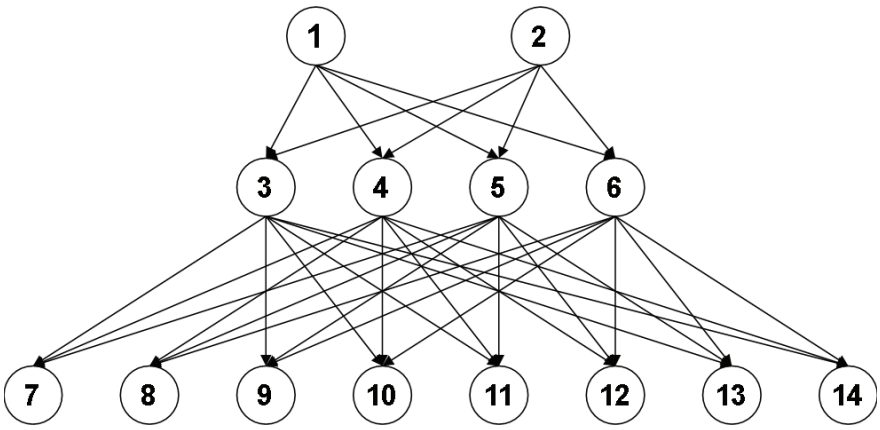

Fig. S.5. Graph structure for Bayesian network experiment.

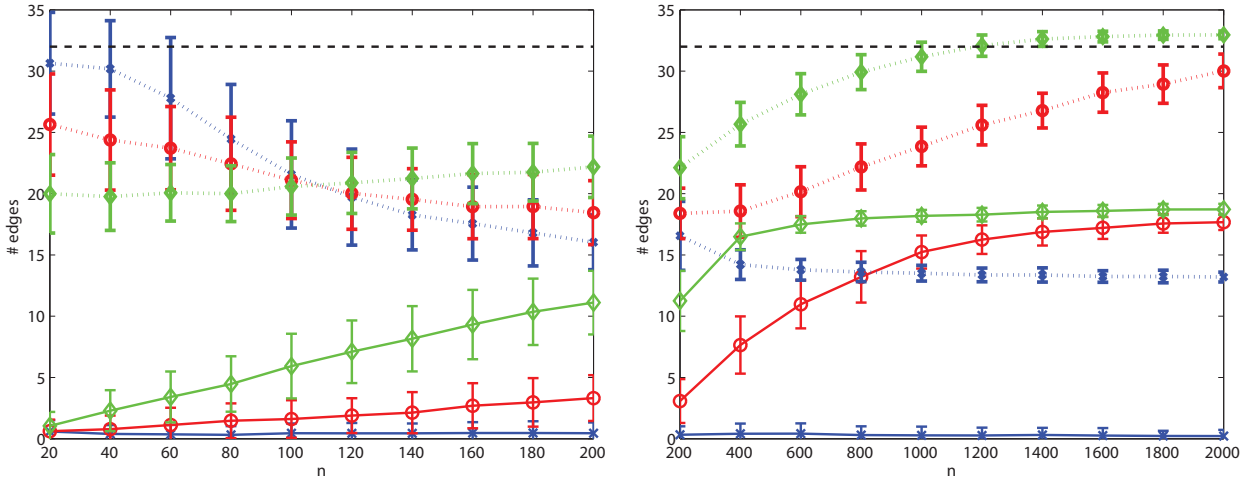

Fig. S.6. Average number of edges learned as a function of sample size. The black dashed line is the number of edges (32) in the true Bayesian network. Results are shown for different degrees of model dependence: $\alpha = 0$ (blue, cross), $\alpha = 0.5$ (red, circle) and $\alpha = 1$ (green, diamond). Solid lines correspond to CAM and dotted lines correspond to K2.

higher values seen in the figure are due to our use of the extended graph, where an edge is added between sibling nodes, for comparison purposes). For small sample sizes CAM learns a small number of edges, eventually saturating at the maximum number that can be learned. In contrast, K2 returns more complex structures (which do not necessarily correspond to the true one). For complete mutual independence ($\alpha = 0$), K2 stabilizes at around 14 edges, whereas CAM correctly learns nearly none.

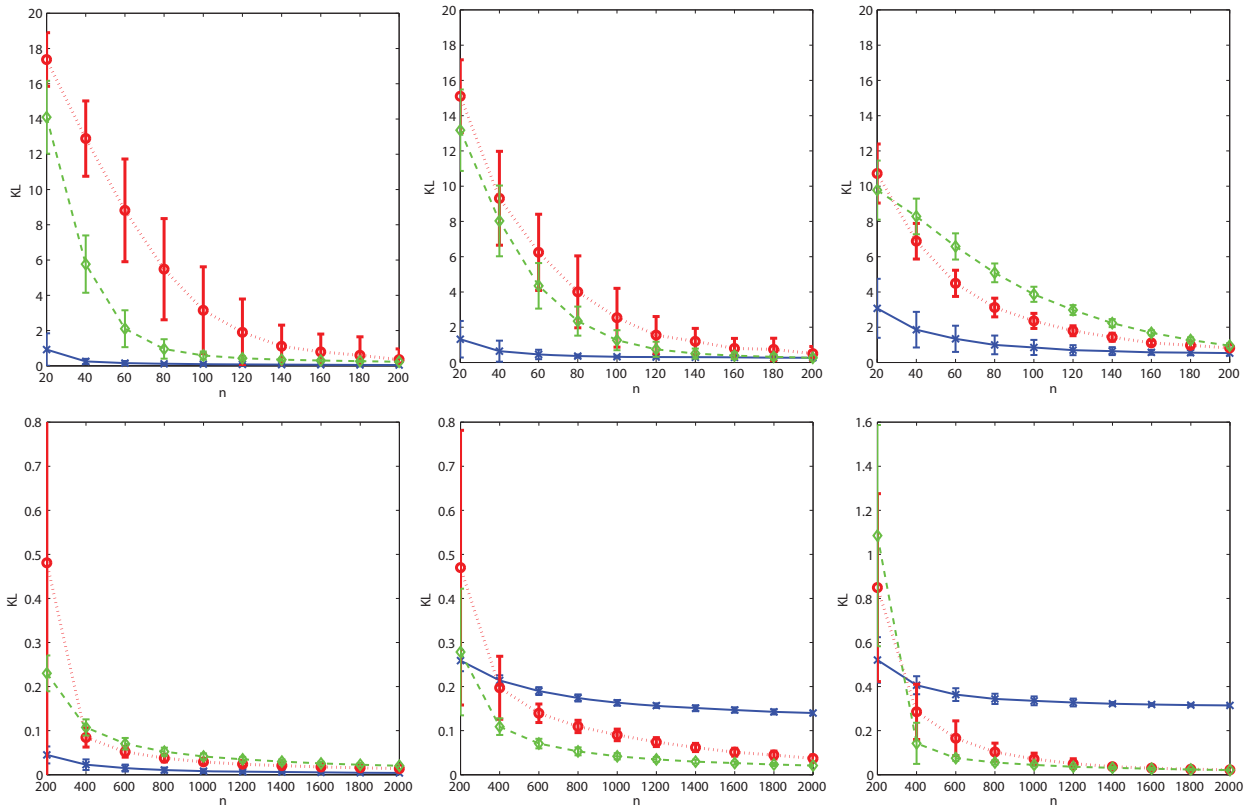

Fig. S.7. KL divergence between the Bayesian network ground truth distribution and the distributions learned using CAM (solid, blue, cross), K2 (dotted, red, circle) and the true generating graph with MLE parameters (dashed, green, diamond). Results are presented for $\alpha = 0$ (left column), $\alpha = 0.5$ (center column) and $\alpha = 1$ (right column). Results were averaged over 100 replicates.

