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

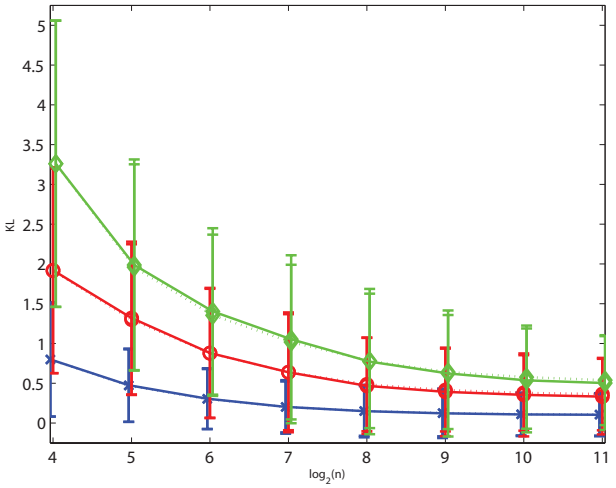

Fig. S.1. Evolution of KL divergence as a function of the sample size (in $\log_2$ scale) for fixed $\epsilon = 1$ and different choices of the number of variables: $d = 5$ (blue, cross); $d = 10$ (red, circle); $d = 15$ (green, diamond). Solid lines correspond to the ILP solution and dotted lines correspond 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

