Convex Relaxations for Learning Bounded-Treewidth Decomposable Graphs

K. S. Sesh Kumar SESH-KUMAR.KARRI@INRIA.FR
Francis Bach FRANCIS.BACH@INRIA.FR
SIERRA project-team, INRIA - Département d'Informatique de l'Ecole Normale Supérieure, Paris, France.

Abstract

We consider the problem of learning the structure of undirected graphical models with bounded treewidth, within the maximum likelihood framework. This is an NP-hard problem and most approaches consider local search techniques. In this paper, we pose it as a combinatorial optimization problem, which is then relaxed to a convex optimization problem that involves searching over the forest and hyperforest polytopes with special structures. A supergradient method is used to solve the dual problem, with a run-time complexity of $O(k^3 n^{k+2} \log n)$ for each iteration, where n is the number of variables and k is a bound on the treewidth. We compare our approach to state-of-the-art methods on synthetic datasets and classical benchmarks, showing the gains of the novel convex approach.

1. Introduction

Graphical models provide a versatile set of tools for probabilistic modeling of large collections of interdependent variables. They are defined by graphs that encode the conditional independences among the random variables, together with potential functions or conditional probability distributions that encode the specific local interactions leading to globally well-defined probability distributions (see, e.g., Bishop et al., 2006; Wainwright & Jordan, 2008; Koller & Friedman, 2009).

In many domains such as computer vision, natural language processing or bioinformatics, the structure of the graph follows naturally from the constraints of the problem at hand. In other situations, it might be desirable to estimate this structure from a set of observations. It allows (a) a statistical fit of rich probability distributions that can be considered for further use, and (b) discovery of structural relationships between different variables. In the former case, distributions with tractable inference are often desirable, i.e., inference with run-time complexity that does not scale exponentially in the number of variables in the model. The simplest constraint to ensure tractability is to impose tree-structured graphs (Chow & Liu, 1968). However, these distributions are not rich enough, and following earlier work (Malvestuto, 1991; Bach & Jordan, 2002; Narasimhan & Bilmes, 2004; Chechetka & Guestrin, 2007; Gogate et al., 2010; Szántai & Kovács, 2011), we consider models with bounded *treewidth*, not simply by one (i.e., trees), but by a small constant k.

Beyond the possibility of fitting tractable distributions (for which probabilistic inference has linear complexity in the number of variables), learning bounded-treewidth graphical models is key to designing approximate inference algorithms for graphs with higher treewidth. Indeed, as shown by Saul & Jordan (1995); Wainwright & Jordan (2008); Kolmogorov & Schoenemann (2012), approximating general distributions by tractable distributions is a common tool in variational inference. However, in practice, the complexity of variational distributions is often limited to trees (i.e., $k = 1$), since these are the only ones with exact polynomial-time structure learning algorithms. The convex relaxation designed in this paper enables us to augment the applicability of variational inference, by allowing a finer trade-off between run-time complexity and approximation quality.

Apart from trees, learning the structure of a directed or undirected graphical model, with or without constraints on the treewidth, remains a hard problem. Two types of algorithms have emerged,

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

based on the two equivalent definitions of graphical models: (a) by testing conditional independence relationships (see, e.g., Spirtes et al., 2001) or (b) by maximizing the log-likelihood of the data using the factorized form of the distribution (see, e.g., Friedman & Koller, 2003). In the specific context of learning bounded-treewidth graphical models, the latter approach has been shown to be NP-hard (Srebro, 2002) and led to various approximate algorithms based on local search techniques (Malvestuto, 1991; Deshpande et al., 2001; Karger & Srebro, 2001; Bach & Jordan, 2002; Shahaf et al., 2009; Gogate et al., 2010; Szántai & Kovács, 2011), while the former approach led to algorithms based on independence tests (Narasimhan & Bilmes, 2004; Chechetka & Guestrin, 2007), which have recovery guarantees when the data-generating distribution has low treewidth.

In this paper, we make the following contributions:

- We provide a novel convex relaxation for learning bounded-treewidth decomposable graphical models from data in polynomial time. This is achieved by posing the problem as a combinatorial optimization problem in Section 2, which is relaxed to a convex optimization problem that involves the graphic and hypergraphic matroids, as shown in Section 3.
- We show in Section 4 how a supergradient ascent method may be used to solve the dual optimization problem, using greedy algorithms as inner loops on the two matroids. Each iteration has a run-time complexity of $O(k^3 n^{k+2} \log n)$, where n is the number of variables. We also show how to round the obtained fractional solution.
- We compare our approach to state-of-the-art methods on both synthetic datasets and classical benchmarks in Section 5, showing the gains of the novel convex approach.

2. Maximum Likelihood Decomposable Graphical Models

In this section, we first review the relevant concepts of decomposable graphs and junction trees; for more details, see Bishop et al. (2006); Wainwright & Jordan (2008); Koller & Friedman (2009). We then cast the problem of learning the maximum likelihood bounded treewidth graph as a combinatorial optimization problem.

2.1. Decomposable graphs and junction trees

We assume we are given an *undirected* graph G defined on the set of vertices $V = \{1, 2, \ldots, n\}$. Let $\mathcal{C}(G)$ denote the set of maximal cliques of G (which we will refer to as cliques). We consider n random variables $X_1, \ldots, X_n$ (referred to as X), associated with each vertex indexed by V. For simplicity, they are assumed to be discrete, but this is not a restriction as maximum likelihood will use only entropies that can be extended to differentiable entropies (Cover & Thomas, 2006).