Fig. S.7 shows the KL divergence to the Bayesian network ground truth distribution for both methods. In all cases, CAM offers the best performance when sample sizes are small by favoring bias over variance. For larger sample sizes, and aside from the case of weak dependence, CAM performs worse than the alternatives. This was expected: when data are plentiful and the dependency structure is rich, learning models more complex than ours becomes feasible.

Figure S.8 shows a precision-recall analysis of the results from the same simulation, but this time we include a comparison with the graphs learned using RN, CLR and ARACNE.

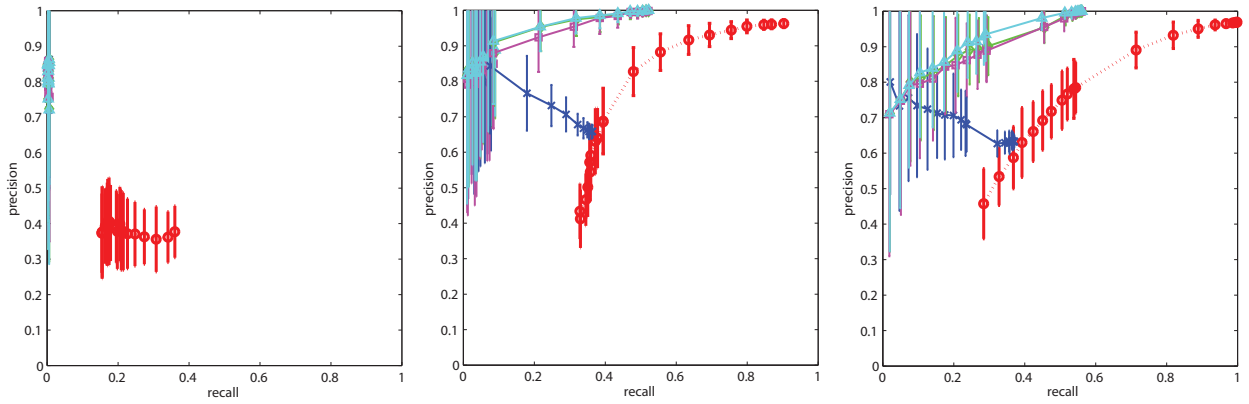

Fig. S.8. Precision-recall curves for the generic Bayesian network simulation. Curves correspond to CAM (blue line, cross marker), K2 (red dotted line, circle marker), RN (green line, diamond marker), CLR (magenta line, square marker) and ARACNE (cyan line, triangle marker). Results are presented for $\alpha = 0$ (left), $\alpha = 0.5$ (middle) and $\alpha = 1$ (right). Results were averaged over 100 replicates.

The points on each curve were drawn by using different sample sizes (actually, the same as in the horizontal axes of Fig. S.7) instead of the traditional approach where different points are obtained by varying some learning threshold. This is why they may appear counterintuitive at first sight: as the sample size increases, we may learn more edges and a larger percentage of all the learned edges may be correct, so precision and recall may increase simultaneously.

For $\alpha = 0$, the curves are rather chaotic and the points do not show a clear spatial trend. This is not surprising, since this choice of parametrization implies complete joint independence among the variables and no structure can be recovered. As α grows, the curves exhibit a better defined monotone increasing pattern. In all cases, we observe that for similar levels of recall, the K2 approach achieves the smallest edgewise precision. RN, CLR and particularly ARACNE seem to offer the best precision-recall performances. For very small samples, which include the first few points in the curves starting from the left, CAM offers a precision-recall performance that is competitive with that of RN, CLR and ARACNE. For larger sample sizes (over 100 samples), CAM exhibits a lower level of precision than RN/CLR/ARACNE (always for a same level of recall), although it consistently outperforms K2 (when remaining within CAM's strongly limited range of recall).

2 Size of Encoding and Empirical Runtimes for the ILP Solution to Structure Search

The sets of constraints that we had described in Section 5.2 of our paper to define the ILP search procedure can be summarized as follows:

(C1) $\forall e \in \mathcal{E}, \quad \sum_{t \in \mathcal{T}_e} x_t = y_e$

(C2) $\forall \psi \in \mathcal{T}_0, \quad (|\psi| - 1)x_\psi \leq \sum_{e \in \psi} y_e$

(C3) $\forall e \in \mathcal{E}, \quad y_e + y_{\bar{e}} \leq 1$

(C4) $\forall v \in V, \quad \sum_{(v',v) \in \mathcal{E}} y_{(v',v)} \leq 1 \quad \text{and} \quad \sum_{(v,v') \in \mathcal{E}} y_{(v,v')} \leq 2.$

(C5) $\forall v \in V, \quad \sum_{\psi \in \Psi_v} x_\psi \leq 1$

(C6) $\forall e \in \mathcal{E},$

$$\begin{cases} -C(1 - y_e) + y_e + \sum_{e' \to e} f_{e'} \leq f_e \leq y_e + \sum_{e' \to e} f_{e'} \\ 0 \leq f_e \leq C y_e \end{cases}$$

(C7) $\forall e \in \mathcal{E},$

$$\begin{cases} 0 \leq h_e \leq y_e \\ h_e \leq 1 - y_{e'} \text{ if } e \to e' \\ h_e \geq 1 - \sum_{e \to e'} y_{e'} - C(1 - y_e) \\ -C(1 - y_e) + \sum_{e \to e'} g_{e'} \leq g_e \leq h_e + \sum_{e \to e'} g_{e'} \\ y_e \leq g_e \leq C y_e \end{cases}$$

$\forall \psi \in \mathcal{T}_{0,T}, \quad |g_{e(\psi)} - g_{e'(\psi)}| \leq 1 + C(1 - x_\psi)$

In the last constraint, $\mathcal{T}_{0,T}$ refers to the subset of primitives with cardinality three (i.e. the subset of triplets in $\mathcal{T}_0$).

Note that, as mentioned in the main paper, our choice of ILP solver for all the synthetic simulations and real-data data experiments presented here was the Gurobi optimizer (version 4.5).

2.1 Size of ILP Encoding

Tables 1 and 2 show the number of variables in the ILP problem and the number of linear constraints associated to each of the conditions above, respectively. Note that, as

we had explained in Section 5.2, condition C3 is made redundant by condition C6 and can therefore be omitted, although in practice we have observed that it helps to speed up the solver.

Name	Description	Cardinality
x_t	Primitive selector	$\lvert\mathcal{T}_0\rvert$
y_e	Edge selector	$\lvert\mathcal{E}\rvert$
f_e	Acyclic flow	$\lvert\mathcal{E}\rvert$
h_e	Terminal edge indicator (used in balance flow)	$\lvert\mathcal{E}\rvert$
g_e	Counter of terminal edge descendants (used in balance flow)	$\lvert\mathcal{E}\rvert$

TABLE 1

Size of ILP encoding: number of variables.

