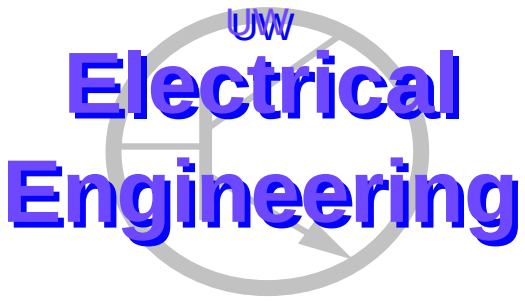
Learning Graphical Models over partial k-trees

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



UWEE Technical Report
Number UWEETR-2004-0009
March 2004

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Learning Graphical Models over partial k-trees

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

University of Washington, Dept. of EE, UWEETR-2004-0009
March 2004

Abstract

We show that the class of strongly connected graphical models with tree-width at most k can be properly efficiently PAC-learnt with respect to the Kullback-Leibler Divergence. Previous approaches to this problem, such as those of Chow ([1]), and Hoffgen ([7]) have shown that this class is PAC-learnable (though not necessarily efficiently unless $k = 1$) by reducing to a NP-complete combinatorial optimization problem. Unless P=NP, these approaches will take exponential amounts of time. Our approach differs significantly from these, in that it first attempts to find approximate conditional independencies by solving (polynomially many) submodular optimization problems, and then using a dynamic programming formulation to combine the approximate conditional independence information to derive a graphical model with underlying graph of the tree-width specified. This gives us an efficient (polynomial time) PAC-learning algorithm which requires only polynomial number of samples of the true distribution, and only polynomial running time.

1 Introduction and Previous work

A graphical model is a pair (P, G), where P is a probability distribution on the vertices of the graph G, such that an absence of an edge implies conditional independence (a more precise definition is given in the next section). In such a case P is said to factorize over G. In general, there is no unique graph over which a given probability distribution factorizes. In fact, all probability distributions factorize over the complete graph. Therefore, if one is given a probabilty distribution, one seeks to find a graph G with as few edges as possible over which the given distribution factorizes, in polynomial time. As stated, there is no solution to this problem unless P=NP ([10]). Therefore, we look at the following variant of this problem. Given a class of graphs $\mathcal{G}$, where it is know that the distribution P factorizes over at least one graph $G \in \mathcal{G}$, find a graph in $\mathcal{G}$ over which P factorizes. This problem is not always easy, and its complexity depends on the class $\mathcal{G}$. Much research has been directed at finding classes $\mathcal{G}$ which can be learnt in polynomial time. There is a plethora of negative results ([8, 10, 13, 15]) showing that learning various classes of graphs, including paths and polytrees, is NP-complete. So far the only positive result in this area (to the best of our knowledge) are the results of ([1, 7]) who show that the class of trees can be properly efficiently learnt. While trees are an important class of distributions, they can only represent very sparse dependencies, and there is much interest in learning other model families in which richer dependence structures are possible.

Prior attempts at learning the model family have relied on reducing the problem to a combinatorial optimization problem. When the combinatorial optimization problem is easy (such as finding minimum/maximum spanning tree), then the learning problem becomes easy. The reverse reduction (reducing an arbitrary instance of a NP-complete problem to a learning problem) is used to demonstrate hardness of learning problem. In this report, we use a very different approach, and show the learnability of a subclass of bounded tree-width networks by first discovering the conditional independencies in the model by solving a polynomial number of submodular optimization problems. Each one of these problems can be solved in polynomial time using an algorithm developed by Queyranne ([9]). Then we give a dynamic programming algorithm that will patch these dependencies together to create a graph of the tree-width specified. While we restrict ourselves to showing the learnability of k-trees, this approach could conceivably be used for other subclasses of graphical models as well.

2 Preliminaries and Notation

A graph $G = (V, E)$ is said to be chordal if every cycle of length at least 4 has a chord. A separator $S \subseteq V$ of G is a set of vertices whose removal disconnects the graph. Chordal graphs can also be characterized as graphs in which all minimal separators are cliques. A connected graph G is said to be a k-tree if it is (isomorphic) to K_{k+1} or if it has at least $k+2$ vertices, and every minimal separator in G is a clique of size k. G is called a partial k-tree if it is a subgraph of a k-tree. A tree-decomposition of G ([6]) is a pair $(T = (I, F), \{V_i\}_{i \in I})$, where T is a tree, $V(G) = \cup_{i \in I} V_i$ and

1. If uv is an edge in $E(G)$, then there is a $i \in I$ such that $u, v \in V_i$, and
2. If $i, j, k \in I$ are such that j is on the path from i to k in T, then $V_i \cap V_k \subseteq V_j$.