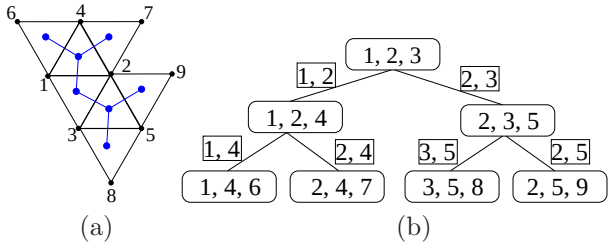


Figure 1. (a) A decomposable graph on the set of vertices $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ having treewidth 2 with an embedded junction tree representing the maximal cliques by blue dots. (b) The corresponding junction tree representation of the decomposable graph with ovals representing the maximal cliques and the rectangles representing the corresponding separator set.

The distribution $p(x)$ of X is said to factorize in the graph G, if and only if it factorizes as a product of potentials that depend only on the variables within maximal cliques.

A graph is said to be *decomposable* if it has a junction tree, i.e., a spanning tree whose vertices are maximal cliques of G (i.e., $\mathcal{C}(G)$ is the vertex set) such that:

- the junction tree connects only cliques that have a common element (*clique tree* property),
- for any vertex $i \in V$, the subgraph of cliques containing i is a tree (*running intersection* property).

Let $\mathcal{T}(G)$ denote the edges of the junction tree over the set of cliques $\mathcal{C}(G)$. When the graph G is decomposable, the distribution $p(x)$ of X factorizes in G if and only if it may be written as $p(x) = \prod_{C \in \mathcal{C}(G)} p_C(x_C) / \prod_{(C,D) \in \mathcal{T}(G)} p_{C \cap D}(x_{C \cap D})$, where $p_C(x_C)$ denotes the marginal distribution of random variables belonging to $C \in \mathcal{C}(G)$ and $p_{C \cap D}(x_{C \cap D})$ denotes the marginal distribution of random variables belonging to the *separator* set $C \cap D$, with $(C, D) \in \mathcal{T}(G)$. See Figure 1. The *treewidth* of G is the maximal size of the cliques in G, minus one.

An alternative representation of decomposable graphs may be obtained by considering *hypergraphs*. Hypergraphs are defined by a base set V and a set of hyperedges, i.e., subsets of V. A hypergraph is said to be acyclic if and only if the resulting graph obtained by connecting all elements within an hyperedge is decom-

posable. Unfortunately, the nice properties of acyclic graphs do not transfer to acyclic hypergraphs. Particularly, the matroid property, which allows exact greedy algorithms, does not hold. In Section 3.2, we will use a lesser known more general notion of acyclicity that will lead to exact greedy algorithms.

2.2. Maximum likelihood estimation

Given N observations $x^1, \dots, x^N$ of X, we denote the corresponding empirical distribution of X by $\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N} \delta(x = x^i)$. Given the structure of a decomposable graph G, the maximum likelihood distribution that factorizes in G may be obtained by combining the marginal empirical distributions on all maximal cliques and their separators. The goal of structure learning is to maximize the log-likelihood with respect to the graph as well. As shown by Malvestuto (1991) and Srebro (2002), this corresponds to minimizing the following combination of entropies with respect to the graph (see a proof in the supplementary material):

$$\sum_{C \in \mathcal{C}(G)} H(C) - \sum_{(C,D) \in \mathcal{T}(G)} H(C \cap D)\,, \qquad (1)$$

where $H(S)$ is the empirical entropy of the random variables indexed by the set $S \subseteq V$, defined by $H(S) = \sum_{x_S}\{-\hat{p}_S(x_S)\log \hat{p}_S(x_S)\}$, and where the sum is taken over all possible values of x_S.

Note that in this paper, we will not be using a traditional model selection term (see, e.g., Friedman & Koller, 2003) as we will only consider models of low tree-width (thus with a bounded number of parameters).

2.3. Combinatorial optimization problem

We now consider the problem of learning a decomposable graph of treewidth less than k. We assume that we are given all entropies $H(S)$ for subsets S of V of cardinality less than $k+1$.

Since we do not add any model selection term, without loss of generality (Szántai & Kovács, 2012), we restrict the search space to the space of *maximal junction trees*, i.e., junction trees with $n-k$ maximal cliques of size $k+1$ and $n-k-1$ separator sets of size k between two cliques of size $k+1$. Our natural search spaces are thus characterized by $\mathcal{D}$, the set of all subsets of size $k+1$ of V, of cardinality $\binom{n}{k+1}$, and $\mathcal{E}$, the set of all potential edges in a junction tree, i.e., $\mathcal{E} = \{(C,D) \in \mathcal{D}\times\mathcal{D}, C\cap D \neq \varnothing, |C\cap D| = k\}$. The cardinality of $\mathcal{E}$ is $\binom{n}{k+2}.\binom{k+2}{2}$ (number of subsets of size $k+2$ times the number of possibility of excluding two elements to obtain a separator).

A decomposable graph will be represented by a clique selection function $\tau : \mathcal{D} \to \{0,1\}$ and an edge selection function $\rho : \mathcal{E} \to \{0,1\}$ so that $\tau(C) = 1$ if C is a maximal clique of the graph and $\rho(C,D) = 1$ if (C,D) is an edge in the junction tree. Both ρ and τ will be referred to as incidence *functions* or incidence *vectors*, when seen as elements of $\{0,1\}^{\mathcal{D}}$ and $\{0,1\}^{\mathcal{E}}$.