The total number of variables in the ILP problem is a linear combination of $|\mathcal{E}|$ and $|\mathcal{T}_0|$ of the form:

$$4 \cdot |\mathcal{E}| + |\mathcal{T}_0|$$

Condition	Motivation	# constraints
C1	Every selected edge must appear in exactly one selected primitive	$\lvert\mathcal{E}\rvert$
C2	All edges in each selected primitive must be selected	$\lvert\mathcal{T}_0\rvert$
C3	No edge and its reversal can be simultaneously selected	$\lvert\mathcal{E}\rvert$
C4	No vertex can have more than one parent and two children	$2\lvert V\rvert$
C5	No α-node overlap for binary primitives	$\lvert V\rvert$
C6	Graph must be acyclic	$4\lvert\mathcal{E}\rvert$
C7	Graph must be almost-balanced at triplets	$7\lvert\mathcal{E}\rvert + \lvert\mathcal{E}\rvert^* + 2\lvert\mathcal{T}_{0,T}\rvert$

TABLE 2

Size of ILP encoding: number of constraints.

The total number of constraints in the ILP problem depends on a combination of $|\mathcal{E}|$, $|V|$, and $|\mathcal{T}_0|$ of the form:

$$13 \cdot |\mathcal{E}| + |\mathcal{E}|^* + 3|V| + |\mathcal{T}_0| + 2|\mathcal{T}_{0,T}|$$

where the term $|\mathcal{E}|^*$ is used to represent the number of constraints associated to the second inequality in C7. For each edge $e = (a, b) \in \mathcal{E}$, this inequality introduces a linear constraint for every edge $e' = (b, c) \in \mathcal{E}$. Therefore, the actual number of constraints will depend on the number of overlapping edges contained in $\mathcal{E}$. The following upper bound is always valid as a "worst case scenario":

$$|\mathcal{E}|^* \leq |\mathcal{E}| \cdot (|V| - 1)$$

Notice that $|\mathcal{E}|$ is also an upper bound for the constraints associated to C3, since in practice it is only necessary to impose this constraint for edges $(a, b) \in \mathcal{E}$ such that $(b, a) \in \mathcal{E}$, and not for every edge in $\mathcal{E}$. Similarly, V is an upper bound for the number of constraints in C5, since there may be nodes that do not appear as α-nodes in any of the primitives contained in $\mathcal{T}_0$ (or that appear in only one of them).

The expressions above for the number of variables and the number of constraints show that the complexity of the ILP problem depends mainly on the properties of the set of candidate primitives that survive the initial stepwise selection process. On the one hand, $|\mathcal{T}_0|$ will be large when there is a large number of significant dependencies among the variables. It is possible to encounter situations where $|V| = d$ is relatively small and yet $|\mathcal{T}_0|$ is relatively large (e.g. the dataset contains few variables but the target network of dependencies is very dense, as is the case for our *20newsgroups* experiment) and viceversa (e.g. the dataset contains many variables but the target network of dependencies is very sparse, as might well be the case for our TP53 experiment). On the other hand, $|\mathcal{E}|$ will be large when those dependencies involve many different edges (i.e. oriented pairs). Again, note that $|\mathcal{E}|$ is not necessarily a monotone function of $|\mathcal{T}_0|$, since there can be relatively large sets of primitives built by reusing a relatively small set of edges. Similarly, $|\mathcal{E}|^*$ needs not be a monotone function of $|\mathcal{E}|$, since there can be large sets of edges with very few overlaps or even no overlaps at all.

2.2 Empirical ILP Runtimes from Synthetic Simulations

We carried out an empirical evaluation of the runtimes for our ILP search approach and we compared it to several alternatives based on our synthetic data simulations. Results are shown in Fig. S.9.

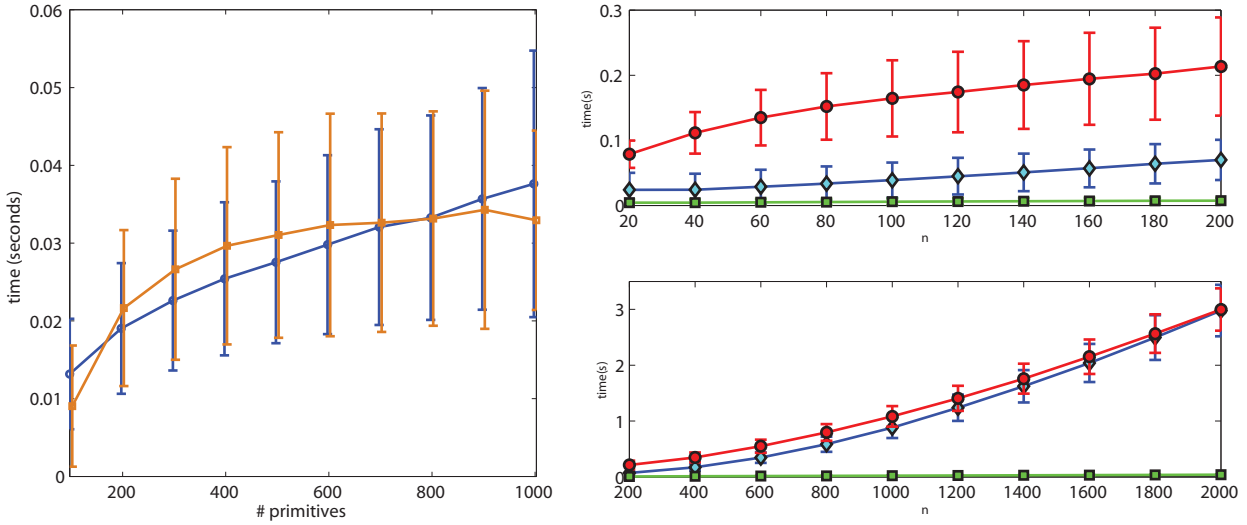

Fig. S.9. Average runtime results for synthetic data simulations. Left panel shows average runtimes as a function of $|\mathcal{T}_0|$ for random choices of structure and parameterization (learned as in the experiment from Fig. S.2, for a fixed choice of $d = 10$, $n = 10$, and averaged over $20e3$ replicates). Results are shown for greedy search (yellow, diamond marker) and ILP search (blue, circle marker). Right panels show average runtimes for the generic Bayesian network experiment described in Section 1.2 of this supplemental material. Results are shown for ILP search (blue), K2 search (red) and RN (green). The values on the horizontal axis correspond to different samplesizes.

Left panel shows average runtimes as a function of $|\mathcal{T}_0|$ for greedy search versus ILP search. Within the same experimental setting that we had used to generate the results shown in Fig. S.2, we generated random structures with random parameterizations involving $d = 10$ variables and for each of them we generated $n = 100$ samples. By varying the selection threshold, we obtained in each case $|\mathcal{T}_0|$ of different cardinalities. We applied greedy search and ILP search to look for the best structure and we measured the runtimes for each approach as a function of $|\mathcal{T}_0|$ (the averages were computed using bins of size 100 primitives). In this experiment, ILP was always run to convergence. The plot shows how greedy search runtimes grow faster for small sets of primitives and then tend to stabilize, while ILP runtimes exhibit a more linear behavior. This confirms what

we had observed in practice: when the number of candidate primitives is relatively small, ILP search is competitive with greedy search in terms of speed, sometimes it is even faster. For large numbers of primitives, the computational cost of ILP continues to grow linearly while the cost of greedy search stabilizes, which results in ILP needing much runtimes.

Finally, the last two panels on the right side of Figure S.9 show the time averages for the second experiment of Section 1.2 of this supplemental material (for the case $\alpha = 0.5$), where we compared our CAM-ILP approach to K2 and RN/ARACNE/CLR using a generic Bayesian network. Since the dimension of the problem was relatively small ($d = 14$), the runtimes for ARACNE and CLR were almost identical to the runtimes for RN, which roughly corresponded to the time required to compute and sort all the pairwise values of mutual information. These are shown in green in the figure. Of course, these average runtimes for RN are much smaller than the runtimes for K2 and ILP search. Once again, we observe that ILP runs faster than K2 for small samples, although their time requirements tend to equalize as the observed sample size grows.

