# Optimization on Seperator Trees

**Mukund Narasimhan** *and* **Jeff Bilmes**
*Dept. of Electrical Engineering*
*University of Washington*
*Seattle, WA 98195*

UWEE Technical Report
Number UWEETR-2004-0018
March 2004

# Optimization on Seperator Trees

**Mukund Narasimhan** and **Jeff Bilmes**
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195

*University of Washington, Dept. of EE, UWEETR-2004-0018*
March 2004

**Abstract**

Given a chordal graph $G$, and a class of subgraphs $\mathcal{H}$ of $G$, we investigate the problem of finding an element $H \in \mathcal{H}$ which minimizes a function $f : \mathcal{H} \to \mathbb{R}$. We show that when $f$ satisfies a decomposability criterion, and when $\mathcal{H}$ admits what we call an efficiently decomposable configuration space with respect to $f$, then this optimization problem can be solved in polynomial time. We give a dynamic programming formulation for solving this problem, which uses a representation of of $G$ called a seperator-tree. The seperator-clique tree is very closely related to the clique-tree representation for $G$, and we give a procedure for constructing a seperator-clique tree from a clique-tree. Gavril [1] and Tarjan [2] have used graph separation properties to solve several combinatorial optimization problems when the size of the minimal separators in the graph is bounded. Our technique is an extension of theirs, and it does not require the minimal separators of $G$ to be of bounded size. We give examples to show that this framework can be used to solve a number of problems in machine learning and statistics that require the reduction of complexity of graphical models.

## 1 Introduction and Preliminaries

The problem considered in this paper is motivated by the need to reduce the complexity of inference in graphical models. The complexity of inference in graphical model $(G, P_G)$ is typically exponential in the size of the largest clique of $G$. Therefore, when inference has to be performed in computation limited situations, such as for embedded or real-time applications, we often try to find a subgraph $H$ of $G$ such that the size of the largest clique in $H$ is smaller than that of $G$. If $\mathcal{H}$ is the set of subgraphs of $G$ consisting of graphs for which inference can be performed in reasonable time, we seek to find an element of $\mathcal{H}$ for which the distortion produced in the result by using $H$ instead of $G$ is minimized. Using $H \in \mathcal{H}$ instead of $G$ for inference results in approximate results, and we seek a $H \in \mathcal{H}$ for which the approximation is best (or distortion is least). The KL-divergence between $P_G$ and a probability distribution $P_H$ that factorizes over $H$ is often used to measure the impact of the using $H$ instead of $G$ on inference. Existing techniques for reducing the complexity of graphical models including edge-removal [4] and annihilation [3] are greedy in nature and cannot make any guarantees regarding the optimality of the solution. In general, if $\mathcal{H}$ is an arbitrary set of subgraphs of $G$, then the problem of finding the optimal element of $\mathcal{H}$ is NP-complete, even when the KL-divergence is used as the optimization criterion.

In this report, we show that for an arbitrary optimization function that satisfies a decomposbility criterion, we can find the optimal element of $\mathcal{H}$ as long as $\mathcal{H}$ satisfies some conditions we describe in Section 3. We show that KL-divergence satisfies this decomposability, as do several other naturally occuring objective functions, and hence we can compute optimal subgraphs from amongst several classes of subgraphs of $G$. In the remainder of this section, we introduce some notation and terminology we will be using. Then in section 2 we describe configuration spaces, and efficiently decomposable configuration spaces. Then in Section 3, ...

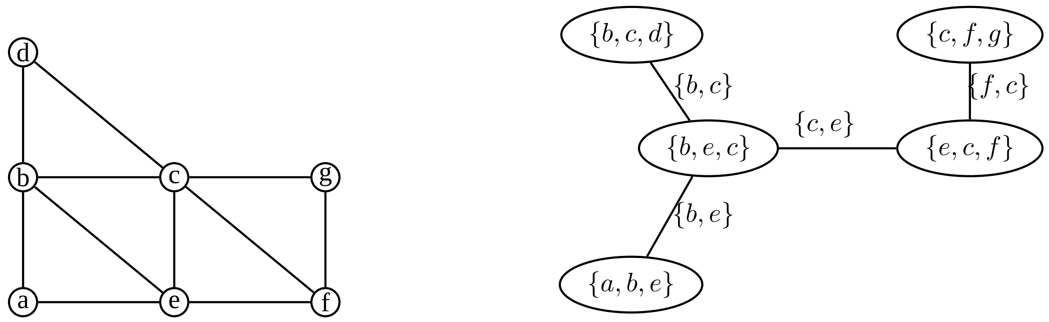

Figure 1: A chordal graph and its clique-tree

## 2 Notation and Terminology

A graph $G = (V, E)$ is said to be triangulated (or chordal) if every cycle of length greater than 3 has a chord. A clique of $G$ is a non-empty set $S \subseteq V$ such that $\{a, b\} \in E$ for all $a, b \in S$. A clique $S$ is maximal if $S$ is not properly contained in another clique. If $\alpha$ and $\beta$ are non-adjacent vertices of $G$ then a set of vertices $S \subseteq V \setminus \{\alpha, \beta\}$ is called an $(\alpha, \beta)$-separator if $\alpha$ and $\beta$ are in distinct components of $G[V \setminus S]$. $S$ is a minimal $(\alpha, \beta)$-separator if no proper subset of $S$ is an $(\alpha, \beta)$-separator. $S$ is said to be a minimal separator if $S$ is a minimal $(\alpha, \beta)$-separator for some non adjacent $a, b \in V$. When $G$ is triangulated, $G$ can be thought of as the intersection graph of the family of subtrees of a tree [6]. A junction-tree $T = (\mathcal{K}, \mathcal{S})$ (or clique-tree) is one such tree, and the vertices $\mathcal{K}$ of $T$ correspond to the maximal-cliques of $G$, while the edges correspond to minimal separators of $G$. If $G$ is triangulated, then the number of maximal cliques is at most $|V|$. For example, in the graph $G$ shown in Figure 1, $\mathcal{K} = \{\{b, c, d\}, \{a, b, e\}, \{b, e, c\}, \{e, c, f\}, \{c, f, g\}\}$. The edges $\mathcal{S}$ of $T$ correspond to minimal-separators in the following way. If $V_i V_j$ is an edge in $T$ (where $V_i, V_j \in \mathcal{K}$ and hence are cliques of $G$), then $V_i \cap V_j \neq \phi$. We label each edge $V_i V_j \in \mathcal{S}$ with the set $V_{ij} = V_i \cap V_j$, which is a non-empty clique in $G$. Note that, in particular, the intersection of any two cliques is not necessarily a minimal separator (e.g., $\{b, e, c\} \cap \{c, f, g\} = \{c\}$ which is not a minimal vertex separator). A graph $G$ is a $k$-tree if $|V| = k+1$ for every $V \in \mathcal{K}$.