Thus, minimizing the problem defined in Eq. (1) is equivalent to minimizing,

$$\mathcal{P}(\tau,\rho) = \sum_{C\in\mathcal{D}} H(C)\tau(C) - \sum_{(C,D)\in\mathcal{E}} H(C\cap D)\rho(C,D), \quad (2)$$

with the constraint that (τ,ρ) forms a decomposable graph.

At this time, we have merely reparameterized the problem with the clique and edge selection functions. We now consider a set of necessary and sufficient conditions for the pair to form a decomposable graph. Some are convex in (τ,ρ), while some are not. The latter ones will be relaxed in Section 3. From now on, we denote by $1_{i\in C}$ the indicator function for $i \in C$ (i.e., it is equal to 1 if $i \in C$ and zero otherwise).

- *Covering* V: all vertices, must be covered,

$$\forall i \in V,\ \sum_{C\in\mathcal{D}} 1_{i\in C}\tau(C) \geq 1. \qquad (3)$$

- *Number of edges*: exactly $n-k-1$ edges are selected,

$$\sum_{(C,D)\in\mathcal{E}} \rho(C,D) = n-k-1. \qquad (4)$$

- *Number of cliques*: exactly $n-k$ cliques are selected,

$$\sum_{C\in\mathcal{D}} \tau(C) = n-k. \qquad (5)$$

- *Running intersection property*: the subtree of cliques containing any vertex $i \in V$ must be spanning, i.e., the number of edges has to be equal to the number of nodes minus one, i.e., $\forall i \in V$,

$$\sum_{(C,D)\in\mathcal{E}} 1_{i\in(C\cap D)}\rho(C,D) - \sum_{C\in\mathcal{D}} 1_{i\in C}\tau(C) + 1 = 0. \quad (6)$$

- *Edges between selected cliques*: we must have the following link between τ and ρ,

$$\forall C \in \mathcal{D},\ \tau(C) = \max_{D\in\mathcal{D},\ (C,D)\in\mathcal{E}} \rho(C,D). \qquad (7)$$

- *Acyclicity of* ρ: ρ has to be the incidence vector of a forest, i.e.,

$$\rho \text{ represents a subforest of the graph } (\mathcal{D},\mathcal{E}). \quad (8)$$

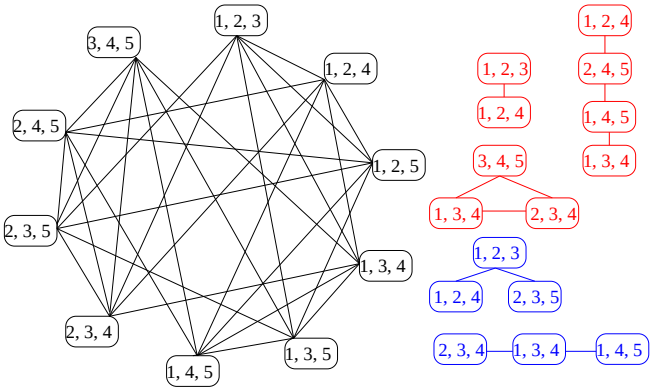


Figure 2. Space of cliques $\mathcal{D}$ denoted by ovals and the space of feasible edges $\mathcal{E}$ denoted by lines for $V = \{1, 2, 3, 4, 5\}$ and treewidth 2 (in black). Clique and edge selections in blue represent decomposable graphs while those in red denote graphs that are not decomposable (best seen in color).

– *Acyclicity of* τ: τ has to be the incidence vector of an acyclic hypergraph, i.e.,

$$\tau \text{ represents an acyclic hypergraph of } (V, \mathcal{D}). \quad (9)$$

The above constraints encode the classical definition of junction trees. Thus our combinatorial problem is exactly equivalent to minimizing $P(\tau, \rho)$ defined in Eq. (2), subject to the constraints in in Eq. (3), Eq. (4), Eq. (5), Eq. (6), Eq. (7), Eq. (8) and Eq. (9). Note that the constraint in Eq. (9) that τ represents an acyclic hypergraph is implied by the other constraints.

Figure 2 shows clique and edge selections in blue which satisfy all these constraints and hence represent a decomposable graph. The clique and edge selections in red violate at least one of these constraints.

3. Convex Relaxation

We now provide a convex relaxation of the combinatorial problem defined in Section 2.3. The covering constraint in Eq. (3), the number of edges and the number of cliques constraints in Eq. (4) and Eq. (5) respectively, and the running intersection property in Eq. (6) are already convex in (τ, ρ).

The non-convex constraint in Eq. (7) that $\forall C \in \mathcal{D},\ \tau(C) = \max_{D \in \mathcal{D},\ (C,D) \in \mathcal{E}} \rho(C, D)$ may be relaxed into:

– *Edge constraint*: selection of edges only if the both the incident cliques are selected, i.e.,

$$\forall C \in \mathcal{D},\ \forall (C, D) \in \mathcal{E},\ \rho(C, D) \leq \tau(C). \quad (10)$$

– *Clique constraint*: selection of a clique if at least an edge incident on it is selected, i.e.,

$$\forall C \in \mathcal{D},\ \tau(C) \leq \sum_{(C,D) \in \mathcal{E}} \rho(C, D). \quad (11)$$

We now consider the two acyclicity constraints in Eq. (8) and Eq. (9).