2.3 Performance Analysis of the ILP Solution for our Real Data Experiments

Table 3 presents the actual values of $|V|$, $|\mathcal{T}_0|$ and $|\mathcal{E}_0|$ for our *20newsgroups* and TP53 experiments. It also includes the number of variables in the dataset (d) and our choice of selection threshold (ϵ).

Problem	d	ϵ	$\lvert V\rvert$	$\lvert\mathcal{T}_0\rvert$	$\lvert\mathcal{T}_{0,T}\rvert$	$\lvert\mathcal{E}_0\rvert$	$\lvert\mathcal{E}_0\rvert^*$
20newsgroups	66	10^{-6}	66	13,820	12,994	1,753	57,586
TP53	2,000	1	76	760	626	290	1,940

TABLE 3

Size of ILP encoding for the *20newsgroups* and the TP53 experiments.

The numbers of rows, columns and nonzero variables in each ILP problem, as well as the final values for the objective function, best bound and dual gap are shown in Table 4.

Problem	Rows	Columns	NonZeros	Final Score	BestBound	Gap
20newsgroups	88,164	20,327	543,712	0.72267	0.78455	8.56 %
TP53	5,288	1,680	35,811	0.04345	0.04345	0.0046 %

TABLE 4

Size of encoding (continued) and final results of ILP search for the *20newsgroups* and the TP53 experiments. These are the values reported by the Gurobi optimizer after the initial presolve step, which typically carries out some problem reductions.

As we had explained in Section 8.1 of the main paper, global optimality of the solution shown in Fig. 5 cannot be guaranteed because the final dual gap reached by the solver was non-negligible. The solution itself was first reached after approximately three and a half days of computation. At that point, the gap was 8.58%. The result appeared to be stable after extensive computation, meaning that no improvement in terms of the objective function for the primal problem was observed after running the Gurobi solver for several additional days on a 16 core machine. There was a small reduction of the bound based on the solution to the dual problem, which pushed the gap down to the 8.56% reported. The fact that this reduction was so small for such a large amount of additional computation suggests that, in situations like this, where the number of rows/columns/nonzeros is large, the gap may indeed be difficult to close. Note that this does not mean that this solution is not optimal, but only that its global optimality cannot be guaranteed based on the dual gap. In fact, in practice it may be possible to find a very good, possibly optimal or near-optimal solution relatively fast. Table 5 illustrates this point.

On the other hand, in Section 8.1, we had also mentioned that the final network that we had learned (shown in Fig. 5) is componentwise optimal. This means that, after learning the global network, we revisited each of the five independent components one by one and we run a separate ILP search restricted to the variables contained in that component. The difficulty of these problems varied a lot from one to another (from the 0.4 seconds required to solve the one for the *cars* category, to the more than 5 days

required to solve the one for the *computers* category) but they were all much more computationally affordable than the original global problem. Some additional details are provided in Table 6. For every one of these ILP subproblems, the solver reached a guaranteed optimal solution (for a gap threshold of 0.01%) which coincided in all cases with the structure that we had learned as a part of the global network.

Time	Objective Function	Best Bound	Gap
36 seconds	0.69141	-0.84677	22.5 %
~ 1 minute	0.70386	0.81512	15.8 %
~ 1 hour	0.71686	0.78854	10.0 %
~ 10 hours	0.72076	0.78589	9.04 %
~ 3.5 days	0.72267	0.78470	8.58 %
~ 6 days	0.72267	0.78455	8.56 %

TABLE 5

Temporal evolution of the global ILP search for the *20newsgroups* dataset. Greedy search reached a final objective function value of 0.6475 after 13.17 seconds.

Subproblem	Variables	Rows	Columns	NonZeros	Final Score	Time
Cars	5	340	129	1,379	0.0255	0.40
Sports	9	2,077	745	10,593	0.11552	4.80
Health	11	2,206	755	11,348	0.07160	7.33
Space and religion	18	6,711	2,238	31,722	0.26189	564
Computers	23	27,787	8,327	141,309	0.24817	436,292

TABLE 6

Size of ILP encoding for each of the five componentwise ILP search problems.

The solution for the ILP search problem in the TP53 case (which corresponds to the network shown in Fig. 6 and Fig. S.12) was found in 441.96 seconds. The solution is guaranteed to be optimal for a dual gap threshold of 0.01% (the solver stopped when an actual gap of 0.0046% was reached). As we had explained above, the fact that the

TP53 search problem is solved much faster than the *20newsgroups* case might seem counterintuitive at first sight, since $d = 2,000$ for TP53 and $d = 66$ for *20newsgroups*. However, the actual complexity of the search problem is determined by $\mathcal{T}_0$. Since there are many more primitives that survive the stepwise selection process for *20newsgroups*, the associated ILP problem becomes much harder to solve.

Finally, please note that, as we had explained at the end of Section 5 in the main paper, the ILP solution may improve upon greedy search even without running to convergence. This claim is supported by the following empirical evidence:

- For the *20newsgroups* dataset, our greedy search algorithm got a final score of 0.6475 after 13.17 seconds. The first ILP solution that improved this result was obtained after approximately 36 seconds and had a score of 0.6914 (with a gap of 22.5 %).
- For the TP53 experiment, our greedy search algorithm got a final score of 0.0395 after 12.43 seconds. The first ILP solution that improved this result was obtained in approximately 1 second and had a score of 0.4246 (with a gap of 12.2 %).

These results also show that the runtimes needed to find an ILP solution that improves the one obtained by greedy search are competitive with the empirical runtimes measured for the greedy search algorithm.

3 Extended discussion of our experiments using the *20newsgroups* dataset

In this section, we present an experiment based on the *20newsgroups* dataset where we show that, within small-sample regimes, the probability distributions learned using CAM provide better predictions on holdout samples than the distributions learned using the K2 algorithm and a baseline edgeless Bayesian network. We also present a comparison between the semantic network learned using CAM on the *20newsgroups* dataset and two networks learned using the relevance networks approach.

3.1 Quantitative Evaluation of the Predictive Performance of CAM Models on Hold-out Samples

We work with the *20newsgroups* dataset and we consider the same set of 66 variables and 16,242 samples that we had used to learn the network shown in Fig. 5. In this case, however, we will split the data into a training set and a learning set. We will use our family of CAM models to learn the joint probability distribution of the 66 variables using the training set and we will evaluate its performance at predicting the holdout samples by computing the average log-likelihood of the test set.

We will compare our results with the ones obtained using K2, since none of the other methods for network reconstruction discussed in our paper is designed to learn the underlying joint probability distribution. We will also compare our results with those associated to a baseline model (BL) consisting of an edgeless Bayesian network (i.e. a model which assumes full independence, so that the joint is simply computed as a product of all the individual marginals).