Let $V = \{1, 2, \ldots, n\}$. For the remainder of the paper, all graphs will have vertex set $V$, and we will consider probability distributions $P$ defined on the set of random variables corresponding to the vertices $V$. If $G$ is a chordal graph with junction tree/clique tree/tree-decomposition $(T = (\mathcal{K}, \mathcal{S}), \{V_i\}_{i \in \mathcal{K}})$, we say that $P$ factors over $G$ if

$$P\left(\{X_v\}_{v \in V}\right) = \frac{\prod_{i \in \mathcal{K}} P\left(\{X_v\}_{v \in V_i}\right)}{\prod_{ij \in \mathcal{S}} P\left(\{X_v\}_{v \in V_{ij}}\right)}$$

If $P$ is any probability distribution, then the distribution $P_G$ that minimizes $D(P \| P_G)$ is given by

$$P_G\left(\{X_v\}_{v \in V}\right) = \frac{\prod_{i \in \mathcal{K}} P\left(\{X_v\}_{v \in V_i}\right)}{\prod_{ij \in \mathcal{S}} P\left(\{X_v\}_{v \in V_{ij}}\right)}$$

## 3 Separator-decompositions and Separator-trees

For the remainder of the paper, we will be assuming that $G = (V, E)$ is some triangulated graph, with clique tree $T = (\mathcal{K}, \mathcal{S})$. Many algorithms, such as the junction tree algorithm, are most naturally thought of in terms of the clique tree of $G$. We will find a different representation of the graph, the *separator-clique tree* representation, more useful for our purpose. The separator-clique tree is very closely related to the clique tree. In this section and the next, we show how a separator-clique tree for a graph may be constructed, and show how it is used to solve a number of optimization problems. We begin with this result from [5].

**Lemma 1.** *If $G = (V, E)$ is a triangulated graph, then a set $S \subseteq V$ is a minimal separator if and only if in every clique tree $T = (\mathcal{K}, \mathcal{S})$, there is some edge $ij \in \mathcal{S}$ such that $S = V_{ij} = V_i \cap V_j$.*

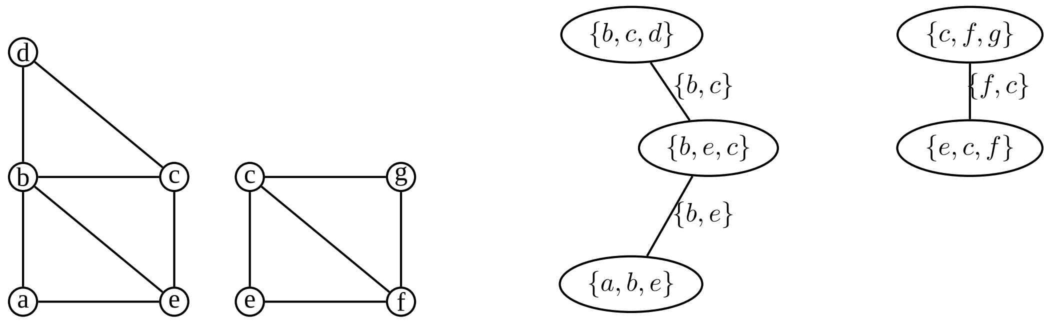

Figure 2: Separation induced by the separator $S = \{c, e\}$

Therefore, the set of minimal separators of $G$ are precisely the set of labels of edges in any clique tree for $G$. Now, if $T = (\mathcal{K}, \mathcal{S})$ is a clique tree for $G$, then the removal of any edge $ij \in \mathcal{S}$ disconnects $T$, and correspondingly, the removal of the set of vertices $V_{ij}$ associated with the edge disconnects $G$. Thus each edge $ij \in \mathcal{S}$ specifies a decomposition of $G$ into two (vertex induced) subgraphs. These two subgraphs are also chordal graphs. The two subgraphs have as clique-trees the two sub-trees of $T$ induced by the removal of the edge $ij \in \mathcal{S}$. Therefore, we can recursively decompose the graph into smaller and smaller subgraphs till the resulting subgraphs are cliques. When the size of all the minimal separators are bounded, we may use these decompositions to easily solve problems that are hard in general. For example, in [7], it is shown that NP-complete problems like graph coloring, and finding maximal independent sets can be solved in polynomial time on graphs with bounded tree-width (which are equivalent to to graph with bounded size separators). Roughly speaking, the reason why bounded size separators help is this : The minimal separators specify decompositions of the graph $G$ into parts that depend on each other only through the separator. Therefore, if the nature of the problem is such that we can compute solutions to the (possibly slightly modified) individual parts, and then *splice* the parts together to get a solution to the original problem, then we can apply straight forward recursive or dynamic programming algorithms to solve the problem in polynomial time. In this paper, we will *not* assume that the the tree-width of the graphs are bounded. Instead, we will show that we may we may obtain some of these benefits even when the tree-width of the graph is large, in certain cases. We do this by introducing a surrogate, which we call the configuration space, and show that picking an optimal graph $H$ from a class of graphs $\mathcal{H}$ can also be done in polynomial time when the configuration space satisfies certain conditions.