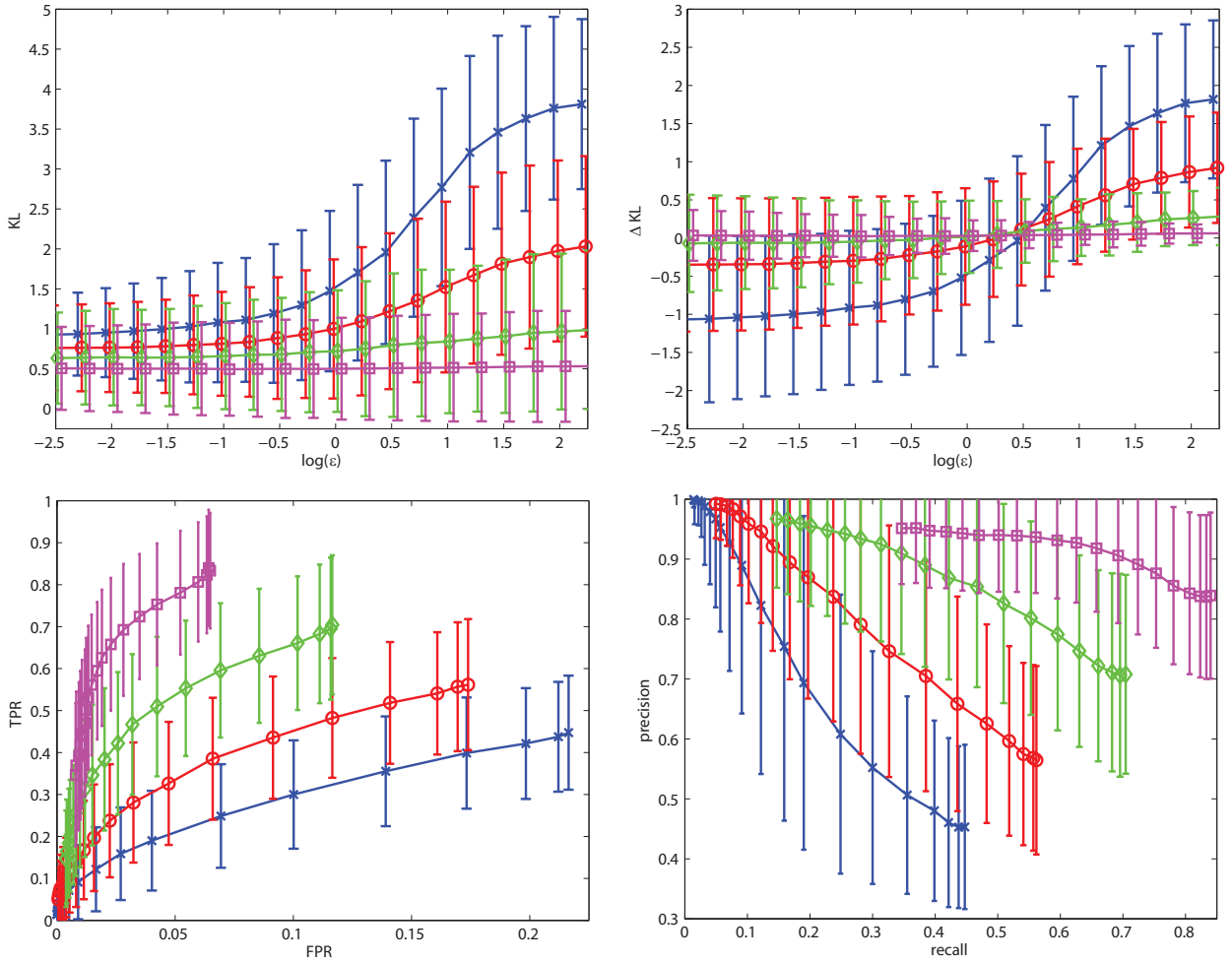

Fig. S.2. Top left: evolution of the KL divergence for $d = 10$ as a function of $\log(\epsilon)$ for different sample sizes ($n = 25$ in blue and cross marker, $n = 50$ in red and circular marker, $n = 100$ in green and diamond marker, $n = 200$ in magenta and square marker). Top right: difference between 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).

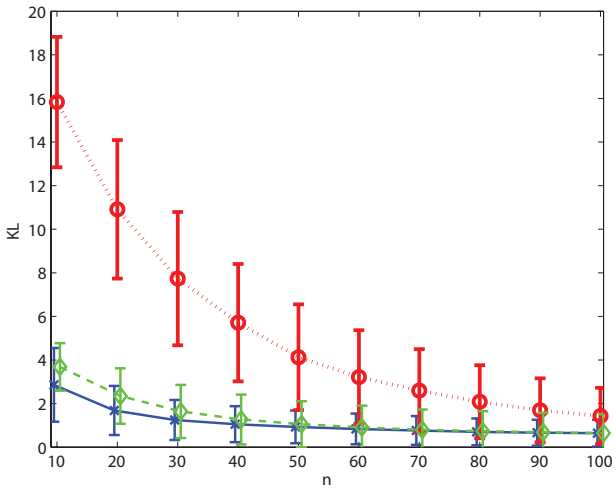

Fig. S.3. Evolution of KL divergence for small sample sizes (linear scale) between the true distribution Q and the distributions learned using CAM (solid, blue, cross), using K2 (dotted, red, circle) and using the true generating graph and simply estimating parameters (dashed, green, diamond). Each curve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