3.1. Forest polytope

Given the graph $(\mathcal{D}, \mathcal{E})$, the *forest polytope* is the convex hull of all incidence vectors ρ of subforests of $(\mathcal{D}, \mathcal{E})$. Thus, it is exactly the convex hull of all $\rho : \mathcal{E} \to \{0, 1\}$ such that ρ satisfies the constraint in Eq. (8). We may thus relax it into:

– *Tree constraint*:

$$\rho \text{ is in the forest polytope of } (\mathcal{D}, \mathcal{E}). \quad (12)$$

While the new constraint in Eq. (12) forms a convex constraint, it is crucial that it may be dealt with empirically in polynomial time. This is made possible by the fact that one may maximize any linear function over that polytope. Indeed, for a weight function $w : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$, maximizing $\sum_{(C,D) \in \mathcal{E}} w(C, D)\rho(C, D)$ is exactly a maximum weight forest problem, and its solution may be obtained by Kruskal's algorithm, i.e., (a) order all (potentially negative) weights $w(C, D)$ and (b) greedily select edges (C, D), i.e., set $\rho(C, D) = 1$, with higher weights first, as long as they form a forest and as long as the weights are positive. When we add the restriction that the number of edges is fixed (in our case $n-k-1$), then the algorithm is stopped when exactly the desired number of edges is selected (whether the corresponding weights are positive or not). See, e.g., Schrijver (2004).

The polytope defined above may also be defined as the independence polytope of the graphic matroid, which is the traditional reason why the greedy algorithm is exact (see, e.g., Schrijver, 2004). In the next section, we show how this can be extended to hypergraphs.

3.2. Hypergraphic matroid

Given the set of potential cliques $\mathcal{D}$ over V, we consider functions $\tau : \mathcal{D} \to \{0, 1\}$ that are equal to one when a clique is selected, and zero otherwise. Ideally, we would like to treat the acyclicity of the associated hypergraph in a similar way than for a regular graph. However, the set of acyclic subgraphs of the hypergraph defined from $\mathcal{D}$ does not form a matroid, and thus the polytope defined as the convex hull of all incidence vectors/functions of acyclic hypergraphs may be defined, but the greedy algorithm is not applicable. In

order to define what is referred to as the *hypergraphic matroid*, one needs to relax the notion of acyclicity.

We now follow Lorea (1975); Frank et al. (2003); Fukunaga (2010) and define a different notion of acyclicity for hypergraphs. An hypergraph $(V, \mathcal{F})$ is an *hyperforest* if and only if for all $A \subset V$, the number of hyperedges in $\mathcal{F}$ contained in A is less than $|A| - 1$. A non-trivially equivalent definition is that we can select two elements in each hyperedege so that the graph with vertex set V and with edge set composed of these pairs is a forest.

Given an hypergraph with hyperedge set $\mathcal{D}$, the set of sub-hypergraphs which are hyperforests forms a matroid. This implies that given a weight function on $\mathcal{D}$, one may find the maximum weight hyperforest with a greedy algorithm that ranks all hyperedges and select them as long as they do not violate acyclicity (with the notion of acyclicity just defined and for which we exhibit a test below).

Testing acyclicity. Checking acyclicity of an hypergraph $(V, \mathcal{F})$ (which is needed for the greedy algorithm above) may be done by minimizing with respect to $A \subset V$

$$|A| - \sum_{G \in \mathcal{F}} 1_{G \subset A}.$$

The hypergraph is an hyperforest if and only if the minimum is greater or equal to one. The minimization of this function may be cast a min-cut/max-flow problem as follows (Fukunaga, 2010):

- single source, single sink, one node per hyperedge in $\mathcal{F}$, one node per vertex in V,
- the source points towards each hyperedge with unit capacity,
- each hyperedge points towards the vertices it contains, with infinite capacity,
- each vertex points towards the sink, with unit capacity.

Link with decomposability. The hypergraph obtained from the maximal cliques of a decomposable graph can easily be seen to be an hyperforest. But the converse is not true.

Hyperforest polytope. We can now naturally define the hyperforest polytope as the convex hull of all incidence vectors of hyperforests. Thus the constraint in Eq. (9) may be relaxed into:

- *Hyperforest constraint*:

$$\tau \text{ is in the hyperforest polytope of } (V, \mathcal{D}). \quad (13)$$

3.3. Relaxed optimization problem

We can now formulate our combinatorial problem as follows:

$$\begin{array}{l} \min \mathcal{P}(\tau, \rho) \text{ subject to} \\ \tau \in \{0,1\}^{\mathcal{D}},\ \rho \in \{0,1\}^{\mathcal{E}}, \\ \text{Eq. (3), Eq. (4), Eq. (5), Eq. (6),} \\ \text{Eq. (10), Eq. (11), Eq. (12) and Eq. (13).} \end{array} \quad (14)$$

All constraints except the integrality constraints are convex. Let *τ-relaxed primal* be the partially relaxed primal optimization problem formed by relaxing only the integral constraint on τ in Eq. (14), i.e., replacing $\tau \in \{0,1\}^{\mathcal{D}}$ by $\tau \in [0,1]^{\mathcal{D}}$. Note that this is not a convex problem due to the remaining integral constraint on ρ, but it remains equivalent to the original problem as the following proposition shows (see proof in the supplementary material):