As Lemma 1 shows, we can associate each minimal separator of $G$ with one (or more) edges of a clique tree for $G$. In the framework that we will present, separators play a more central role than cliques. Therefore, we will consider an alternate representation, a rooted tree which we call the *separator-clique tree*, in which vertices correspond to separators and cliques of $G$. While we can specify the construction of the separator-clique tree independently of the clique tree representation of $G$, it is easier to describe (and constuct) in terms of a clique tree for $G$. As indicated by Lemma 1, each edge of $S \in \mathcal{S}$ of the clique-tree $T = (\mathcal{K}, \mathcal{S})$ corresponds to a seperator of $G$. The deletion of this edge of $T$ will disconnect $T$ into two sub-trees, and will also disconnect $G$. For example, Figure 2 shows the result of the removal of the edge corresponding to the separator $S = \{c, e\}$ on the graph shown in Figure 1 Therefore, each separator of $G$ induces a decomposition of $G$. Each one of these components are also chordal graphs (as they are vertex induced subgraphs of $G$), and the decompososed sub-trees of $T$ are in fact clique-trees for the components of $G$. Therefore, we can continue to decompose the graphs so obtained till we end up with a graph with no separators (i.e., a graph which is complete). Therefore, we have a hierarchial structure of decompositions of $G$, each associated with a minimal separator of $G$, or alternatively, with an edge of $T$. A natural way of representing this hierarchy of decompositions is with a rooted binary tree, which we will call a separator-clique tree, and denote by $R$. The interior vertices of $R$ correspond to $\mathcal{S}$ (the edges of $T$ and hence the minimal separators of $G$) and the leaves of $R$ corespond to $\mathcal{K}$ (the vertices of $T$ and hence the maximal cliques of $G$). The interior vertices of $R$ represents decompositions of subgraphs of $G$. Before we describe the construction of $R$, we establish some additional notation. If $i, j \in \mathcal{K}$ and $ij \in \mathcal{S}$ is an edge of the clique-tree $T$, the the removal of the edge $ij$ disconnects $T$ ( and $G$). For example, if $i \leftrightarrow \{b, e, c\}$ and $j \leftrightarrow \{e, c, f\}$, then $ij \leftrightarrow \{c, e\}$. Then the removal of the edge $ij$ disconnects $T$ as shown in Figure 2. We call the two (disjoint) clique-trees that result $T^{(i)}$ and $T^{(j)}$. We will denote by $\mathcal{K}^{(i)}$ and $\mathcal{K}^{(j)}$ the vertices of $T^{(i)}$ and $T^{(j)}$, and by $G^{(i)}$ and $G^{(j)}$ the (chordal) subgraphs corresponding to the clique tree $T^{(i)}$ and $T^{(j)}$. Then the vertex sets of $G^{(i)}$ and $G^{(j)}$ are $V^{(i)} = \cup_{k \in \mathcal{K}^{(i)}} V_k$ and $V^{(j)} = \cup_{k \in \mathcal{K}^{(j)}} V_k$ respectively. If $G$ is not a complete graph, then $G$ has a minimal separator, and so the separator-clique tree $R$ has an interior vertex. The root $r$ of $R$ represents a decompostion induced by a separator $S \in \mathcal{S}$. We label $r$ by the triple $(V^{(i)}, ij, V^{(j)})$. Since $G^{(i)}$ and $G^{(j)}$ are also chordal graphs, we can find separator trees $R^{(i)}$ and $R^{(j)}$ (with roots $r^{(i)}$ and $r^{(j)}$) for $G^{(i)}$ and $G^{(j)}$ respectively. We then construct $R$ by making $r^{(i)}$ and $r^{(j)}$ the left and right child of $r$. This procedure is illustrated in Figure 3. The algorithm is formally specified as Algorithm 1. To ensure that adjacencies in $R$ correspond to adjacencies in $T$, the

```
Algorithm 1: separator_tree
  Data: A graph $G = (V, E)$, a clique-tree $T = (\mathcal{K}, \mathcal{S})$ for $G$ and an element $ij \in \mathcal{S}$ (if $\mathcal{S} \neq \phi$).
  Result: A separator-clique tree $R$ for $T$ (with root $r$ corresponding to $ij$ if $ij$ is specified).
  if $T$ has at least 2 vertices then
      $(T^{(i)}, T^{(j)}) \longleftarrow$ decomposition corresponding to $ij \in \mathcal{S}$;
      Pick edge $il$ of $T^{(i)}$ (if $G^{(i)}$ is not a clique);
      Pick edge $jk$ of $T^{(j)}$ (if $G^{(j)}$ is not a clique);
      $R^{(i)} \longleftarrow$ separator_tree$(G^{(i)}, T^{(i)}, il)$; $R^{(j)} \longleftarrow$ separator_tree$(G^{(j)}, T^{(j)}, jk)$;
      $R \longleftarrow$ tree with root $r \leftrightarrow (V^{(i)}; ij; V^{(j)}$ and children $r^{(i)}$ and $r^{(j)}$;
  else
      // $G$ is a complete graph, and $T$ has just one vertex. The third parameter is ignored.
      return trivial tree with a single vertex corresponding the clique $G$.
```

separator for the root of each subtree is picked so that it is adjacent to the separator corresponding to the root. This also ensures that $R$ satisfies the Helly property [6]. The effect of this procedure on the clique tree of Figure 1 is shown in Figure 3, where the decomposition associated with the interior vertices is shown inside the vertices.

One advantage of this form of representation is that it corresponds very naturally to our notion of a decomposable probability distribution. For any internal vertex $ij \in \mathcal{S}$, we will let $\mathsf{l}(ij)$ and $\mathsf{r}(ij)$ be the left and right children of $ij$. We will call the trees rooted at the left and right children of $ij$ to be the left and right child-tree of $ij$, denoted $\mathsf{tl}(ij)$ and $\mathsf{tr}(ij)$. Again we note that the leaves of $R$ correspond to $\mathcal{K}$, the maximal cliques of $G$, and the interior vertices of $R$ correspond to $\mathcal{S}$, the minimal separators of $G$. We will denote by $\mathcal{M} = \mathcal{K} \cup \mathcal{S}$, the set of all vertices of $R$.

## 4 Decomposable subgraph classes and decomposable objective functions

Let $\mathcal{H}$ be a set of subgraphs of $G$. Then for any $H \in \mathcal{H}$, and any minimal separator $ij \in \mathcal{S}$ of $G$, $S$ is also separator of $H$. We will denote by $H^{(i)}$, $H^{(j)}$ and $H^{(ij)}$ the graphs $H[V^{(i)}]$, $H[V^{(j)}]$ and $H[V_{ij}]$ respectively. Therefore, $H^{(i)}$ is a subgraph of $G^{(i)}$, $H^{(j)}$ a subgraph of $G^{(j)}$ and $H^{(ij)}$ is a subgraph of $G^{(ij)} = V_{ij}$.

**Definition 1.** *A set $\mathcal{H}$ of subgraphs of $G$ is a decomposable set of subgraphs of $G$ if whenever $H_1, H_2 \in \mathcal{H}$ with $H_1^{(ij)} = H_2^{(ij)}$, then there is a $H \in \mathcal{H}$ with $H^{(i)} = H_1^{(i)}$, $H^{(j)} = H_2^{(j)}$ and $H^{(ij)} = H_1^{(ij)} = H_2^{(ij)}$.*

The algorithms we present will recursively build up subgraphs of $G$ from subgraphs of $G^{(i)}$ and $G^{(j)}$. Decomposibility of $\mathcal{H}$ ensures that we can piece together arbitrary subgraphs of $G^{(i)}$ and $G^{(j)}$ as long as the subgraphs are compatible in the sense that they agree on the vertex set $V^{(ij)}$. Thus, $\mathcal{H}^* = \left\{ H^{(i)} : H \in \mathcal{H} \text{ and } ij \in \mathcal{S} \right\}$ represents the collection of basic building blocks which we use to build subgraphs of $G$ from. We also define $\mathcal{H}^{\mathcal{S}} = \{ H[V_{ij}] : H \in \mathcal{H} \text{ and } ij \in \mathcal{S} \}$.