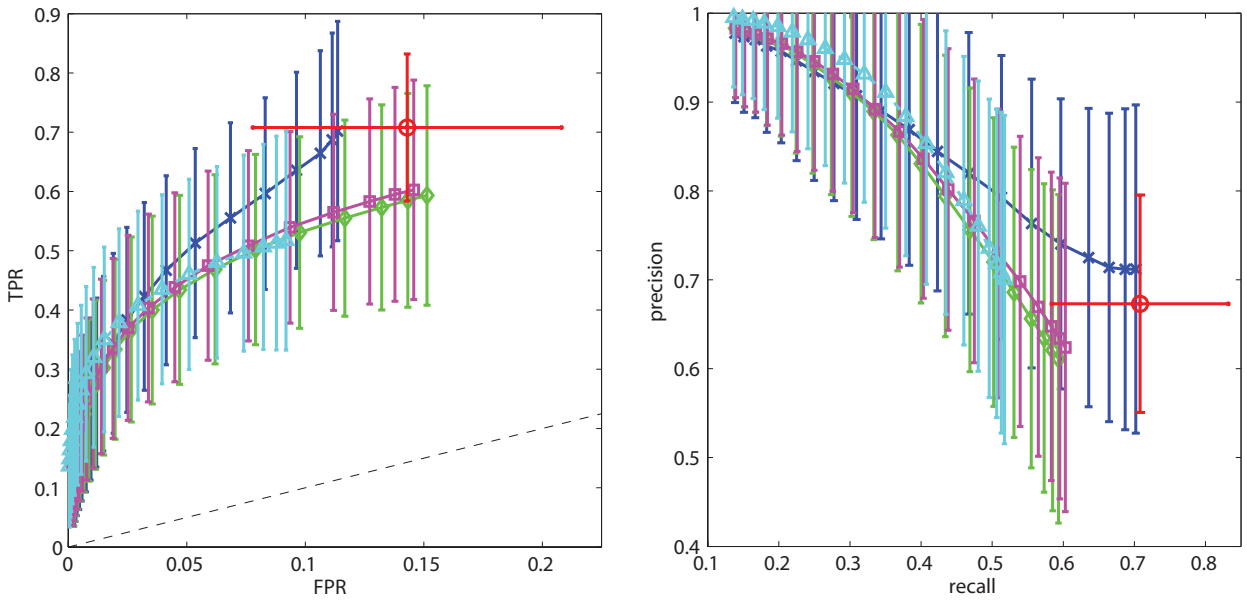

Fig. S.4. Comparison of (undirected) edge recovery accuracy. Curves correspond to CAM (blue,cross), relevance networks (green, diamond), CLR (magenta, square) and ARACNE (cyan, triangle). Results for K2 are shown as a single encircled red dot. The black dashed line on the left panel corresponds to random 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

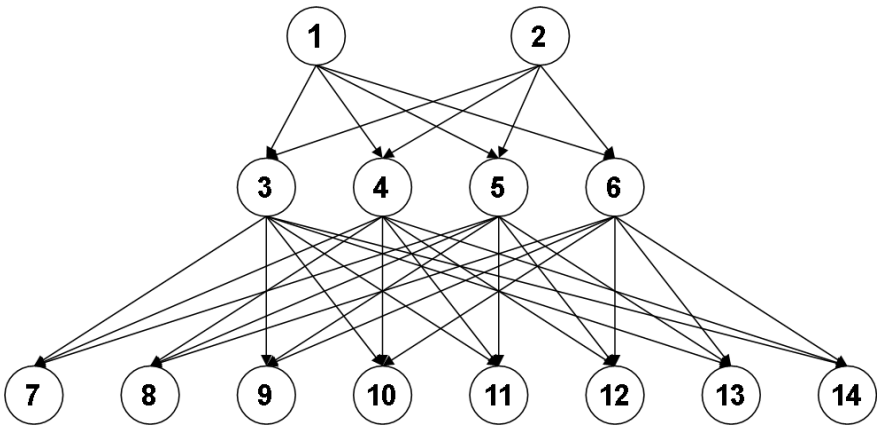

Fig. S.5. Graph structure for Bayesian network experiment.

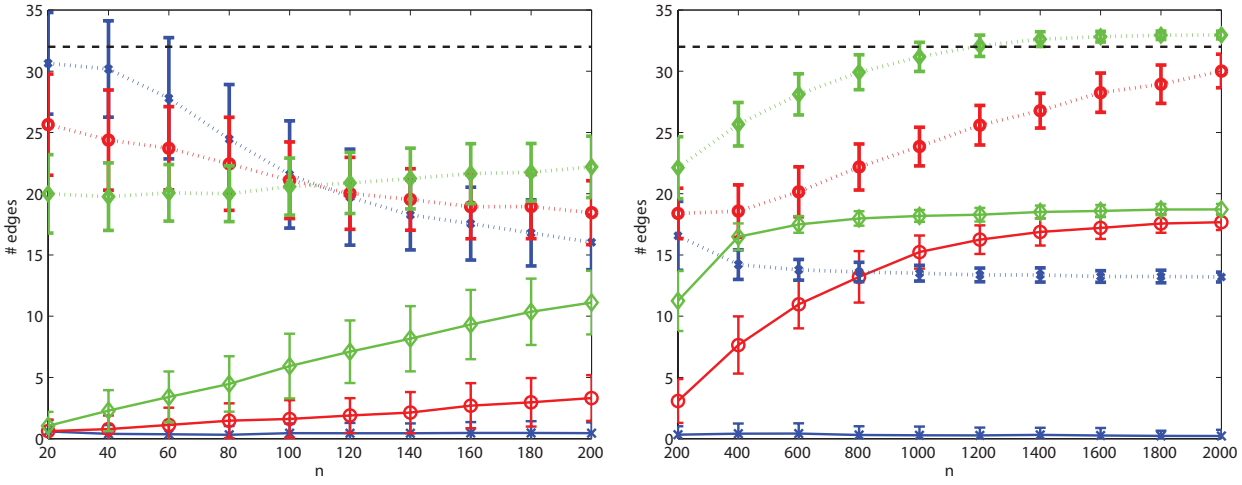

Fig. S.6. Average number of edges learned as a function of sample size. The black dashed line is the number of edges (32) in the true Bayesian network. Results are shown for different degrees of model dependence: $\alpha = 0$ (blue, cross), $\alpha = 0.5$ (red, circle) and $\alpha = 1$ (green, diamond).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