Proposition 1 *The combinatorial problem in Eq. (14) and the τ-relaxed primal problem are equivalent.*

The *convex relaxation* for the primal optimization problem formed by relaxing the integral constraint on both τ and ρ can now be defined as

$$\begin{array}{l} \min \mathcal{P}(\tau, \rho) \text{ subject to} \\ \tau \in [0,1]^{\mathcal{D}},\ \rho \in [0,1]^{\mathcal{E}}, \\ \text{Eq. (3), Eq. (4), Eq. (5), Eq. (6),} \\ \text{Eq. (10), Eq. (11), Eq. (12) and Eq. (13).} \end{array} \quad (15)$$

It is not tight anymore in general, i.e., the minimum value of the convex problem may be smaller than the minimal value of the non-convex problem. However, when $k = 1$ (i.e., trees), our relaxation is tight and we recover the results of the Chow-Liu algorithm (see proof in supplementary material):

Proposition 2 *If $k = 1$, the convex relaxation in Eq. (15) is equivalent to Eq. (14).*

4. Solving the dual problem

We now show how the convex problem may be minimized in polynomial time. Among the constraints of our convex problem in Eq. (14), some are simple linear constraints, some are complex constraints depending on the forest and hyperforest polytopes defined in Section 3. We will define a dual optimization problem by introducing the least possible number of Lagrange multipliers (i.e., dual variables) (see, e.g., Bertsekas, 1999) so that the dual function (and a supergradient) may be computed and maximized efficiently. We introduce the following dual variables:

- Set cover constraints in Eq. (3): $\gamma \in \mathbb{R}_+^V$.

$$q_1(\gamma,\mu,\lambda,\eta) = \inf_{\substack{\tau\in[0,1]^{\mathcal{D}} \\ \sum_{C\in\mathcal{D}}\tau(C)=n-k \\ \tau\in \text{ hyperforest polytope of } (V,\mathcal{D})}} \sum_{C\in\mathcal{D}} \Big(H(C) - \sum_{i\in C}(\mu_i+\gamma_i) - \sum_{(C,D)\in\mathcal{E}} \lambda_{CD} + \eta_C \Big)\tau(C). \quad (16)$$

$$q_2(\gamma,\mu,\lambda,\eta) = - \sup_{\substack{\rho\in[0,1]^{\mathcal{E}} \\ \sum_{(C,D)\in\mathcal{E}}\rho(C,D)=n-k-1 \\ \rho\in \text{ forest polytope of } (\mathcal{D},\mathcal{E})}} \sum_{(C,D)\in\mathcal{E}} \Big(H(C\cap D) - \sum_{i\in(C\cap D)} \mu_i - \lambda_{CD} - \lambda_{DC} + \eta_C + \eta_D \Big)\rho(C,D). \quad (17)$$

$$q_3(\gamma,\mu,\lambda,\eta) = \sum_{i\in V}(\mu_i+\gamma_i). \quad (18)$$

Figure 3. Components of the dual cost function, $Q(\gamma,\mu,\lambda,\eta)$

– Running intersection property in Eq. (6): $\mu \in \mathbb{R}^V$.
– Edge constraints in Eq. (10): $\lambda \in \mathbb{R}^{\mathcal{E}}_+ \times \mathbb{R}^{\mathcal{E}}_+$.
– Clique constraints in Eq. (11): $\eta \in \mathbb{R}^{\mathcal{D}}_+$.

Therefore, the dual variables are $(\gamma,\mu,\lambda,\eta)$. We may then compute the dual function $Q(\gamma,\mu,\lambda,\eta)$ by introducing the Lagrange multipliers defined above.

As shown in the supplementary material, it is decomposed in three parts defined in Eq. (16), Eq. (17) and Eq. (18) respectively (see Figure 3):

$$Q(\gamma,\mu,\lambda,\eta) = q_1(\gamma,\mu,\lambda,\eta)+q_2(\gamma,\mu,\lambda,\eta)+q_3(\gamma,\mu,\lambda,\eta).$$

The dual functions $q_1(\gamma,\mu,\lambda,\eta)$ and $q_2(\gamma,\mu,\lambda,\eta)$ may be computed using the greedy algorithms defined in Section 3.1 and Section 3.2; q_1 can be evaluated in $O(r\log(r))$, where r is the cardinality of the space of cliques, $\mathcal{D}$, i.e., $\binom{n}{k+1}$ and q_2 can be evaluated in $O(m\log(m))$, where m is the cardinality of feasible edges, $\mathcal{E}$, i.e., $\binom{n}{k+2}.\binom{k+2}{2}$. This complexity is due to sorting the edges and hyperedges based on their weights (the costliest operation in each iteration). This leads to an overall complexity of $O(k^3 n^{k+2}\log n)$ per iteration of the projected supergradient method which we now present.