**Definition 2.** *A function $f : \mathcal{H}^* \to \mathbb{R}$ is said to be decomposable if there is some $g : \mathcal{H}^{\mathcal{S}} \to \mathbb{R}$ such that for any separator $ij \in \mathcal{S}$ and any $H \in \mathcal{H}$, we have*

$$f(H) = f(H^{(i)}) + f(H^{(j)}) + g(H^{(ij)})$$

The rationale for decomposable functions will become clear from Lemma 4. Several commonly used criteria, including the KL-divergence discussed earlier, are decomposable objective functions as the following results show.

**Lemma 2.** *Suppose that $(G, P_G)$ is a graphical model, $H$ is a chordal subgraph of $G$, and $P_H$ factors over $H$. Then*

$$D(P_G \| P_H) = D(P_{G^{(i)}} \| P_{H^{(i)}}) + D(P_{G^{(j)}} \| P_{H^{(j)}}) - D(P_{G[V_{ij}]} \| P_{H[V_{ij}]})$$

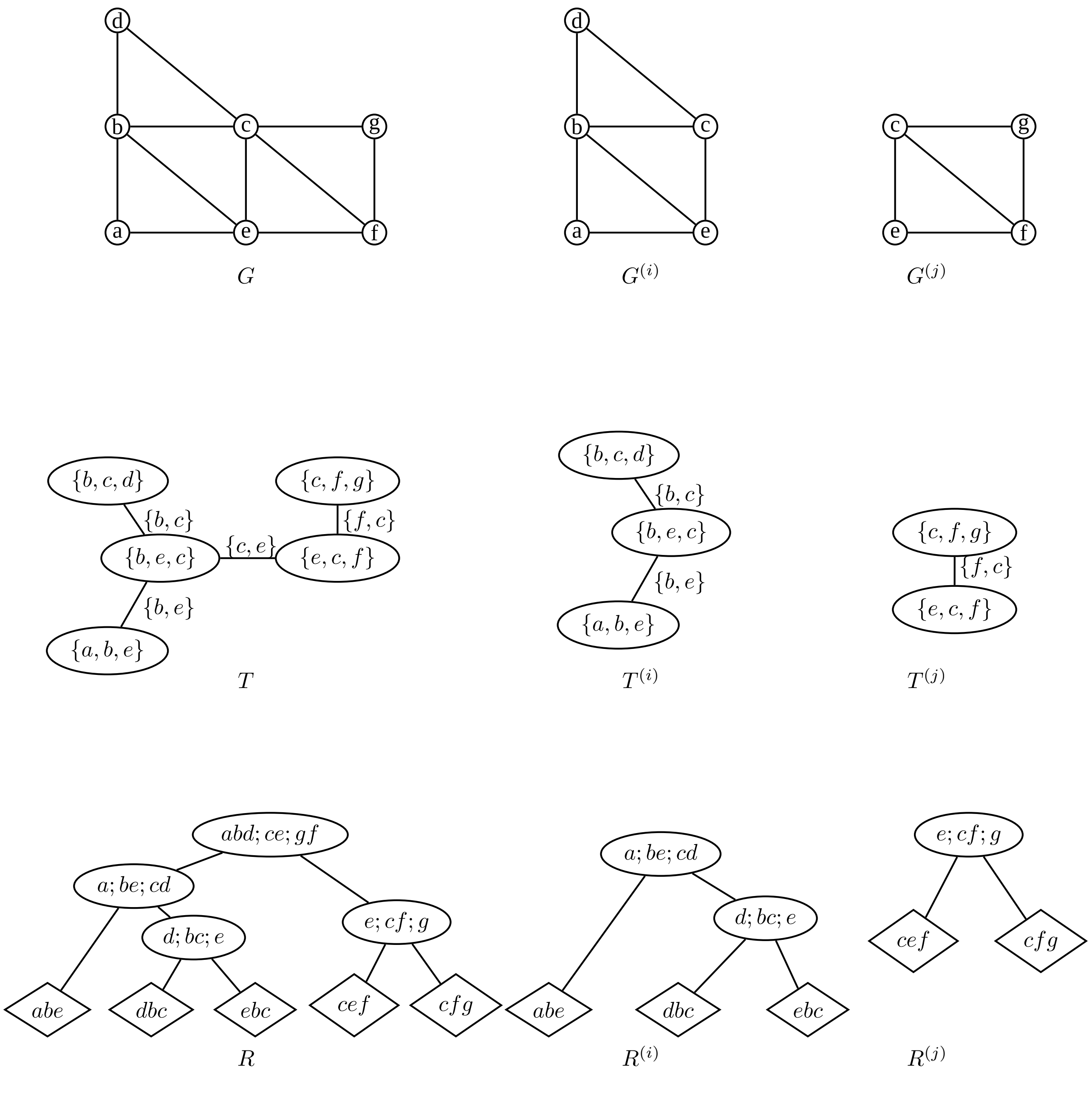

Figure 3: Recursive construction of $R$

*Proof.* Since $H$ is a subgraph of $G$, and $V_{ij}$ is a separator of $G$, $V_{ij}$ must also be a separator of $H$. Therefore, $P_H\left(\{X_v\}_{v\in V}\right) = \frac{P_{H^{(i)}}(\{X_v\}_{v\in V^{(i)}})\cdot P_{H^{(j)}}(\{X_v\}_{v\in V^{(j)}})}{P_{H[V_{ij}]}}$. Therefore,

$$\begin{aligned}
D(P_G\|P_H) &= E_{P_G}\left[\log \frac{P_G\left(\{X_v\}_{v\in V}\right)}{P_H\left(\{X_v\}_{v\in V}\right)}\right] \\
&= E_{P_G}\left[\log \frac{\frac{P_{G^{(i)}}(\{X_v\}_{v\in V^{(i)}})\cdot P_{G^{(j)}}(\{X_v\}_{v\in V^{(j)}})}{P_{G[V_{ij}]}}}{\frac{P_{H^{(i)}}(\{X_v\}_{v\in V^{(i)}})\cdot P_{H^{(j)}}(\{X_v\}_{v\in V^{(j)}})}{P_{H[V_{ij}]}}}\right] \\
&= E_G\left[\log \frac{P_{G^{(i)}}(\{X_v\}_{v\in V^{(i)}})}{P_{H^{(i)}}(\{X_v\}_{v\in V^{(i)}})}\right] + E_G\left[\log \frac{P_{G^{(j)}}(\{X_v\}_{v\in V^{(j)}})}{P_{H^{(j)}}(\{X_v\}_{v\in V^{(j)}})}\right] - E_G\left[\log \frac{P_{G^{(ij)}}(\{X_v\}_{v\in V^{(ij)}})}{P_{H^{(ij)}}(\{X_v\}_{v\in V^{(ij)}})}\right] \\
&= D(P_{G^{(i)}}\|P_{H^{(i)}}) + D(P_{G^{(j)}}\|P_{H^{(j)}}) - D(P_{G[V_{ij}]}\|P_{H[V_{ij}]})
\end{aligned}$$