The tree-width of this tree-decomposition is $\max_{i \in I} |V_i| - 1$. If G is k-tree, then G is a chordal graph which has a tree-decomposition with width k. Therefore, a partial k-tree is a chordal graph which has a tree-decomposition of width k. For any edge $e = ij \in F$, we let $V_e = V_{ij} = V_i \cap V_j$. Note that $V_e \subseteq V$ is a separator in G for every $e \in F$, and in fact, if G is chordal, then these are the unique minimal separators of G (see [5]). If T_s is a subtree of T, then $(T_s, \{V_i\}_{i \in V(T)})$ is a tree-decomposition for the induced subgraph $G[\cup_{i \in V(T)} V_i]$. Therefore, we may associate an induced subgraph $G[T_s] = G[\cup_{i \in V(T)} V_i]$ with each subtree T_s of T. If S is a minimal separator of a chordal graph G, and $C_1, C_2, \ldots, C_m$ are the connected components of $G[V \setminus S]$, then we say that $\{G_i\}_{1 \leq i \leq m}$ are the fragments of S where $G_i = G[C_i \cup S]$. We also say that $\{G_1, \ldots, G_m\}$ is a fragmentation of G with respect to S.

Let P be a probability distribution over random variables $\{X_v\}_{v \in V}$. For any $A \subseteq V$, we let P_A be the (marginalized) probability distribution of the random vector $(X_a)_{a \in A}$. We say that the probability distribution factorizes according to the chordal graph $G = (V, E)$ if for every separator $S \subseteq V$ which separates the graph into components A and B, the marginal distributions $P_{A \cup S}$ and $P_{B \cup S}$ factorize with respect to $G[A \cup S]$ and $G[B \cup S]$ respectively. Let us denote by $H_P(A)$ the binary entropy $H(\{X_v\}_{v \in A})$ with respect to the probability distribution P, and by $I_P(A; B|S)$ the mutual information $I(\{X_v\}_{v \in A}; \{X_v\}_{v \in B} \,|\, \{X_v\}_{v \in S})$ with respect to P. If P factorizes over G, and S is a minimal separator of G with fragments $G_1, G_2, \ldots, G_m$, then $I_P(V(G_i); V(G_j)|S) = 0$ for all $i \neq j$. S is called an α-strong separator if we cannot partition the vertices in C_i into $C_i = C_{i1} \cup C_{i2}$ satisfying $I_P(C_{i1}; C_{i2}|S) \leq \alpha$. We say that the graphical model (P, G) is α-strongly connected if every minimal separator S of the chordal graph G is an α-strong separator. We say that (P, G) is strongly connected if there is some $\alpha > 0$ such that every minimal separator is an α-strong separator.

A class of distributions $\mathcal{T}$ is properly efficiently learnable (with respect to the KL-divergence criteria) if there exists a probabilistic algorithm which for every $P \in \mathcal{T}$, and for every $\epsilon, \delta > 0$, finds a $\tilde{P} \in \mathcal{T}$ such that $D(P \| \tilde{P}) < \epsilon$ with probability at least $1 - \delta$ in time polynomial in n, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ using only polynomial number of samples from the true distribution P. We will assume the existence of an algorithm that can estimate the entropies of an arbitrary subset of the random variables with precision ϵ and confidence $1 - \delta$ using $f(n, \epsilon, \delta)$ samples, where $f(n, \epsilon, \delta)$ is polynomial in n, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. The exact algorithm might depend on the nature of the distribution (i.e., depending on whether the random variables are discrete, continuous, Gaussian etc.). An application of Hoeffding's inequality shows that this is possible for discrete distributions (alternatively see [7]). It is also possible to do this for continuous/mixed distributions by discretizing, though much more efficient algorithms exist for distributions such as Gaussians. It is clear that if each query can be computed in polynomial time, with at most polynomial number of samples, then any algorithm which issues at most a polynomial number of such queries, and runs in polynomial time other than these queries is in fact a polynomial time algorithm.