Projected supergradient ascent. The dual optimization problem defined by maximizing $Q(\gamma,\mu,\lambda,\eta)$ can be solved using the projected supergradient method. In each iteration t of the algorithm, the dual cost function, $Q(\gamma^t,\mu^t,\lambda^t,\eta^t)$, is evaluated through estimation of q_1 and q_2 by solving Eq. (16) and Eq. (17) respectively. In the process of solving these equations, the corresponding primal variables (τ^t,ρ^t) are also estimated and allows the computations of the supergradients of Q (i.e., opposites of subgradients of $-Q$) (see, e.g., Bertsekas, 1999). As shown in Algorithm 1, a step is made toward the direction of the supergradient and projection onto the positive orthant is performed for dual variables that are constrained to be nonnegative. With step sizes α_t proportional to $1/\sqrt{t}$, this algorithm is known to converge to a dual optimal solution (Nedić & Ozdaglar, 2009) at rate $1/\sqrt{t}$. Moreover, the average of all visited primal variables, i.e., after t steps, $(\hat{\tau}_t,\hat{\rho}_t) = \frac{1}{t}\sum_{u=0}^{t}(\tau^u,\rho^u)$ is known to be approximately primal-feasible (i.e., it satisfies all the linear constraints that were dualized up to a small constant that is also going to zero at rate $1/\sqrt{t}$). The convergence to primal feasibility is illustrated in Figure 6(a), where, on one of the synthetic examples from Section 5, the different constraint violations. Note that these are not the number of each of these constraints violated but the maximum value by which they are violated. It can be observed that the constraint violations reduce to zero over iterations.

Approximate greedy primal solution. Given an (approximate) primal-feasible but *fractional* solution $(\hat{\tau}_t,\hat{\rho}_t)$ of our convex optimization problem, we propose to find an integral candidate by discarding $\hat{\rho}_t$, and (a) ordering the components of $\hat{\tau}_t$ in decreasing order and (b) selecting the cliques C greedily with higher values of $\hat{\tau}_t(C)$ first, while maintaining decomposability of the resulting graph. The time complexity of the rounding algorithm is $O(n^{k+2})$. This is due to decomposability test with run time complexity $O(n^{k+1})$, that is performed while adding $O(n)$ cliques. See more details in the supplementary material.

5. Experiments and Results

In this section, we show the performance of the proposed algorithm on synthetic datasets and classical benchmarks.

Decomposable covariance matrices. In order to easily generate controllable distributions with entropies which are easy to compute, we use several decomposable graphs and we consider a Gaussian vec-

Algorithm 1 Projected Supergradient

Input: clique and edge entropies H, step-size constant a and number of iterations T
Output: sequence of clique and edge selections over iterations (τ^t, ρ^t)
Initialize $\gamma^0 = 0$, $\mu^0 = 0$, $\lambda^0 = 0$, $\eta^0 = 0$
for $t = 0$ **to** T **do**
  solve Eq. (16) and evaluate $q_1(\gamma^t, \mu^t, \lambda^t, \eta^t)$ to obtain τ^t
  solve Eq. (17) and evaluate $q_2(\gamma^t, \mu^t, \lambda^t, \eta^t)$ to obtain ρ^t
  update dual variables, $(\gamma^{t+1}, \mu^{t+1}, \lambda^{t+1}, \eta^{t+1})$ using supergradients and stepsize: $\alpha_t = \frac{a}{\sqrt{t}}$

$$\gamma_i^{t+1} = \Big[\gamma_i^t + \alpha_t\Big(1 - \textstyle\sum_{C\in\mathcal{D}} 1_{i\in C}\tau^t(C)\Big)\Big]^+$$
$$\mu_i^{t+1} = \mu_i^t + \alpha_t\Big(\textstyle\sum_{(C,D)\in\mathcal{E}} 1_{i\in(C\cap D)}\rho^t(C,D) - \sum_{C\in\mathcal{D}} 1_{i\in C}\tau^t(C) + 1\Big)$$
$$\lambda_{CD}^{t+1} = \Big[\lambda_{CD}^t + \alpha_t\Big(\tau^t(C) - \rho^t(C,D)\Big)\Big]^+$$
$$\eta_C^{t+1} = \Big[\eta_C^t + \alpha_t\Big(\textstyle\sum_{(C,D)\in\mathcal{E}} \rho^t(C,D) - \tau^t(C)\Big)\Big]^+$$

  end for

tor with covariance matrix Σ, generated as follows: (a) sample a matrix Z of dimensions $n \times d'$ with entries uniform in $[0, 1]$ and consider the matrix $\Sigma' = \frac{d}{d'}ZZ^\top + (1 - \frac{d}{d'})I$, (b) normalize Σ' to unit diagonal, and (c) find the projection of Σ' onto the graph G, i.e., find the covariance matrix that factorizes in G and which is closest to Σ' in Kullback-Leibler divergence. The last operation may be performed in closed form (Lauritzen, 1996), while the first operation leads to strong correlations when d is large, and small correlations when d is small.

We use the graph structures representing a *chain junction tree* as in Figure 4 and a *star junction tree* as in Figure 5 to analyze the performance of our algorithm for decomposable covariance matrices generated with different correlations.

Table 1 and Table 2 show the performance of our algorithm on these two graphs. Decomposable covariance matrices are generated as above with different values of the correlation parameter d (all averaged over ten different random covariance matrices). We show the difference between the cost function in Eq. (2) and the optimal entropy, i.e., the one of the actual structure represented by the covariance matrices. The differences in the table are multiplied by 10^3 for brevity.

The first column ΔDual represents the optimal value of our convex relaxation (obtained from the dual function), while the second column ΔDual^r represents the optimal value by replacing the hyperforest constraint by the simply $\tau \in [0, 1]^{\mathcal{D}}$. We can see from the two tables, that the two values are strictly negative (i.e., we indeed have a relaxation) and that the hyperforest constraint is key to obtaining tighter relaxations. Note that the associated solutions are only fractional.