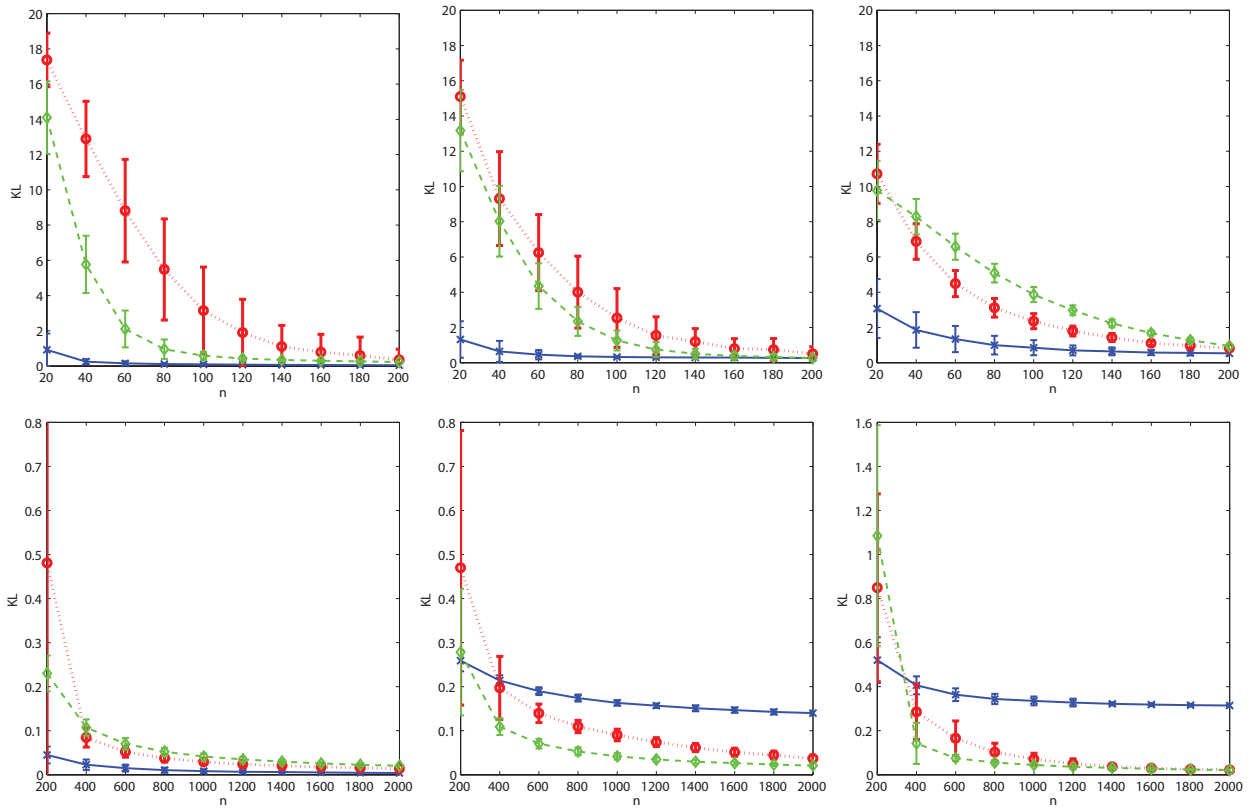

Fig. S.7. KL divergence between the Bayesian network ground truth distribution and the distributions learned using CAM (solid, blue, cross), K2 (dotted, red, circle) and the true generating graph with MLE parameters (dashed, green, diamond). Results are presented for $\alpha = 0$ (left column), $\alpha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