3 Conditional Independences and Partitions

Consider a graphical model (P, G). If $S \subseteq V(G)$ is a (minimal) separator in G, the fragments of S are conditionally independent given S. In particular, if $C_1, C_2, \ldots, C_m$ are the connected components of $G[V \setminus S]$ corresponding to fragments $G_1, G_2, \ldots, G_m$ of S, then we must have $V(G_i) \perp\!\!\!\perp V(G_j)|S$, whenever $i \neq j$. Let $\pi_S = \{V(C_1), V(C_2), \ldots, V(C_m)\}$. Then π_S is a partition of $V_S = V \setminus S$. We will denote by the pair (S, π_S) the conditional independence statement $V(G_i) \perp\!\!\!\perp V(G_j)|S$ for all $i \neq j$.

While it may be more natural to think of the edges in G representing dependences between the random variables, we really should be concentrating on the absence of edges, which represent the conditional independences of the probability model. Therefore, the graphical model may be thought of as as encoding the collection $\{(S_i, \pi_{S_i})\}_{i=1}^{N}$, where each S_i is a minimal separator with associated partition π_{S_i} of $V_{S_i} = V \setminus S_i$. Now, an obvious question that

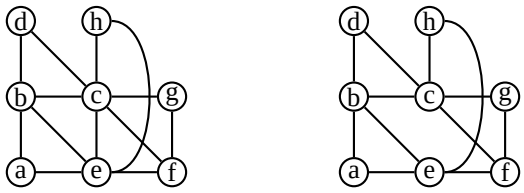


Figure 1: Two graphs with identical separators.

Separator	Partition
$\{c, e\}$	$\{\{a, b, d\}, \{h\}, \{g, f\}\}$
$\{b, c\}$	$\{\{d\}, \{a, e, g, h, f\}\}$
$\{c, f\}$	$\{\{g\}, \{a, b, d, e, h\}\}$

Table 1: The separators for the graphs in Figure 1.

arises is whether these two representations are equivalent. That is, given one representation, can one reconstruct the other representation ? Clearly, if we are given the graph G, then it is an easy matter to determine all the minimal separators and their associated partitions. In fact, if G has tree-width no more than k, one can in time $n^{O(k)}$ determine all the minimal separators and associated partitions. However, given the separators, there is not necessarily a unique graph that induces these minimal separators. For example, both graphs shown in Figure 1 have identical separators, which are shown in Table 1.

Note that both the graphs shown in Figure 1 have the same tree-width, and so even when the tree-width is fixed, the separator representation is not unique. It would seem therefore that this representation is not a useful way of specifying graphical models. However, we should note that in a graphical model (P, G), while the probability distribution P is really important and central, the graph G is just a useful representation. Very often, there is no one true graph that can be determined from the probability model. In fact one might argue that the only thing that the probability distribution specifies is the conditional independence statements (which translate into separators, and their associated partitions). The actual graph itself is just a representational artifact. In many (most ?) applications, if P factorizes over two graphs G_1 and G_2, then there is little reason to prefer G_1 to G_2 unless the tree-widths of the two graphs differ. Since the efficiency of various algorithms, including inference procedures, are determined by the tree-width of the graph, it makes sense to pick a graph with as small a tree-width as possible. Now, for applications we are interested in, the graphs are required to be chordal. In this case, finding a graph with a given tree-width is equivalent to finding a tree-decomposition of a given width. In fact the tree-decomposition is an ideal representation of a (chordal) graph because it emphasizes the separators instead of the edges, fitting in well with our assertion that the only real objects of interest are the conditional independences. Therefore, given a set of separators/partitions $\{(S_i, \pi_{S_i})\}_{i=1}^{N}$ that the probability model satisfies, we seek a tree-decomposition that is, in some sense, compatible with this set of separators/partitions. What we mean by compatibility is intuitively clear : We want the the set of conditional independences implied by the tree-decomposition to be a subset of the set of separators/partitions that we are given. More formally, we note that if $ij \in F$ is an edge in the tree-decomposition, then the removal of the edge induces two subtrees, T_a and T_b, such that $V(T_a) \cap V(T_b) = V_i \cap V_j$. So, $S = V_i \cap V_j$ is a separator in any graph corresponding to the given tree-decomposition, and π_S refines the partition $\{V(T_a) \setminus S, V(T_b) \setminus S\}$. We say that P factorizes over this tree-decomposition if for ever edge $ij \in F$, the vertex sets of the two subtrees are conditionally independent given $V_i \cap V_j$. Therefore, we want the set of conditional independences implied by the tree-decomposition to be a subset of the allowable conditional independencies. Note that it is perfectly acceptable for the resulting tree-decomposition to yield fewer conditional independencies, as long as the tree-width constraint is not violated. Conditional independences not present in the original model however are unacceptable. It will be shown in Section 5 that using the tree-decomposition representation simplifies many quantities we are interested in as well. It will also be shown that when in fact the "true" conditional independences of the model are not known, but only "approximate" conditional independences are known, then we can still find a tree-decomposition that yields a graphical model $(\tilde{P}, \tilde{G})$ that approximates (P, G). The quality