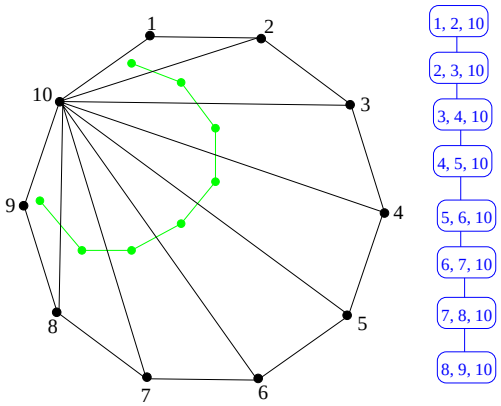
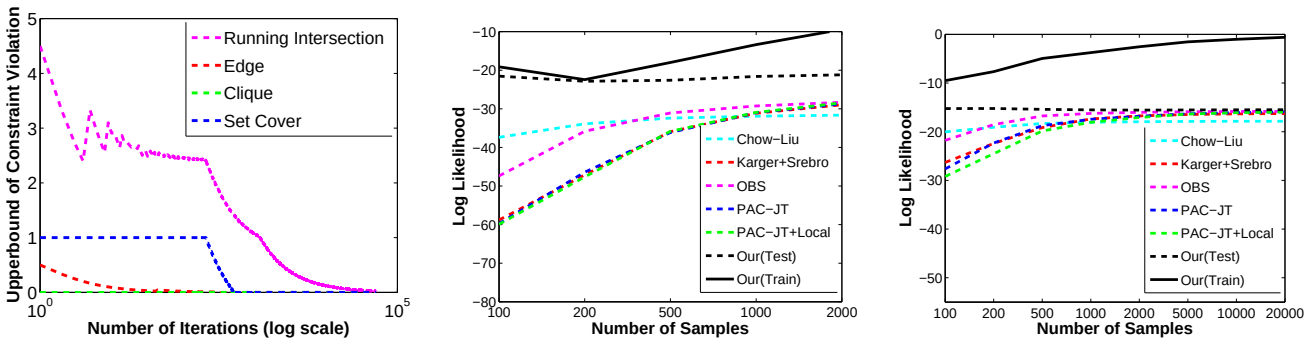


Figure 4. Graph representing a chain junction tree with an embedded junction tree in green and its junction tree representation in blue.

The third column ΔPrimal represents the cost function after the rounding step; it is compared in the fourth column with a simple greedy algorithm that sorts all mutual information and keep adding the cliques with largest mutual information as long as decomposability is maintained. Although the relaxation is not tight, our rounding scheme leads empirically to the optimal solution when the correlations are strong enough (i.e., large values of d) and outperforms the simple greedy algorithm.

Table 1. Performance on chain junction trees.

d	ΔDual	ΔDual^r	ΔPrimal	ΔGreedy
1	-0.7±0.1	-32.7±16.4	0.2±0.1	0.2±0.1
2	-0.4±0.1	-23.4±9.6	0	0.5±0.2
4	-1.1±0.1	-31.2±9.7	0	1.9±0.3
8	-0.6±0.1	-23.9±9.8	0	7.9±0.3
16	-1.9±0.2	-3.4±2.7	0	25.6±1.2
32	-2.9±0.5	-3.2±0.3	0	57.3±1.5



(a) (b) (c)

Figure 6. Left: (a) Upper bound of constraint violations for d=2 and a chain junction tree. Right: Log likelihood of the structures learnt using various algorithms on (b) TRAFFIC and (c) ALARM datasets with $k = 3$ except Chow-Liu ($k = 1$). Note that the performance is better with higher values as we compare log-likelihoods.

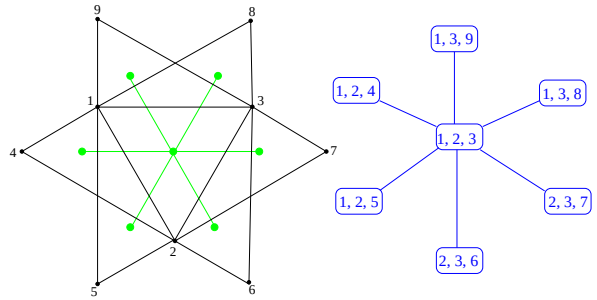


Figure 5. Graph representing a star junction tree with an embedded junction tree in green and its junction tree representation in blue.

Table 2. Performance on star junction trees.

d	ΔDual	ΔDual^r	ΔPrimal	ΔGreedy
1	-0.8±0.1	-31.4±13.4	0.2±0.1	0.9±0.1
2	-0.5±0.2	-26.6±13.3	0	0.4±0.3
4	-0.3±0.0	-16.6±4.1	0	1.7±0.2
8	-0.4±0.0	-16.0±9.6	0	6.9±0.3
16	-1.2±0.5	-3.1±0.3	0	26.3±1.5
32	-6.8±0.4	-8.5±1.2	0	58.3±1.9

Performance Comparison. We compare the quality of the graph structures learned by the proposed algorithm with the ones produced by Ordering Based Search (OBS) (Teyssier & Koller, 2005), the combinatorial optimization algorithm proposed by Karger and Srebro (Karger+Srebro) (Karger & Srebro, 2001), the Chow-Liu trees (Chow-Liu) (Chow & Liu, 1968) and different variations of PAC-learning based algorithms (PAC-JT, PAC-JT+local) (Chechetka & Guestrin, 2007). We use a real-world dataset, TRAFFIC (Krause & Guestrin, 2005) and an artificial dataset, ALARM (Beinlich et al., 1989) to compare the performances of these algorithms.