□

**Corollary 3.** *Suppose that $(G, P_G)$ is a graphical model, with clique-tree $T = (\mathcal{K}, \mathcal{S})$ and $\mathcal{H}$ is a decomposable collection of chordal subgraphs of G. If $f : \mathcal{H} \to \mathbb{R}$ is defined by $f(H) = \inf_{\{P_H : P_H \text{ factors over } H\}} D(P_G\|P_H)$, then $f$ is a decomposable function.*

*Proof.* For any $H \in \mathcal{H}$, let $P_H$ a distribution on $\{X_v\}_{v\in V}$ such that $D(P_G\|P_H)$ is minimized, and $P_H$ factors over $H$ (such a distribution always exists and is unique). Then take $g : \mathcal{H}^{\mathcal{S}} \to \mathbb{R}$ to be the function $g(H^{(ij)}) = -D(P_{G[V_{ij}]}\|P_{H[V_{ij}]})$. □

In fact, if we use use a Bregman divergence to measure the approximation loss between two probability distributions instead of the KL-divergence (which is just a special case), we still get a decomposable function. Several other functions are also decomposable, such as description length which also allow for MDL type optimization critera.

## 5 Optimizing decomposable functions

We will now consider the problem of finding an element $H \in \arg\min_{H\in\mathcal{H}} f(H)$, where $f$ is a decomposable function.

**Lemma 4.** *Suppose that $\mathcal{H}$ is a decomposable class of subgraphs of $G$, and $f$ a decomposable function. If $H_1 \in \arg\min_{H\in\mathcal{H}} f(H)$, where $f$ is a decomposable function. If $H_2 \in \mathcal{H}$ with $H_1^{(ij)} = H_2(^{ij})$, then we must have $f(H_1^{(i)}) \le f(H_2^{(i)})$ and $f(H_1^(j)) \le f(H_2^{(j)})$.*

*Proof.* Since $f$ is decomposable, we must have $f(H_1) = f(H_1^{(i)}) + f(H_1^{(j)}) + g((H_1^{(ij)}))$, and $f(H_2) = f(H_2^{(i)}) + f(H_2^{(j)}) + g((H_2^{(ij)}))$. Since $H_1^{(ij)} = H_2^{(ij)}$, we have $f(H_1^{(i)}) + f(H_1^{(j)}) \le f(H_2^{(i)}) + f(H_2^{(j)})$. If the claim does not hold, then either $f(H_1^{(i)}) > f(H_2^{(i)})$ or $f(H_1^{(j)}) > f(H_2^{(j)})$. If $f(H_1^{(i)}) > f(H_2^{(i)})$, then by the decomposability of $\mathcal{H}$, we can find a $H_3 \in \mathcal{H}$ with $H_3^{(i)} = H_2^{(i)}$, $H_3^{(j)} = H_1^{(j)}$ and $H_3^{(ij)} = H_1^{(ij)}$. Therefore,

$$\begin{aligned}
f(H_3) &= f(H_3^{(i)}) + f(H_3^{(j)}) + g(H_3^{(ij)}) \\
&= f(H_2^{(i)}) + f(H_1^{(j)}) + g(H_1^{(ij)}) \\
&< f(H_1^{(i)}) + f(H_1^{(j)}) + g(H_1^{(ij)}) \\
&= f(H_1)
\end{aligned}$$

This contradicts the optimality of $H_1$. □

Lemma 4 tells us that we can reformulate the problem of finding an optimal subgraph $H \in \mathcal{H}$ as one of finding subgraphs of $H^{(i)} \subseteq G^{(i)}$ and $H^{(j)} \subseteq G^{(j)}$ that are compatible, in the sense that they match up on the overlap $V_{ij}$, and for which the expression $f(H^{(i)}) + f(H^{(j)}) + g(H^{(ij)})$ is minimized. Requiring $\mathcal{H}$ to be decomposable is overly restrictive, and indeed, several interesting classes of subgraphs of $G$ are not decomposable. We therefore seek an alternative that will still preserve the essential advantages of decomposibility. Decomposibility allows us to find an optimal subgraph of $G$ by finding an optimal subgraph of $H^{(i)}$ of $G^{(i)}$ and by finding an optimal subgraph $H^{(j)}$ of $G^{(j)}$ that are compatible. While we will formally define a notion of a configuration later, it makes intuitive sense that if for every possible configuration of the overlap $V_{ij}$, we can find the optimal subgraphs $H^{(i)}$ of $G^{(i)}$ and $H^{(j)}$ of $G^{(j)}$ which are compatible with the configuration, then we can simply enumerate all possible configurations of $ij$ and pick the optimal one to get the optimal subgraph $H \subseteq G$. Therefore the key to formulating this as a divide-and-conquer algorithm is this : Each configuration of the vertices in the separator acts as a constraint on the optimization of the two subgraphs, and we can find optimal subgraphs of $G$ if we can find constrained optimal subgraphs of $G^{(i)}$ and $G^{(j)}$. Therefore, we associate a set of configurations or constraints, $\mathcal{C}_M$ on $\mathcal{H}$ with each $M \in \mathcal{M}$ (recall $\mathcal{M}$ is the set of vertices of $R$). We call $\mathcal{C}_M$ the configuration set associated with $M \in \mathcal{M}$. The collection of configuration sets is called a configuration space.