of the approximation is in terms of KL-divergence criterion, $D(P\|\tilde{P})$. In the next section, we give more details about this, define "approximate" conditional independence, and give a procedure that generates a set of "approximate" conditional independences.

4 Mutual Information and Submodularity

A set function $f : 2^V \to \mathbb{R}^+$ is called submodular if for any $A, B \subseteq V$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. f is said to be symmetric if $f(A) = f(V \setminus A)$. Submodular functions may be thought of as the discrete analog of convex functions, and have several similar properties. For example, there are polynomial time algorithms to minimize submodular functions (see [3]). Our interest in submodular functions stems from the fact that the mutual information function is a submodular function.

Lemma 1. *Let P be any probability distribution, and $S \subseteq V$ be a fixed set. Then $H_P(\cdot|S) : 2^V \to \mathbb{R}$ is a submodular function.*

Proof. $0 \leq I_P(A; B|S) = H_P(A|S) + H_P(B|S) - H_P(A \cup B|S) - H_P(A \cap B|S)$. □

Proposition 2. *Let P be an arbitrary distribution on the random variables $\{X_v\}_{v \in V}$. Let $F : 2^V \to \mathbb{R}^+$ be given by $F(A) = I_P(A; V \setminus A)$. Then F is symmetric and submodular.*

Proof. We can write $F(A) = H_P(A) + H_P(V \setminus A) - H_P(V)$. Therefore,

$$\begin{aligned}
F(A) + F(B) &= H_P(A) + H_P(V \setminus A) - H_P(V) + H_P(B) + H_P(V \setminus B) - H_P(V) \\
&= [H_P(A) + H_P(B)] + [H_P(V \setminus A) + H_P(V \setminus B)] - 2H_P(V) \\
&\geq [H_P(A \cap B) + H_P(A \cup B)] + [H_P(V \setminus A \cap B) + H_P(V \setminus A \cup B)] - 2H_P(V) \\
&= [H_P(A \cap B) + H_P(V \setminus A \cap B) - H_P(V)] + [H_P(A \cup B) + H_P(V \setminus A \cup B) - H_P(V)] \\
&= F(A \cap B) + F(A \cup B)
\end{aligned}$$

Hence $F(\cdot)$ is submodular. The symmetry of F is immediate from its definition. □

We will be investigating conditional independence relationships between random variables. Now, if a set of random variables $\{X_a\}_{a \in A}$ is conditionally independent of the set of random variables $\{X_b\}_{b \in B}$, given the set $\{X_s\}_{s \in S}$ according to the probability distribution P, then $I_P(A; B|S) = 0$. Note that in this definition, we have not required that A and B be disjoint (or $A \cap B = \phi$). However since $I_P(A \cup S; B \cup S|S) = I_P(A; B|S)$ it will be convenient for our application to let $A \cap B = S$. Let $V_S = V \setminus S$. The following corollary is immediate.