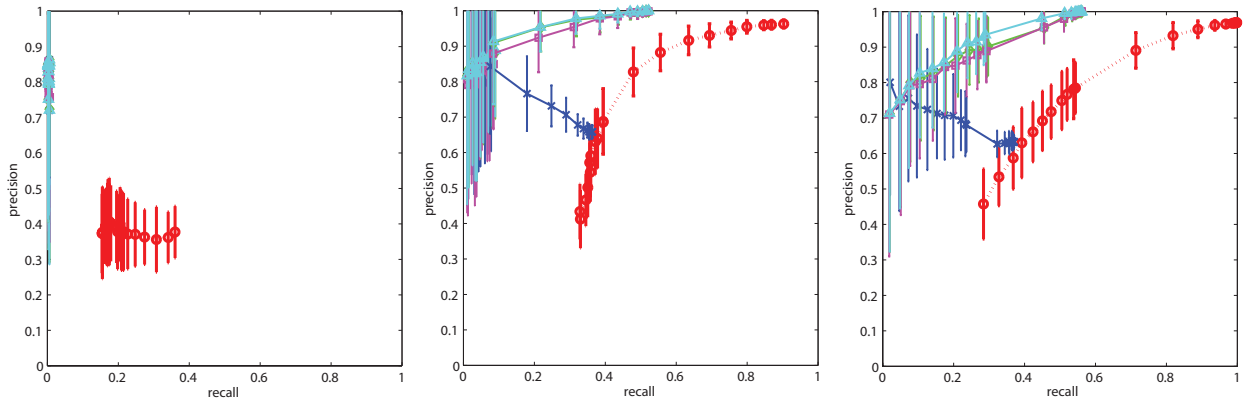

Fig. S.8. Precision-recall curves for the generic Bayesian network simulation. Curves correspond to CAM (blue line, cross marker), K2 (red dotted line, circle marker), RN (green line, diamond marker), CLR (magenta line, square marker) and ARACNE (cyan line, triangle marker). Results are presented for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

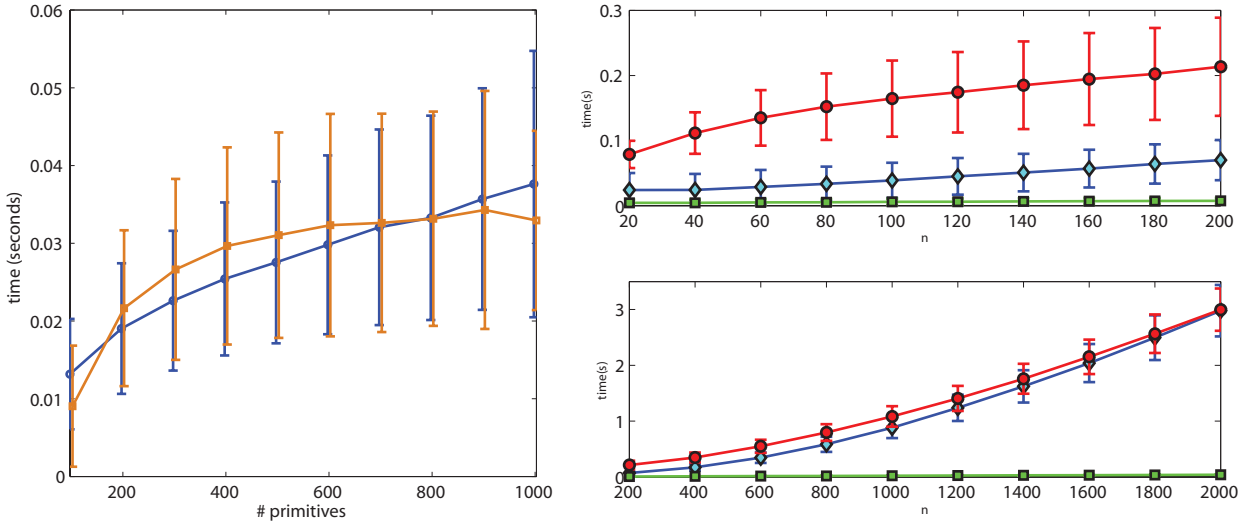

Fig. S.9. Average runtime results for synthetic data simulations. Left panel shows average runtimes as a function of $|\mathcal{T}_0|$ for random choices of structure and parameterization (learned as in the experiment from Fig. S.2, for a fixed choice of $d = 10$, $n = 10$, and averaged over $20e3$ replicates). 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.