**Definition 3.** *Let $\mathcal{H}$ be a collection of chordal subgraphs of $G$, and $\{\mathcal{C}_M\}_{M\in\mathcal{M}}$ be a collection of sets (called configurations). Then $(\mathcal{H}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}})$ is a* configuration-space *if $\lambda_M : \mathcal{H} \to \mathcal{C}_M$ is a function that assigns to each $H \in \mathcal{H}$ a configuration for every $M \in \mathcal{M}$.*

A graph $H \in \mathcal{H}$ satisfies a constraint $c_M \in \mathcal{C}_M$ if $\lambda_M(H) = c_M$. Therefore, for every constraint $c_M \in \mathcal{C}_M$, the set of graphs $\{ H \in \mathcal{H} : \lambda_M(H) = c_M\}$ is a subset of $\mathcal{H}$ that satisfies the constraint $c_M$. Every $H \in \mathcal{H}$ must satisfy at least one constraint in $\mathcal{C}_M$ (namely $\lambda_M(H)$), though it is possible that $H$ satisfies more than one constraint in $\mathcal{C}_M$. When this is possible, we will require that there is a unique strongest constraint that $H$ satisfies. As an example, for a given $M \in \mathcal{M}$, we might index the configurations $\mathcal{C}_M$ by subsets of the edges of $M$, representing the missing edge. Therefore, the set $A \in \mathcal{C}_M$ might represent the constraint "$H$ is missing the edges in $A$". In this case if $A_1 \subseteq A_2$, then $H$ automatically satisfies $A_1$ if it satisfies $A_2$. We will also be interested in knowing if constraints $c_{M_1} \in \mathcal{C}_{M_1}$ and $c_{M_2} \in \mathcal{C}_{M_2}$ are compatible in the sense that there is some graph $H$ that satisfies both constraints. More formally,

**Definition 4.** *Suppose that $(\mathcal{H}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}})$ is a configuration-space. If $M_p$ and $M_d$ are vertices of a separator-clique tree $R$ and $c_p \in \mathcal{C}_{M_p}$ and $c_d \in \mathcal{C}_{M_d}$, then we say that $c_p$ and $c_d$ are* compatible *if there is a $H \in \mathcal{H}$ with $\lambda_{M_p}(H) = c_p$ and $\lambda_{M_d}(H) = c_d$.*

If we denote by $\mathcal{H}_{\{c_{M_1}, c_{M_2}, \ldots, c_{M_m}\}}$ the set $\{ H \in \mathcal{H} : \lambda_{M_i}(H) = c_{M_i}$ for $1 \le i \le m\}$, then $c_{M_p}$ and $c_{M_d}$ are compatible if and only if $\mathcal{H}_{\{c_{M_p}, c_{M_d}\}} = \mathcal{H}_{c_{M_p}} \cap \mathcal{H}_{c_{M_d}} \neq \phi$. The definitions we have given so far are too general to be of use algorithmically. We therefore place some restrictions on configuration spaces which will enable us to efficiently find optimal solutions.

**Definition 5.** *The configuration space $(\mathcal{H}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}})$ is* efficiently-decomposable *with respect to the objective function $f$ if all of the following conditions hold.*

1. **[Efficiency]** *$|\mathcal{C}_M|$ is polynomial in $n$ for each $M \in \mathcal{M}$, and for any given $H \in \mathcal{H}$, the function $\lambda_M(H)$ can be computed in polynomial time.*

2. **[Sufficiency]** *The set $\{\lambda_M(H)\}_{M\in\mathcal{M}}$ uniquely specifies $H \in \mathcal{H}$. Further, given $\{\lambda_M(H)\}_{M\in\mathcal{M}}$, we can reconstruct $H$ in polynomial (in $n$) time. Note that we do not require that every set of configurations $\{c_M\}_{M\in\mathcal{M}}$ yields a graph, but only that the graph be unique if it exists. So the set of graphs implied can be empty.*

3. **[Conditional separation]** *Suppose that $M_p, M_d$ and $M_o$ are vertices of the separator-clique tree with $M_d$ a child of $M_p$, and $M_o$ a non-descendent of $M_p$. Let $c_{M_d} \in \mathcal{C}_{M_d}$, $c_{M_p} \in \mathcal{C}_{M_p}$ and $c_{M_o} \in \mathcal{C}_{M_o}$. If $\mathcal{H}_{\{c_{M_p}, c_{M_o}\}} \neq \phi$ and $\mathcal{H}_{\{c_{M_p}, c_{M_d}\}} \neq \phi$, then $\mathcal{H}_{\{c_{M_p}, c_{M_o}, c_{M_d}\}} \neq \phi$.*

4. **[Compatibility]** *If $M_p$ is the parent of $M_d$, then for any configuration $c_p \in M_p$ and $c_d \in M_d$, we can determine if $c_p$ and $c_d$ are compatible in polynomial time.*

5. **[Computability]** *If $M$ is a leaf vertex (corresponding to a maximal clique of $G$), and $c_M \in \mathcal{C}_M$ is any configuration, then we can find a subgraph $H \in \mathcal{H}$ compatible with $c_M$ to minimize $f(H[M])$ in polynomial time.*

The basic goal is to represent elements of $\mathcal{H}$ by the set of constraints (configurations) that they satisfy. The *sufficiency* condition ensures that, given a complete set of compatible constraints $\{c_M\}_{M\in\mathcal{M}}$, there is at most one element of $\mathcal{H}$ that satisfies all the constraints. Therefore, a complete set of compatible configurations contains enough information to determine uniquely the element of $\mathcal{H}$. The *efficiency* condition ensures that for any $M \in \mathcal{M}$, there are not too many configurations/constraints, and so for example, all the configurations may reasonably be enumerated. It also ensures that we can represent an element of $H \in \mathcal{H}$ uniquely and efficiently as $\{\lambda_M(H)\}_{M\in\mathcal{M}}$. The *conditional separation* condition essentially says that if $H$ satisfies the configuration of a vertex $M_p$, then the configurations of the descendants and the non-descendants may be chosen separately.

Suppose that $ij \in \mathcal{M}$ is the root of $R$. Let $\mathsf{l}(ij)$ and $\mathsf{r}(ij)$ be the left and right children of $ij$, and $\mathcal{H}^{(i)}$ and $\mathcal{H}^{(j)}$ be the set of subgraphs of elements in $\mathcal{H}$ induced by $V^{(i)}$ and $V^{(j)}$ respectively. Further, let $\mathcal{M}^{(i)}$ and $\mathcal{M}^{(j)}$ be the vertex set of the separator (sub) trees $\mathsf{tl}(ij)$ and $\mathsf{tr}(ij)$. We then have the following result.