Corollary 3. *For any probability distribution P, the function $F_{P,S} : 2^{V_S} \to \mathbb{R}^+$, given by $F_{P,S}(A) = I_P(A; V_S \setminus A|S)$, is symmetric and submodular.*

One reason for the importance of submodular functions is the existence of efficient algorithms to minimize submodular set functions (see [2]). If $f : 2^V \to \mathbb{R}^+$ is both symmetric and submodular, then there exist combinatorial algorithms to compute this minimum. For example, Queyranne's Algorithm, described in [9], will return a proper subset $A \subseteq V$ such that $A \in \arg\min_{B \in 2^V \setminus \{V, \phi\}} f(B)$. Let us denote this algorithm by QA. QA takes as input a f-value oracle, and runs in time $O(|V_S|^3)$, using at most $|V_S|^3$ oracle calls. In particular, since $F_{P,S} : 2^{V_S} \to \mathbb{R}^+$ is symmetric and submodular, we can (quickly) find a set A such that $F_{P,S}(A)$ is minimized. If $S \subseteq V$, then we say that S is a ϵ-separator of V if there is a partition $A \cup B$ of V_S such that $I_P(A; B|S) \leq \epsilon$. In this case, we call $\{A, B\}$ an ϵ-partition for (V_S, S, P). Clearly, we can use QA, and a $F_{P,S}$-value oracle, to determine if there are any ϵ-partitions for (V_S, S, P).

When we are trying to estimate or learn a probability distribution P by sampling, we do not have a $F_{P,S}$-value oracle. However, sampling will let us estimate $F_{P,S}$. So, suppose that we had a $\widetilde{F_{P,S}}$-value oracle, which satisfies $\left|\widetilde{F_{P,S}}(A) - F_{P,S}(A)\right| \leq \epsilon_1$ for every $A \subseteq V_S$. One problem is that $\widetilde{F_{P,S}}(A)$ need not be submodular, and hence QA need not return the minimum value of $\widetilde{F_{P,S}}(A)$. However, we do have the following result.

Lemma 4. *Suppose that $F_{P,S} : 2^{V_S} \to \mathbb{R}^+$ is a symmetric submodular function, and $\widetilde{F_{P,S}} : 2^{V_S} \to \mathbb{R}^+$ is another function that is not necessarily submodular, but satisfies $\left|\widetilde{F_{P,S}}(A) - F_{P,S}(A)\right| \leq \epsilon_1$ for every $A \subseteq V_S$. Then* QA *will return a non-empty proper subset $\tilde{A} \subset V_S$ such that for every non-empty proper subset $A \subset V_S$,*

$$\widetilde{F_{P,S}}(\tilde{A}) - \widetilde{F_{P,S}}(A) \leq |V_S| \cdot \epsilon_1 \leq |V| \cdot \epsilon_1$$

Proof. See Appendix. □

So, if $\tilde{A}$ is the value returned by QA, then $\tilde{A}$ is an approximate minimizer for $\widetilde{F_{P,S}}(\cdot)$. In fact it is also an approximate minimizer for $F_{P,S}(\cdot)$. To see this, let $A \in \arg\min_{D \in 2^{V_S} \setminus \{V_S, \phi\}} F_{P,S}(D)$. Then

$$\begin{aligned}
0 &\leq F_{P,S}(\tilde{A}) - F_{P,S}(A) \\
&\leq (\widetilde{F_{P,S}}(\tilde{A}) + \epsilon_1) - (\widetilde{F_{P,S}}(A) - \epsilon_1) \\
&= 2\epsilon_1 + \widetilde{F_{P,S}}(\tilde{A}) - \widetilde{F_{P,S}}(A) \\
&\leq 2\epsilon_1 + |V| \cdot \epsilon_1 \\
&= (|V| + 2)\, \epsilon_1
\end{aligned}$$

Therefore, if we run QA using the $\widetilde{F_{Q,S}}$-value oracle, we will get a set $\tilde{A}$ that is $(|V|+2)\epsilon_1$-close to optimal for $F_{P,S}$. In particular, for all proper subsets $B \subseteq V_S$, $F_{P,S}(B) \geq F_{P,S}(\tilde{A}) - (|V|+2)\epsilon_1$. It is clear that this holds whenever $\left|F_{P,S}(A) - \widetilde{F_{P,S}}(A)\right| \leq \epsilon_1$ for every query issued by the run of QA. It is clear that if each oracle query $\widetilde{F_{P,S}}(A)$ can be computed in polynomial time, and using only polynomially many samples, with precision ϵ_1 and confidence at least $1 - \delta_1$. Therefore, we have $\left|\widetilde{F_{P,S}}(A) - F_{P,S}(A)\right| \leq \epsilon_1$ for all the A's queried by the run of QA with confidence at least $1 - |V_S|^3 \delta_1$ by the union bound[1]. We summarize this discussion in the proposition below