3.1.1 Experimental Setting

We want to evaluate the performance of the three methods mentioned in the previous paragraph as a function of the available sample size, n. For this, we propose the following procedure:

- At replicate i, do:
  1) Randomly choose a subset of N samples from the original set of $16,242$ observations. This will be our training set $\mathcal{L}^{(i)}$. All the remaining $16,242-N$ samples will be used as the test set $\mathcal{H}^{(i)}$ (or "holdout set") for replicate i.
  2) Index the N samples in the training set $\mathcal{L}^{(i)}$, so that $\mathcal{L}^{(i)} = \{\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_N^{(i)}\}$ (where each $\mathbf{x} \in \mathbb{R}^d$).
  3) For N_j from N_0 to N in increments of Δ_N, do:
     a) Let the current partial training set $\mathcal{L}^{(i,j)}$ contain the first N_j samples in the training set $\mathcal{L}^{(i)}$, i.e. $\mathcal{L}^{(i,j)} = \{\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_{N_j}^{(i)}\}$.
     b) Learn the joint probability distribution of $\mathbf{X}$ using BL, K2 and CAM.
     c) Compute the average log likelihood of the samples in the test set $\mathcal{H}^{(i)}$ under the joint distributions learned with each of the previous models:

$$LL(\mathcal{L}^{(i,j)}, \mathcal{H}^{(i)}) = \sum_{\mathbf{x}_k \in \mathcal{H}^{(i)}} \frac{\log P_{\mathcal{L}^{(i,j)}}(\mathbf{x}_k)}{|\mathcal{H}^{(i)}| \cdot d}$$

     where $|\mathcal{H}^{(i)}| = 16,242 - N$ is the number of holdout samples.

We repeated the steps above for a total of 100 replicates and averaged the final results in order to evaluate the performance associated to BL, K2 and CAM. The results are shown in Fig. S.10.

The K2 algorithm assumes knowledge of the causal ordering of the variables, i.e. it needs to be told which variables can be parents of which variables in the final structure. Of course, the true causal ordering in the current problem is unknown. We decided to run the K2 algorithm using a frequency (or, equivalently, entropy) ordering, according to which a variable can only be a parent of another one if the word associated to the parent appears in more documents than the word associated to the child. Note that, since all the frequencies are relatively small in this example (and far below 0.5), entropy is a monotone function of frequency and therefore the entropy and the frequency orderings coincide. In order to provide a fair comparison with CAM, we also restricted the space of candidate CAM structures to enforce the same constraint, i.e. we discarded all the primitives where the frequency of parents was not larger than the frequency of their children. Once again, we do not claim that this is a "true" causal ordering, but we can guarantee that the same ordering was enforced for the two approaches.

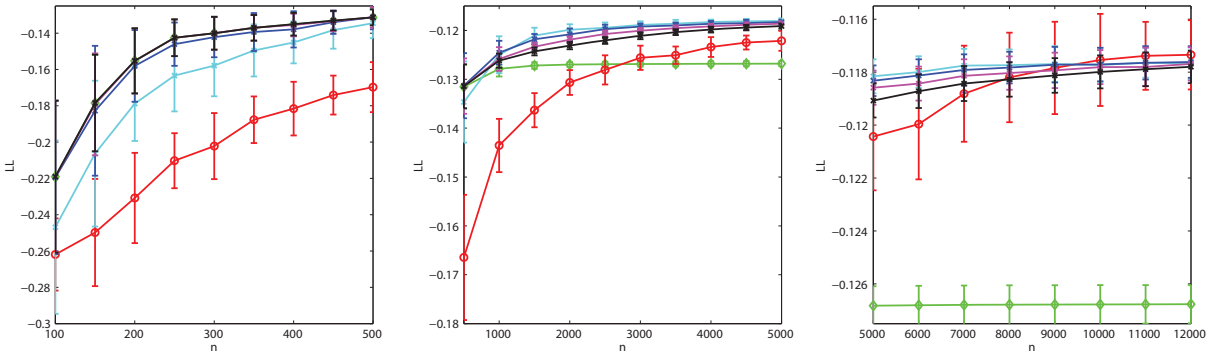

Fig. S.10. Average log-likelihood over holdout samples as a function of observed sample size. Results are shown for BL (green, diamond), K2 (red, circle) and CAM (cross marker). CAM results are shown for $\epsilon = 1$ (cyan), $\epsilon = 10^{-3}$ (blue), $\epsilon = 10^{-6}$ (magenta) and $\epsilon = 10^{-9}$ (black). The three panels cover different ranges of sample sizes. On the left panel, the green and the black lines overlap because the results for BL and CAM($\epsilon = 10^{-9}$) are very similar (identical for most sample sizes).

When it comes to our CAM models, attempting to find a guaranteed optimal solution for the ILP search problem may be computationally prohibitive for this given dataset, depending on the available sample size (which will influence the actual cardinality of $\mathcal{T}_0$, for more details see the discussion in Section 2.3 of this supplemental material). Still, a good solution can typically be found in a relatively short time. In order to illustrate this point, we imposed an upper bound of five minutes for each run of the ILP solver, meaning that a solution is accepted if either it is guaranteed to be optimal or if the time limit of 300 seconds is reached. In most cases, particularly those corresponding to small samples, the optimal solution was found for runtimes far below the five minute limit.

3.1.2 Discussion

Our results support our claim of the fact that CAM models perform well within small sample scenarios.

For sample sizes between 100 and 500 samples, the performance of CAM is comparable to that of BL. This makes sense: since the available sample size is so small relative to the number of variables, the model with the best generalization properties is the edgeless graph (let us keep in mind the fact that we are attempting to learn a distribution with a state space of 2^{66} configurations from just a few hundreds of observations). For

sufficiently small values of ϵ, CAM will simply "refrain" itself from learning any edges and thus the results are identical to those obtained with BL. A large choice of ϵ may lead to overfitting, as is the case for the cyan line in the figure. K2, however, always learns a relatively complex graph and therefore always ends up severely overfitting the data within this range of samples.

For sample sizes between 500 and 5,000, CAM clearly outperforms the other two approaches. BL is too simple to achieve a good performance. The performance of K2 improves as sample size grows, so that it does better than BL after approximately 2,000 samples. However, K2 still performs worse that CAM because the observed sample size is not enough to support the complexity of the models that it is attempting to learn. We remark that, for this range of sample sizes, CAM consistently outperforms its alternatives over a very wide range of choices of ϵ (from $\epsilon = 1$ to $\epsilon = 10^{-9}$).

Finally, when a sufficiently large set of samples is observed, K2 will overpass CAM (as shown in the third panel of Fig. S.10). This was expected and is the type of result that agrees with the conclusions that we had reached from our synthetic data simulations: when the number of samples in the training set is large enough, models that are more complex than CAM exhibit better learning performance.

3.2 Comparison with Relevance Networks

Fig. 5 in Section 8.1 shows the final network learned using CAM models for the whole *20newsgroups* dataset. For comparison purposes, we also looked at the kind of graphs that would be learned using standard relevance networks. Results are shown in Figure S.11.

We considered two different network building strategies in order to provide a fair comparison. On the one hand, the threshold for the mutual information was chosen to be the same as the one used to select binary primitives for the network in Fig. 5. On the other hand, we sorted all the pairwise mutual information values and only kept the top scorers, in order to learn a graph with the same number of edges as the one that we had learned using CAM. In the first case, all the variables appeared in the graph as members of a unique connected component. The large number of pairs that survived thresholding led to a very crowded network representation with lots of nodes