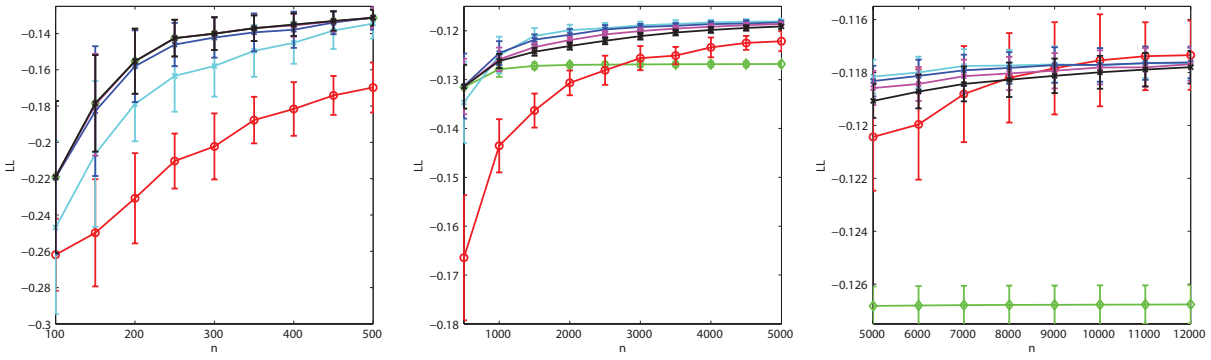

Fig. S.10. Average log-likelihood over holdout samples as a function of observed sample size. Results are shown for BL (green, diamond), K2 (red, circle) and CAM (cross marker). CAM results are shown for $\epsilon = 1$ (cyan), $\epsilon = 10^{-3}$ (blue), $\epsilon = 10^{-6}$ (magenta) and $\epsilon = 10^{-9}$ (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

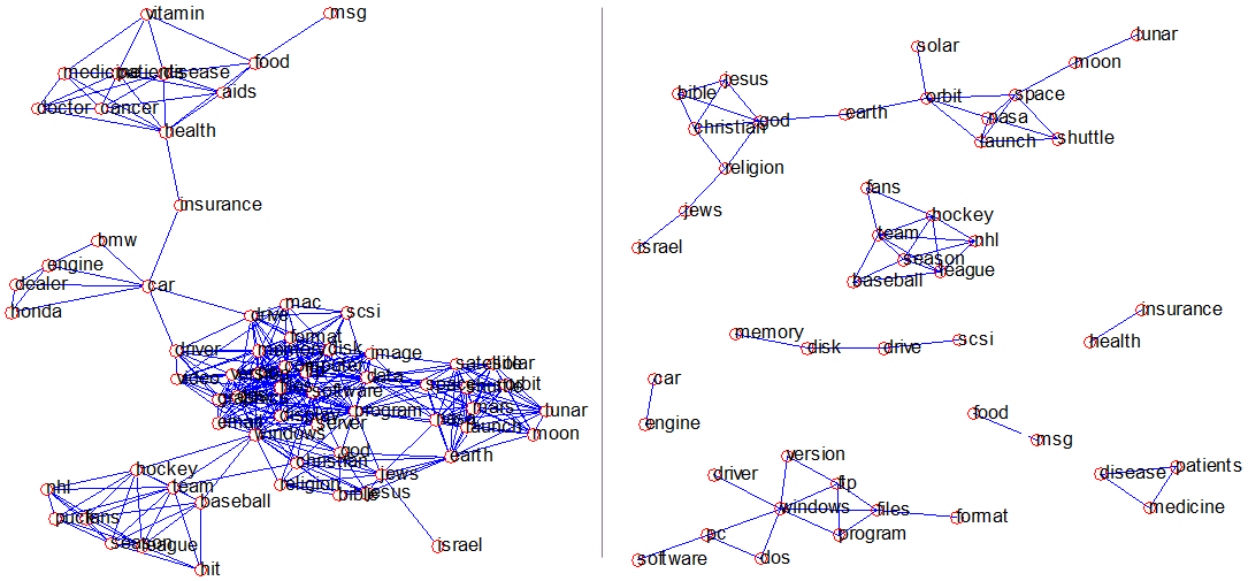

Fig. S.11. Two graphs learned by relevance networks using the *20newsgroups* dataset. Left panel: the mutual information threshold is the same as the threshold that was used for selecting binary primitives in the network shown in Figure 5 of the main paper. Right panel: the threshold is chosen so that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