Proposition 5. *Suppose that $\widetilde{F_{P,S}}$ represents an approximation to $F_{P,S}$ which satisfies $\left|\widetilde{F_{P,S}}(A) - F_{P,S}(A)\right| \leq \epsilon_1$ with confidence at least $1 - \delta_1$ for any (particular) $A \subseteq V_S$. Then running* QA *with a $\widetilde{F_{P,S}}$-value oracle results in a solution $\tilde{A} \subseteq V_S$ which is within $(|V|+2)\epsilon_1$ close to optimal (for $F_{P,S}$) with confidence at least $1 - |V_S|^3 \delta_1$. In particular, if $\widetilde{F_{P,S}}(\tilde{A}) \geq (|V|+2)\epsilon_1 + \epsilon_2$, then there is no ϵ_2-partition for (V_S, S, P) (with confidence at least $1 - |V_S|^3 \delta_1$.*

The function $F_{P,S}(A) = I_P(A; V_S \setminus A|S)$, represents the mutual information between the random variables indexed by A and the remaining random variables once S is given. We will call $I_{\tilde{P}}(\cdot|S)$ or $\widetilde{F_{P,S}}$ the sample or the approximate mutual information. Suppose that $A \cup B = V_S$. If $I_{\tilde{P}}(A; B|S) \leq \epsilon$, we will say that $\{A, B\}$ is an ϵ-partition for $(V_S, S, \tilde{P})$.

If we think of a graphical model representing P as a collection of conditional independencies, then learning the graphical model is equivalent to learning these conditional independencies. If $A \perp\!\!\!\perp B|S$, then $I_P(A; B|S) = 0$. This of course means that $\{A, B\}$ is an 0-partition for (V_S, S, P). Hence it is a ϵ_1-partition for $(V_S, S, \tilde{P})$ with high confidence. We should note however, that the converse is not necessarily true. If $\{A, B\}$ is a ϵ_1-partition for $(V_S, S, \tilde{P})$, then it need not be a 0-partition for (V_S, S, P). However, it is a $(2\epsilon_1)$-partition for (V_S, S, P) and so for sufficiently small ϵ_1, A and B are "almost" independent given S. In fact, as we will show in a later section, we can always find almost independent partitions so that the total KL divergence between the resulting distribution, and the original distribution remains small.

If G is a graphical model representing the probability distribution P, and $A \perp\!\!\!\perp B|S$, this implies that $G[V \setminus S]$ is a disconnected graph, and A and B are the union of components of G. For example, in the graph in Figure 2, if $S = \{c, e\}$, then the removal of S disconnects the graph into the parts $\{a, b, d\}$ and $\{g, f, h\}$. Note that $\{g, f, h\}$ is not connected in the residual graph, but is the union of $\{g, f\}$ and $\{h\}$. It helps to consider the components themselves, rather than the union of components, and so, we extend our notion of almost conditional independence to partitions of V_S of the form $\pi = \{A_1, A_2, \ldots, A_m\}$, where $m \geq 2$. Since π is a partition of V_S, we require $A_i \cap A_j = \phi$

[1] If each A_i occurs with confidence $1 - p_i$, then A_i does not occur with probability p_i. Therefore, the probability that at least one of the A_i does not occur is at most $\sum p_i$. Hence all the A_i occur with probability at least $1 - \sum p_i$, hence all of the A_i occur with confidence $1 - \sum p_i$.

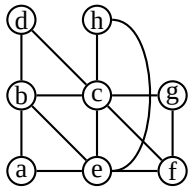


Figure 2: $S = \{c, e\}, \pi_S = \{\{g, f\}, \{a, b, d\}, \{h\}\}$

when $1 \leq i \neq j \leq m$, and $\cup_{i=1}^{m} A_i = V_S$. We call π an ϵ-partition for (V_S, S, P) if $I_P(A_i; A_j|S) \leq \epsilon$ for all $1 \leq i \neq j \leq m$. Now, it is clear that if $A \subseteq V_S$, and π is an ϵ-partition for (V_S, S, P), then $\pi^A = \{A_i \cap A : A_i \in \pi\}$ is a partition for A. Now, since $I_P(A_i \cap A; A_j \cap A|S) \leq I_P(A_i; A_j|S)$, it also follows that π^A is an ϵ-partition for (A_S, S, P). However, we also have the following result, which might be thought of as a partial converse.