**Lemma 5.** *Suppose that $(\mathcal{H}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}})$ is a (efficiently-decomposable) configuration space for $G$. Then for $k \in \{i,j\}$, $(\mathcal{H}^{(k)}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}^{(k)}})$ is a (efficiently-decomposable) configuration space for $G^{(k)}$.*

**Algorithm 2:** Finding optimal set of configurations

```
Data: A graph $G$, A separator-clique tree $R$ for $G$, a efficiently-decomposable configuration space
      $(\mathcal{H}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}})$, and a decomposable function $f : \mathcal{H}^* \to \mathbb{R}$.
Result: An optimal collection of compatible configurations $\{c_M\}_{M\in\mathcal{M}}$.
for each vertex $M$ of $R$ in order of increasing height (bottom up) do
    for each configuration $c_M$ of $M$ do
        if $M$ is an interior vertex of $R$ corresponding to edge $ij$ of the clique tree then
            Let $M_l$ and $M_r$ be the left and right children of $M$;
            Pick configuration $c_l \in \mathcal{C}_{M_l}$ compatible with $c_M$ to minimize $\mathsf{table}[M_l, c_l]$;
            Pick configuration $c_r \in \mathcal{C}_{M_r}$ compatible with $c_M$ to minimize $\mathsf{table}[M_r, c_r]$;
            $\mathsf{table}[M, c_M] \leftarrow \mathsf{table}[M_l, c_l] + \mathsf{table}[M_r, c_r] + g(H^{(ij)})$;
        else
            // $M$ is a leaf vertex of $R$, and hence a maximal clique.
            Pick subgraph $H[M]$ of the clique $M$ satisfying $c_M$ to minimize $f(H[M])$ ;
            $\mathsf{table}[M, c_M] \leftarrow f(H[M])$;
```

Suppose that $R$ is a separator-clique tree for $G$, and $(\mathcal{H}, \{\mathcal{C}_M\}_{M\in\mathcal{M}}, \lambda_M)$ is an efficiently-decomposable configuration space. We will be interested in finding an $H \in \mathcal{H}$ for which $D(P_G \| P_H)$ is minimized. Suppose that $ij$ is the root of $R$. By Lemma 2, $D(P_G\|P_H) = D(P_{G^{(i)}}\|P_{H^{(i)}}) + D(P_{G^{(j)}}\|P_{H^{(j)}}) - D(P_{G[V_{ij}]}\|P_{H[V_{ij}]})$. By the sufficiency condition, finding the optimal $H \in \mathcal{H}$ is equivalent to finding an optimal set of (compatible) configurations $\{\lambda_M(H)\}_{M\in\mathcal{M}}$. If we could find the optimal set of (compatible) configurations for the left and right trees, then by the conditional separation condition, it suffices to check every possible configuration for the root, and then pick the best left and right configurations that are compatible with this. This observation immediately leads to the dynamic programming algorithm shown as Algorithm 2.

**Theorem 6.** *Algorithm 2 computes the optimal set of configurations, in polynomial time.*

*Proof.* We will actually show a stronger result. We show that if $M_r$ is the root of the separator-clique tree, then for each configuration $c_r \in \mathcal{C}_{M_r}$, Algorithm 2 will compute the optimal element $H \in \mathcal{H}$ that is compatible with $c_r$ in polynomial time, and $\mathsf{table}[M_r, c_r] = f(H)$. Since $|\mathcal{C}_{M_r}|$ is polynomial in $n$, this implies we can enumerate all the configurations and pick the optimal solution in polynomial time.

We show this result by induction on the height of $R$, the separator-clique tree for $G$. If the height of $R$ is 1, then $R$ has just one vertex, and so $G$ consists of a single clique. Therefore by the computabilty property, for every configuration $c_r \in \mathcal{M}_r$, the optimal subgraph $H \in \mathcal{H}$ compatible with $c_r$ can be determined in polynomial time. Now, suppose that the result holds for all graphs with separator-clique tree of height at most $h$. Suppose that $G$ is a graph with separator-clique tree $R$ of height $h+1$. Let $ij$ be the root of $R$, with children $M_l$ and $M_r$. Then by Lemma 5, for any $k \in \{i,j\}$, $(\mathcal{H}^{(k)}, \{\mathcal{C}_M, \lambda_M\}_{M\in\mathcal{M}^{(k)}})$ is a (efficiently-decomposable) configuration space for $G^{(k)}$. Further, $R^{(k)}$ is of height at most $h$. Therefore, by the inductive hypothesis, $\mathsf{table}[M_l, c_l]$ and $\mathsf{table}[M_r, c_r]$ contain the optimal values of the objective functions compatible with $c_l$ and $c_r$ respectively. Therefore, by the decomposability of $f$, and by the conditional separation and compatibilty properties, $\mathsf{table}[M, c]$ can be computed in polynomial time for each $c \in \mathcal{C}_M$. Therefore, the algorithm computes the optimal value of $f(H)$ in polynomial time. □

We now give some specific examples of efficiently-decomposable configuration spaces for some classes of chordal subgraphs of $G$.

## 5.1 Optimal chordal subgraphs with $|E(G)| - d$ edges

Suppose that we are interested in a (chordal) subgraph of $G$ that contains $d$ fewer edges that $G$ does. That is, we want to find an optimal subgraph $H \subset G$ such that $|E(H)| = |E(G)| - d$. Of course, optimality is with respect to KL-divergence. Two possibilities for finding such an optimal subgraph are

1. Use the procedure described in [4]. This is a greedy procedure which works in $d$ steps by deleting an edge at each step. At each state, the edge is picked from the set of edges whose deletion leaves a triangulated graph. Then the edge which causes the least increase in KL-divergence is picked at each stage.

2. For each possible subset $A$ of $E(G)$ of size $d$, whose deletion leaves a triangulated graph, compute the KL divergence using the formula above, and then pick the optimal one. Since there are at most $\binom{|E(G)|}{d}$ such sets, this can be done in polynomial time.

The first greedy algorithm is not guaranteed to yield the optimal solution. The second takes time that is $O(n^{2d})$. Now, let us solve this problem using the framework we've described.

Let $\mathcal{H}$ be the set of subgraphs of $G$ which may be obtained by deletion of at most $d$ edges. For each $M = ij \in \mathcal{M}$, let $\mathcal{C}_M = \left\{ (l, r, c, A) : l + r - c = d, A \in \binom{E(G[M])}{c} \right\}$. The constraint represented by $(l, r, c, A)$ is this : $A$ is a set of $c$ edges of $G[M]$ that are missing in $H$, $l$ edges are missing from the left subgraph, and $r$ edges are missing from the right subgraph. $c$ represents the double count, and so is subtracted from the total. If $k$ is the size of the largest clique, then the total number of such configurations is bounded by $d \cdot \binom{\binom{k}{2}}{d} \in O(k^{2d})$. This is an efficiently decomposable configuration space, and the dynamic programming formulation that we described will take only $O(n \cdot k^{2d})$ time (for fixed $d$). This will lead to a substantial reduction in complexity on most graphs (exponentially less if the size of the largest clique is, say, at most half the size of the graph).