This ALARM dataset was sampled from a known Bayesian network (Beinlich et al., 1989) of 37 nodes, which has a treewidth equal to 4. We learn an approximate decomposable graph of treewidth 3. The TRAFFIC dataset is the traffic flow information every 5 minutes for a month at 8000 locations in California (Krause & Guestrin, 2005). The traffic flow information is collected at 32 locations in San Francisco Bay area and the values are discretized into 4 bins. We learn an approximate decomposable graph of treewidth 3 using our approach. Empirical entropies are computed from the generated samples of each data set and we infer the underlying structure from them using our algorithm. Figure 6(b) and Figure 6(c) show the log-likelihoods of structures learnt using various algorithms on Traffic and Alarm datasets respectively. These figures illustrate the gains of the convex approach over the earlier non-convex approaches.

6. Conclusion and Future Work

In this paper, we have provided a convex relaxation for learning the maximum likelihood decomposable graph with bounded treewidth, in polynomial-time, which empirically outperforms previously proposed algorithms. We are currently exploring two avenues for improvements: (a) design sufficient conditions for tightness of our relaxation, following the recent literature on relaxation of variable selection problems (see, e.g., Candès & Tao, 2005), and (b) use heuristics to speed-up the algorithms to allow application to larger graphs.

Acknowledgements. We acknowledge support from the European Research Council grant SIERRA (project 239993), and would like to thank Anton Chechetka, Carlos Guestrin, Nathan Srebro and Percy Liang for sharing code and datasets.

References

Bach, F. R. and Jordan, M. I. Thin junction trees. In *Adv. NIPS*, 2002.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., and Cooper, G. F. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In *Proc. of Euro. Conf. on AI in Medicine*, 1989.

Bertsekas, D. P. *Nonlinear Programming*. Athena Scientific, 1999.

Bishop, C. M. et al. *Pattern recognition and machine learning*. Springer New York, 2006.

Candès, E. J. and Tao, T. Decoding by linear programming. *IEEE Trans. Inf. Theory*, 51(12):4203–4215, 2005.

Chechetka, A. and Guestrin, C. Efficient principled learning of thin junction trees. In *Adv. NIPS*, 2007.

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

Cover, T. M. and Thomas, J. A. *Elements of information theory*. John Wiley & Sons, 2006.

Deshpande, A., Garofalakis, M. N., and Jordan, M. I. Efficient stepwise selection in decomposable models. In *Proc. UAI*, 2001.

Frank, A., Király, T., and Kriesell, M. On decomposing a hypergraph into k connected subhypergraphs. *Discrete Applied Mathematics*, 131(2): 373–383, 2003.

Friedman, N. and Koller, D. Being Bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 50(1):95–125, 2003.

Fukunaga, T. Computing minimum multiway cuts in hypergraphs from hypertree packings. *Integer Prog. Comb. Optimization*, pp. 15–28, 2010.

Gogate, V., Webb, W. A., and Domingos, P. Learning efficient Markov networks. In *Adv. NIPS*, 2010.

Karger, D. and Srebro, N. Learning Markov networks: maximum bounded tree-width graphs. In *Proc. ACM-SIAM symposium on Discrete algorithms*, 2001.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kolmogorov, V. and Schoenemann, T. Generalized sequential tree-reweighted message passing. *ArXiv e-prints*, May 2012.

Krause, A. and Guestrin, C. Near-optimal nonmyopic value of information in graphical models. In *Proc. UAI*, 2005.

Lauritzen, S. L. *Graphical Models*. Oxford University Press, Oxford, 1996.

Lorea, M. Hypergraphes et matroides. *Cahiers Centre Etud. Rech. Oper*, 17:289–291, 1975.

Malvestuto, F. M. Approximating discrete probability distributions with decomposable models. *IEEE Trans. Systems, Man, Cybernetics*, 21(5), 1991.

Narasimhan, M. and Bilmes, J. PAC-learning bounded tree-width graphical models. In *Proc. UAI*, 2004.

Nedić, A. and Ozdaglar, A. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM J. Opt.*, 19(4), February 2009.

Saul, L. and Jordan, M. I. Exploiting tractable substructures in intractable networks. In *Adv. NIPS*, 1995.

Schrijver, A. *Combinatorial optimization: Polyhedra and efficiency*. Springer, 2004.

Shahaf, D., Chechetka, A., and Guestrin, C. Learning thin junction trees via graph cuts. In *Proc. AISTATS*, 2009.

Spirtes, P., Glymour, C., and Scheines, R. *Causation, prediction, and search*, volume 81. MIT press, 2001.

Srebro, N. Maximum likelihood bounded tree-width Markov networks. In *Proc. UAI*, 2002.

Szántai, T. and Kovács, E. Discovering a junction tree behind a Markov network by a greedy algorithm. *ArXiv e-prints*, April 2011.

Szántai, T. and Kovács, E. Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. *Annals OR*, 193(1), 2012.

Teyssier, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proc. UAI*, 2005.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Found. and Trends in Machine Learning*, 1(1-2), 2008.