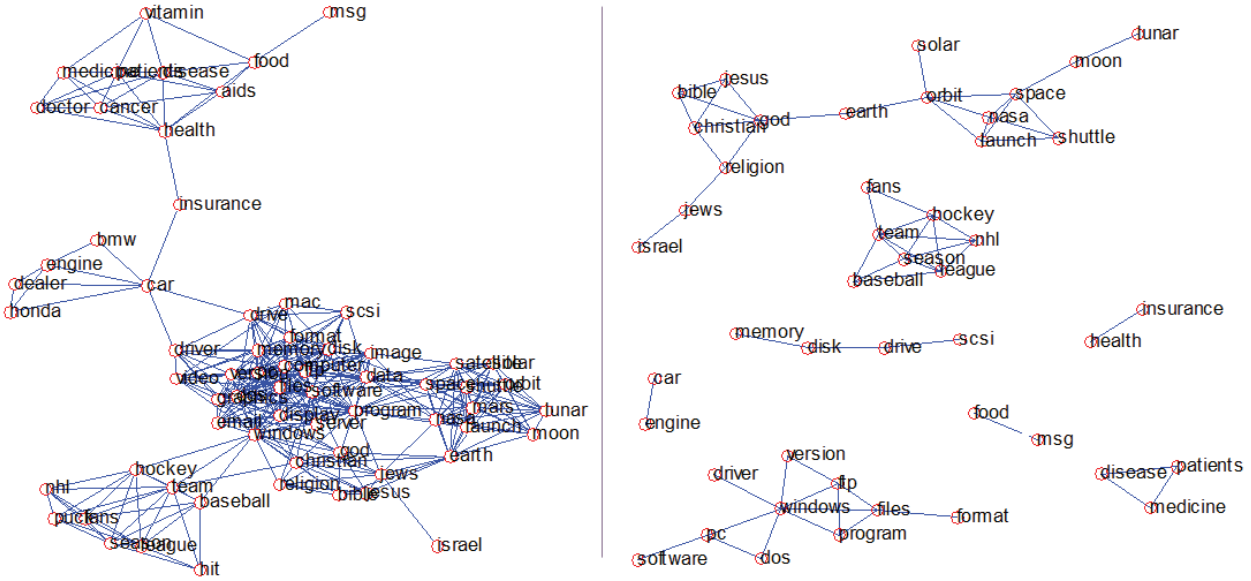

Fig. S.11. Two graphs learned by relevance networks using the *20newsgroups* dataset. Left panel: the mutual information threshold is the same as the threshold that was used for selecting binary primitives in the network shown in Figure 5 of the main paper. Right panel: the threshold is chosen so that the learned relevance network has the same number of edges as the network shown in Figure 5.

of high degree. Although the words are certainly clustered by semantic categories in terms of their proximity within the network, the large number of detected interactions makes it difficult to further interpret the results. In the case where only the top edges were kept, a total of 20 variables (out of the original 66) were left out of the final structure and treated as singletons (they were not linked to any others). This means that many meaningful interactions remained undiscovered, even if the multiple independent components seem to correspond well to different semantic categories. In either case, we must not forget that, unlike relevance networks, our balanced compositional trees provide a truly generative model based on an efficient parametrization of the target global joint probability distribution and this offers advantages that go beyond those associated to a merely more attractive visual representation.

4 Analysis of Results for the TP53 Experiment

The final network learned using our CAM approach on the full IARC TP53 dataset contained a total of 68 different mutations structured in several independent components. It is shown in Fig. S.12.

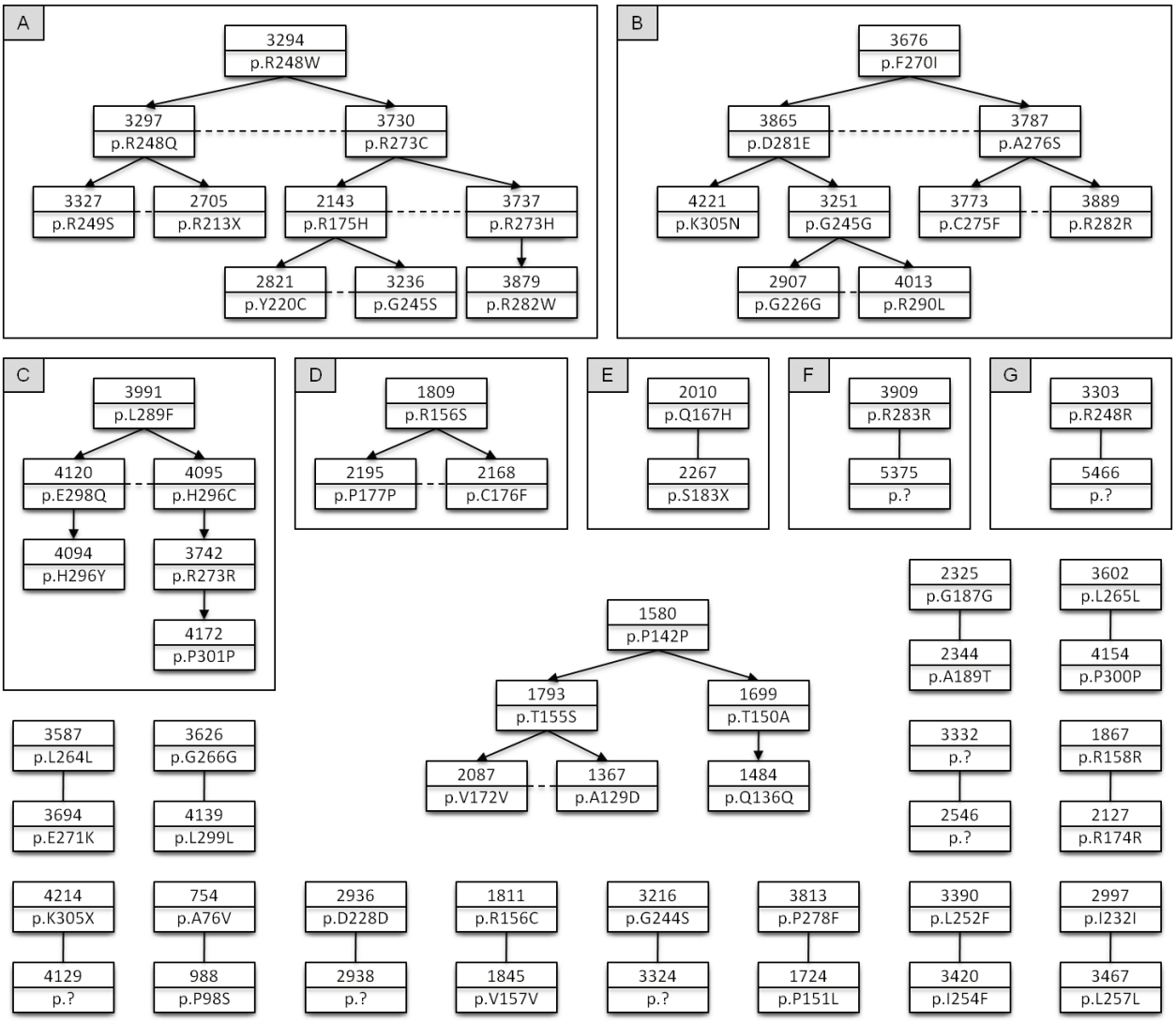

Fig. S.12. Network of interactions among somatic mutations for the IARC TP53 dataset learned using CAM models. Dashed lines are used to link siblings that belong to the same triplet primitive. For each node, we provide the unique mutation identifier in the IARC Database and the (standard) mutation description at the protein level, where *p.XzY* represents the substitution of amino acid X by amino acid Y at codon number z; for example, $p.R248W$ represents substitution of *Arg* by *Trp* at codon 248. Components labeled with a capital letter contain enriched annotations which are detailed in Table 7.

As we had explained in Section 8.2, each somatic mutation in the IARC Database is annotated with biochemical details about the actual nucleotide variation, as well as clinical information for the patient and tissue where the mutation was observed. The IARC Database also provides a unique nucleotide position for each mutation. The average distance between nodes within the same connected component in our network is 486.8 nucleotides, whereas the average distance among unconnected nodes (including singletons) is 1,436.5 nucleotides.