## 5.2 Optimal sub $(k-1)$-trees of $k$-trees

**Lemma 7.** *Suppose that $G$ is a $k$-tree, and $S = V_{ij}$ a minimal separator in $G$ corresponding to the edge $ij \in T$. If $H \subseteq G$ is a $(k-1)$-tree then there is a $x \in S$ such that $S \setminus \{x\}$ is complete in $H$. Further, one of the following cases must hold.*

1. *In $H$, $x$ is connected to one or more vertices in $V^{(i)} \setminus S$ or to one or more vertices in $V^{(j)} \setminus S$ but not to vertices in both.*

2. *There is a $y \in S$ such that $H[S]$ is only missing the edge $\{x, y\}$.*

*Proof.* Since the removal of $S$ disconnects $G$, the removal of $S$ must also disconnect $H$. Therefore, $S$ must contain a minimal separator of $H$. Since $H$ is a $(k-1)$-tree, all minimal separators of $H$ must contain $k-1$ vertices, and so there is some $x \in S$ such that $S_x = S \setminus \{x\}$ is a minimal separator. Now, suppose that $x$ is connected (in $H$) to a vertex $y_i \in V^{(i)} \setminus S$ and to a vertex $y_j \in V^{(j)} \setminus S$. In $H$, $S$ is a (non-minimal) $y_i/y_j$ separator, and so must contain a complete set $S_1$ which is a minimal vertex separator. Since $x$ is connected to both $y_i$ and $y_j$, we must have $x \in S_1$ (see [8], Lemma 2.1.3). Since $H$ is a $(k-1)$-tree, $|S_1| = k-1$. Let $y$ be the unique element in $S \setminus S_1$. Then $H[S]$ contains all edges except possibly $\{x, y\}$. Suppose that $H[S]$ also contained the edge $\{x, y\}$. Then $S$ is a minimal separator of size $k$, which contradicts the fact that $H$ is a $(k-1)$-tree. $\square$

Therefore, we associate the following constraints with each separator $S = V_{ij}$ for $G$ : There is some $x \in S$ such that $x$ is either (a) not connected to any vertex in $(V^{(i)} \cup V^{(j)})$ or (b) not connected to any vertex in $V^{(j)} \setminus S$ or (c) not connected to any vertex in $V^{(i)} \setminus S$ or (d) $H[S]$ is missing edge $\{x, y\}$ for some $y \in S$. It is clear that the constraints are not mutually exclusive in the sense that there can be $H \in \mathcal{H}$ that satisfy more than one of these constraints. However that is perfectly acceptable. By the lemma, $H$ must satisfy at least one of these constraints. If $|S| = k$, then there are $3k + \binom{k}{2} = O(k^2)$ possible configurations. Since $k \leq n$, there are only $O(n^2)$ possible configurations for each element of $\mathcal{M}$. Therefore, the optimal $(k-1)$-tree can be found using Algorithm 1, and in this case, the complexity of Algorithm 1 is $O(n(k+1)^2)$. This procedure can be generalized to find the optimal sub $(k-d)$- tree for any fixed

| | k = 10 | k=15 | k=20 | k=30 | k=40 | k=50 |
|---|---|---|---|---|---|---|
| n=100 | 1.544 | 3.698 | 7.829 | 23.22 | 61.10 | 114.7 |
| n=200 | 3.563 | 8.156 | 20.40 | 57.38 | 159.6 | 338.6 |
| n=500 | 7.747 | 21.09 | 46.45 | 156.7 | 421.2 | 949.2 |
| n=1000 | 15.996 | 43.36 | 96.26 | 330.7 | 955.0 | 2049 |

Table 1: Time taken to find optimal $(k-1)$-subtrees of $n$ vertex $k$-trees (in seconds)

$d$. However, the number of configurations grows exponentially with $d$ (though it is still polynomial in $n$). Therefore for small, fixed values of $d$, we can compute the optimal sub $(k-d)$-tree of $G$. While we can compute $(k-d)$-trees of $G$ by first going from a $k$ tree to a $(k-1)$ tree, then from a $(k-1)$-tree to a $(k-2)$-tree, and so on in a greedy fashion, this will not be optimal in general. However, this might be a good algorithm to try when $d$ is large.

### 5.3 Optimal complexity-distortion tradeoffs

We can find optimal complexity-distortion tradeoffs (by taking $\mathcal{H}$ to be the set of all chordal subgraphs of $G$) if the complexity measure also factorizes over trees like the KL-divergence. For example, the sum of the sizes of the cliques, or the sum of the sizes of the state spaces of the cliques also factor. If $|V_{ij}| = k \in O(\sqrt{\log n})$, then $2^{k^2}$ is polynomial in $n$, and so we can compute the optimal complexity-distortion tradeoffs as long as the size of the largest clique is $O(\sqrt{\log n})$.

## 6 Results

The algorithm shown was implemented to find optimal $(k-1)$-subtrees of $k$-trees using C++/STL/MATLAB. The algorithm computes the optimal subgraph, and in Table 1, we present the times taken on random graphs, with Gaussian probability distributions. It can be seen that the algorithm runs fairly quickly, and so is practical (no hidden constants) even when $n$ and $k$ are relatively large.

## 7 Acknowledgements

This work was supported by NSF grant IIS-0093430 and an Intel Corporation Grant.

## References

[1] F. Gavril, “Algorithms on clique separable graphs”, Discrete Mathematics v. 9 (1977), pp. 159–165.

[2] R. E. Tarjan. “Decomposition by Clique Separators”, Discrete Mathematics, v. 55 (1985), pp. 221–232

[3] F. Jensen and S. K. Anderson, “Approximation in Bayesian belief universes for knowledge based systems”, Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence, pp. 162–169, 1990

[4] U. Kjaerulff. “Reduction of computational complexity in Bayesian networks through removal of weak dependencies”, Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence, pp. 374–382, 1994

[5] C. W. Ho and R. C. T. Lee. “Counting clique trees and computing perfect elimination schemes in parallel”, Inf. Proc. Lett. 31, (1989), pp. 61-68.

[6] T. A. McKee and F. R. McMorris. “Topics in Intersection Graph Theory”, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[7] A. Brandstädt, V. B. Lee and J. P. Spinrad. “Graph Classes : A Survey”, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[8] T. Kloks, “Treewidth: Computations and Approximations”, Springer-Verlag, 1994