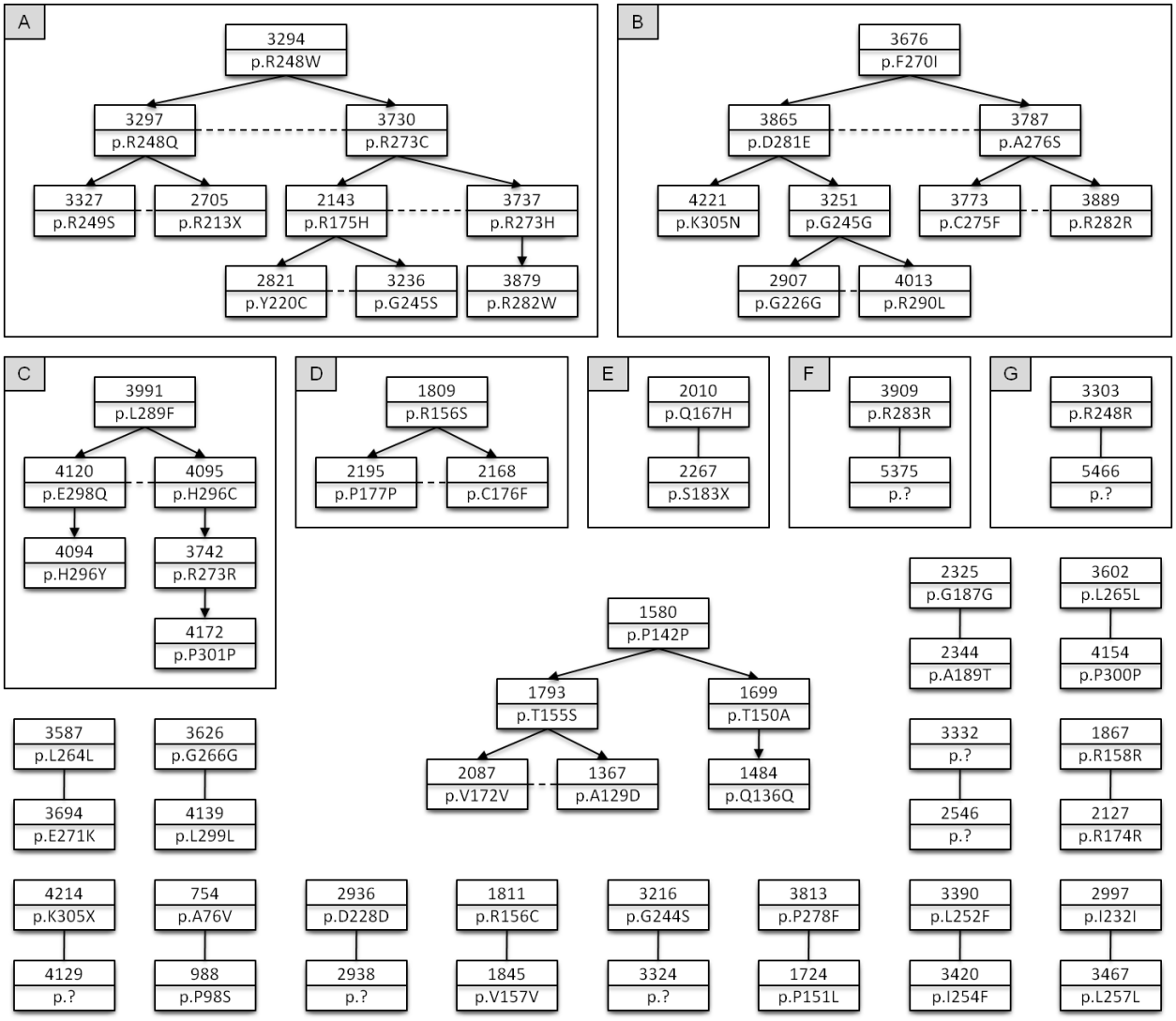

Fig. S.12. Network of interactions among somatic mutations for the IARC TP53 dataset learned using CAM models. Dashed lines are used to link siblings that belong to the same triplet primitive. For each node, we provide the unique mutation identifier in the IARC Database and the (standard) mutation description 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.