We first work with two types of annotation: *topography* and *morphology*. Topography refers to the site of the tumor in which the mutation was observed, as defined by organ or group of organs; there are 74 distinct labels and examples include *breast*, *brain*, *prostate* or *colon*. Morphology refers to the tumor type; there are 323 distinct labels and examples include *adenoma*, *malignant lymphoma* or *leukemia*.

We hypothesize that mutations which are close in our network are more likely to be functionally related. In particular, pairs of mutations linked by an edge should be more likely to share annotation than pairs of mutations chosen at random. It is easy to show that sharing at least 5 topography terms or at least 5 morphology terms has probability less than 0.05 for a random pair. In the learned network, of the fifty-eight edges in the graph, all connected pairs share at least one topography term and 14 share at least six. Similarly, all share at least one morphology term, and 16 out of the 58 share at least six. These are all rare events under random pairing.

We investigated enrichment of annotations at the component level, permuting mutation IDs and computing hypergeometric p-values. These p-values represent the probabilities of observing an equally large or larger number of mutations within a component that are associated with a given term under the null hypothesis of independent labels within a randomly-composed component of the same size. Using the fifth percentile for significance, we found that several components in our network contained significantly enriched annotations for topography and morphology. They are shown in Table 7, where the labels in the first column correspond to the labels in Fig. S.12 and the numbers in the second column correspond to the IARC mutation identifiers.

COMPONENT	MUTATIONS	TOPOGRAPHY	MORPHOLOGY
A	2143,2705,2821, 3236, 3294, 3297, 3327,3730,3737, 3879	*53 terms*	*98 terms*
B	2907,3251,3676, 3773,3787,3865, 3889,4013,4221	LUNG	ADENOCARCINOMA
C	3742,3991,4094, 4095,4120,4172	STOMACH	-
D	1809, 2168,2195	ADRENAL GLAND	GERMINOMA, YOLK SAC TUMOR, ASTROCYTOMA
E	2010,2267	THYROID	PAPILLARY CARCINOMA, MATURE T-CELL LYMPHOMA.
F	3909,5375	-	HEMANGIOSARCOMA
G	3303,5466	-	HEMANGIOSARCOMA

TABLE 7

Components with enriched topography and morphology annotations in TP53 network.

The component of size ten is significantly enriched for a large number of annotations, both in terms of topography and morphology. Interestingly, it contains the nine most frequent p53 mutations, which are well-known to be localized in seven mutation hotspots [6], [7]. The top eight most frequent mutations belong to the DNA-binding domain of the p53 protein (R248W, R248Q, R273C, R249S, R175H, R273H, G245S and R282W). Mutations within this region can result in the removal of DNA contacts, or can have a structural effect by destabilizing the local conformation or inducing global denaturation. One of the other two mutations in the component, Y220C, is located in the β-sandwich of the protein and is the most common cancer mutation outside the DNA-binding surface, accounting for 1.4% of somatic missense mutations in p53 [8]. The remaining mutation, R213X, is of the nonsense type, which means that it introduces a stop codon which leads to premature translational stop.

PROPERTY	ALL MUTATIONS	MUTATIONS IN NETWORK	P-VALUE
CPG ASSOCIATION	213/2000	13/68	0.0239
AVERAGE MUTATION RATE	0.1381	0.2935	$4 \cdot 10^{-4}$
DELETERIOUS(SIFT)	200/321	13/15	0.0368
DELETERIOUS(AVCVD)	196/321	13/15	0.0296
GAIN OF FUNCTION	136/514	11/22	0.0136

TABLE 8

Properties of the mutations in our TP53 network.

We also evaluated several properties of the variables in our network using additional data from the same database. Results are shown in Table 8. Some details follow.

CpG islands are associated with methylation, which usually leads to failure in the silencing of certain oncogenes and their corresponding over expression. This is a feature found in many cancer cells [9]. Based on a hypergeometric null, CpG enrichment is borderline significant in our network (p=0.024). In addition, mutation rates are available for 1,348 out of the 2,000 variables in our model, and for 57 out of the 68 mutations in our learned network. The average mutation rate over the whole set of 1,348 mutations is 0.138, whereas it is 0.293 over the 57 mutations in our network ($p = 4 \cdot 10^{-4}$). The functional impact of mutations is classified as "deleterious", "neutral" or "unclassified" using two different approaches, namely the Sorting Intolerant From Tolerant (SIFT) program and the Average Graham Variation and Graham Deviation (AVGVD) indicator. Another permutation test shows that the mutations in our connected components are significantly more likely to be deleterious using either method ($p = 0.0368$ and $p = 0.0296$ respectively). Also known as "driver" mutations, these are known to have a negative impact on the phenotype relative to "neutral" or "passenger" mutations. Finally, in terms of protein descriptors, 514 out of the 2,000 mutations have functional annotations and 136 of those have at least one associated gain of function (GoF) term. Within our network, 22 out of the 68 mutations have functional annotations and 11 of them have at least one associated GoF term. The corresponding hypergeometric p-value is, once again, borderline significant ($p = 0.0136$).

APPENDIX A

PROPOSITION S.1

If $\Psi = \{\psi_1, \ldots, \psi_N\} \in \mathcal{T}_0^*$, then, letting $\psi_k = (\pi_k, A_k, O_k)$ and $J_k = J(\pi_k)$:

(i) For a given k, either $A_k \in R_\Psi$, or the collection of nonempty sets in $A_k \cap O_l, l \neq k$, forms a partition of A_k. The set R_Ψ cannot be empty and is a singleton if $\Psi \in \mathcal{T}_0^*$.

(ii) We have $D = S \cup (\bigcup_{k \in R_\Psi} A_k) \cup (\bigcup_{k=1}^N (J_k \setminus A_k))$ and Ψ specifies a unique probability $P \in \mathcal{F}^*$ given by

$$P(x) = \prod_{j \in S} P_j(x_j) \prod_{k \in R_\Psi} \pi_k(x_{A_k}) \prod_{k=1}^N \pi_k(x_{J_k} | x_{A_k}). \tag{1}$$

where variables indexed over $S := D \setminus \bigcup_{k=1}^N J_k$ are mutually independent under P and correspond to singleton distributions in Eq. (2) from the main paper.

(iii) Define the directed graph $G(\Psi)$ on $\{1, \ldots, N\}$ by drawing an edge from k to l if and only if $A_l \subset O_k$. Then $G(\Psi)$ is acyclic.

Proof. We prove (i) by induction on the number of merges. First, however, note that, by construction, the α-sets of any $\phi \in \mathcal{T}_\Psi$ must be one of the α-sets of the original family, Ψ, since no new α-set is created by the merge operation. If Ψ is an atomic decomposition, this α-set cannot overlap with any of the ω-sets of Ψ and the associated primitive therefore provides a root. Conversely, the α-set of the connector is the only α-set in the merge operation that does not intersect an ω-set, so that any root of an atomic decomposition must coincide with it. This proves that any decomposition in $\mathcal{T}_0^*$ has exactly one root.

Let $\mathcal{U}^{(0)} = \Psi$ and, for $k \geq 1$, $\mathcal{U}^{(k)}$ be the union of $\mathcal{U}^{(k-1)}$ and of the set of results of single merge operations involving elements of $\mathcal{U}^{(k-1)}$. Assume that (i) is true for any subset of Ψ providing an atomic decomposition of elements of $\mathcal{U}^{(k)}$, and let us show that it is true also for those associated to elements of $\mathcal{U}^{(k+1)}$. Consider $\phi \in \mathcal{U}^{(k+1)} \setminus \mathcal{U}^{(k)}$, which therefore can be written as $\phi = \Gamma(\phi_0, \ldots, \phi_r)$ with each $\phi_j = (\pi_j, A_j, O_j) \in \mathcal{U}^{(k)}$. Then, an atomic decomposition of ϕ is obtained by taking the union of decompositions of ϕ_j's by elements of Ψ. Let $\Psi_0, \ldots, \Psi_r$ denote these decompositions and $\tilde{\Psi}$ their union. Since the ϕ_j such that $j \geq 1$ must have disjoint supports, with $J(\pi_j) \cap J(\pi_0) = A_j \subset O_0$, no non-root α-set in Ψ_j ($j \geq 1$) can intersect an ω-set from another decomposition and therefore (i)

remains true for these α-sets. The only change happens with A_j such that $j \geq 1$, which were roots in Ψ_j, and now are included in O_0. Since this ω-set is recursively defined as disjoint unions of ω-sets of merges obtained from elements of $\tilde{\Psi}$, it is a disjoint union of some ω-sets of elements of Ψ, and each A_j with $j \geq 1$ is partitioned by its intersection with these ω-sets.

This proves (i) for elements of $\mathcal{T}_0^*$, and the corresponding statement for $\mathcal{F}_0^*$ is straightforward. Statements (ii) and (iii), can be proved with similar induction arguments. For (ii), this proceeds directly from the definition of Γ, and for (iii), the above discussion shows that $G(\tilde{\Psi})$ is deduced from $G(\Psi)$ by adding edges between the indices of the roots of each $G(\Psi_j)$, $j \geq 1$ and some of the indices of the output sets in Ψ_0. Since these edges only allow to leave $G(\Psi_0)$ (but not to go back) and do not create any communication between $G(\Psi_j)$ and $G(\Psi_{j'})$ for $j \neq j'$ and $j, j' \geq 1$, the resulting graph is acyclic if this was the case for the original subcomponents, which is the induction assumption. $\square$

APPENDIX B
PROOF OF PROPOSITION 2

First, since single-variable distributions are explicitly maximized for $P_j(\lambda) = P_j^*(\lambda)$, and due to Eq. (1) from Proposition S.1 in Appendix A, the problem with fixed Ψ reduces to maximizing the following expression:

$$\ell(\sigma_1, \tau_1, \ldots, \sigma_N, \tau_N) = \sum_{k \in R_\Psi} E_{P^*} \log \pi_k(X_{A_k}; \sigma_k) + \sum_{k=1}^{N} E_{P^*} \log \pi_k(X_{J_k} | X_{A_k}; \tau_k) - \sum_{j \in S} H(P_j^*)$$

where $H(P) = -E_P(\log P)$ is the entropy of P and $S := D \setminus \bigcup_{k=1}^{N} J(\pi_k)$. Assuming, for notational ease, that the maxima over individual parameters are achieved, the maximum of ℓ is given by

$$\begin{aligned}
\hat{\ell} &= \sum_{k \in R_\Psi} \max_{\sigma_k} E_{P^*} \log \pi_k(X_{A_k}; \sigma_k) + \sum_{k=1}^{N} \max_{\tau_k} E_{P^*} \log \pi_k(X_{J_k} | X_{A_k}; \tau_k) - \sum_{j \in S} H(P_j^*) \\
&= \sum_{k \in R_\Psi} E_{P^*} \log \frac{\pi_k(X_{A_k}; \hat{\sigma}_k)}{\prod_{j \in A_k} P_j^*(X_j)} + \sum_{k=1}^{N} E_{P^*} \log \frac{\pi_k(X_{J_k}; \hat{\sigma}_k, \hat{\tau}_k)}{\pi_k(X_{A_k}; \hat{\sigma}_k) \prod_{j \in J(\pi_k) \setminus A_k} P_j^*(X_j)} \\
&+ \sum_{k \in R_\Psi} E_{P^*} \log \prod_{j \in A_k} P_j^*(X_j) + \sum_{k=1}^{N} E_{P^*} \log \prod_{j \in J(\pi_k) \setminus A_k} P_j^*(X_j) - \sum_{j \in S} H(P_j^*) \\
&= \sum_{k \in R_\Psi} \mu(\psi_k) + \sum_{k=1}^{N} \rho(\psi_k) - \sum_{j \in D} H(P_j^*)
\end{aligned}$$

□

APPENDIX C
PROOF OF PROPOSITION 3

The "only if" part being obvious, we briefly discuss the proof of the "if" part, which can go by induction on the number of elements in Ψ. The result is true if Ψ is empty, or has a single element. Assume that it is true for any Ψ with cardinality N or less and take a family $\Psi = \{\psi_1, \ldots, \psi_{N+1}\}$ such that $G(\Psi)$ is a union of balanced trees. Let $\psi_k = (\pi_k, A_k, O_k)$ (with $J_k = A_k \cup O_k$). Assume that ψ_j's are ordered so that ψ_{N+1} is a root in Ψ and let $\Psi' = \{\psi_1, \ldots, \psi_N\}$. Since the α-sets are singletons, the α-node of ψ_{N+1} must also be a root of the connected component, say T, in $G(\Psi)$ that contains it.

Since the definition of almost-balanced trees is recursive, removing the top component either separates T into two balanced subtrees (when ψ_{N+1} is a triplet with two nonempty subtrees appended to its ω-nodes), or leaves a – possibly empty – single balanced tree (in the other cases). The fact that α-nodes cannot be shared among ψ_k's removes the problematic case of two pairs sharing an α-node at the root of a tree.

In all cases, $G(\Psi')$ is a union of almost-balanced trees, and, since (i) and (ii) are obviously inherited by the restriction, $\Psi' \in \mathcal{F}_0^*$. Since putting ψ_{N+1} back to its former position is a legal merge operation, we find that $\Psi \in \mathcal{F}_0^*$ also. □

REFERENCES

[1] D. Heckerman, "A tutorial on learning Bayesian networks," Microsoft Research, Tech. Rep. MSR-TR-95-06, March 1995.

[2] A. J. Butte and I. S. Kohane, "Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements," *Pac. Symp. Biocomput.*, pp. 418–29, 2000.

[3] A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Favera, and A. Califano, "ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, vol. 7, p. S7, 2006.

[4] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner, "Large-scale mapping and validation of E. coli transcriptional regulation from a compendium of expression profiles," *PLoS Biol.*, vol. 5, no. 1, p. e8, Jan 2007.

[5] G. F. Cooper and T. Dietterich, "A Bayesian method for the induction of probabilistic networks from data," in *Machine Learning*, 1992, pp. 309–347.

[6] A. N. Bullock, H. J., and A. R. Fersht, "Quantitative analysis of residual folding and DNA binding in mutant p53 core domain: Definition of mutant states for rescue in cancer therapy," *Oncogene*, no. 19, pp. 1245–1256, 2000.

[7] T. E. Baroni, T. Wang, H. Qian, L. R. Dearth, L. N. Truong, J. Zeng, A. E. Denes, S. W. Chen, and R. K. Brachmann, "A global suppressor motif for p53 cancer mutants," *Proc. Natl. Acad. Sci. USA*, vol. 101, no. 14, pp. 4930–4935, 2004.

[8] A. C. Joerger and A. R. Fersht, "Structure-function-rescue: The diverse nature of common p53 cancer mutants," *Oncogene*, vol. 26, no. 15, pp. 2226–2242, April 2007.

[9] P. A. Jones and D. Takai, "The role of DNA methylation in mammalian epigenetics," *Science*, vol. 293, no. 5532, pp. 1068–1070, 2001.