Lemma 6. *Suppose that π is an ϵ_2-partition for $(V_S, S, \tilde{P})$, and $A \in \pi$. If ϕ is an ϵ_3-partition for $(A, S, \tilde{P})$, then $\psi = (\pi \setminus \{A\}) \cup \phi$ is a $\max(\epsilon_2, \epsilon_3)$-partition for $(V_S, S, \tilde{P})$.*

Proof. It is clear that ψ is in fact a partition for V_S. Let $\epsilon = \max(\epsilon_2, \epsilon_3)$. Then if $C, D \in \psi$, we consider 3 cases. First, if $C, D \in \pi$, then $I_{\tilde{P}}(C; D|S) \leq \epsilon_2 \leq \epsilon$. If $C, D \in \psi$, then $I_{\tilde{P}}(C; D|S) \leq \epsilon_3 \leq \epsilon$. If neither of these cases occur, we may assume (by symmetry) that $C \in \psi$ and $D \in \pi$. Then $I_{\tilde{P}}(C; D|S) \leq I_{\tilde{P}}(C; A|S) \leq \epsilon_2 \leq \epsilon$. □

Lemma 6 suggests a possible strategy for finding the almost independent components, for a given fixed separator S. Start with some partition π of V_S into nearly independent parts, and then iteratively attempt to partition the partitions, till the information loss is too large. This is shown in Algorithm 1. For any $S \subseteq V$ and $\epsilon > 0$, the algorithm shown

$\pi_S^0 \leftarrow \{V_S\}$; $i \leftarrow 0$;
while $\exists X_i \in \pi_S^i$ *such that* $\{A_i, B_i\}$ *is an* ϵ*-partition of* $(X_i, S, \tilde{P})$ **do**
  $\pi_S^{i+1} \leftarrow \left(\pi_S^i \setminus \{X_i\}\right) \cup \{A_i, B_i\}$; $i \leftarrow i + 1$;
end
$\pi_S(\epsilon) \leftarrow \pi_S^i$

Algorithm 1: Algorithm to compute ϵ-partition $\pi_S(\epsilon)$ (for any given ϵ and S)

computes a partition $\pi_S(\epsilon)$ of V_S. We say that a partition ψ is a refinement of π if each element of π can be written as the union of elements of ψ. Therefore, in Algorithm 1, the partition π_S^i refines π_S^j for all $j \leq i$. We then have the following result.

Lemma 7. *The partition $\pi_S(\cdot)$ produced by the above algorithm satisfies the following properties with confidence at least $1 - O(|V|^5)\delta_1$:*

1. *If $\{A, B\}$ is any ϵ_2-partition of (V_S, S, P), $\{A, B\}$ is refined by $\pi_S(\epsilon_2 + (|V| + 2)\epsilon_1)$.*
2. *If $C, D \in \pi_S(\epsilon_2 + (|V| + 2)\epsilon_1)$, then $I_P(C; D|S) \leq \epsilon_2 + (|V| + 3)\epsilon_1$.*
3. *If $\{C, D\}$ is any partition (of V_S) that is refined by $\pi_S(\epsilon_2 + 2\epsilon_1)$, then $I_P(C; D|S) \leq |V|^2 (\epsilon_2 + (|V| + 3)\epsilon_1) \leq |V|^3 (\epsilon_2 + 3\epsilon_1)$.*

Proof. We note that the loop in the algorithm shown above iterates at most $|V_S|$ times. During each iteration QA is called at most $|V_S|$ times, and hence there are at most $|V_S|^5 - 1$ queries of the form $\widetilde{F_{P,S}}(A)$ made by the algorithm shown. We have $\left|\widetilde{F_{P,S}}(A) - F_{P,S}(A)\right| \leq \epsilon_1$ for all queries $\widetilde{F_{P,S}}(A)$ by QA, with confidence at least $1 - (|V|^5)\delta_1$ by the union bound. In particular, if $\{A, B\}$ is an ϵ_2-partition of (V_S, S, P), then with confidence at least $1 - |V|^5 \delta_1$, it has to be a $(\epsilon_2 + (|V| + 2)\epsilon_1)$-partition of $(V_S, S, \tilde{P})$. Therefore, if $\pi_S(\epsilon_2 + (|V| + 2)\epsilon_1)$ is not a refinement of $\{A, B\}$, then the algorithm should not have stopped where it did, a contradiction.

The second assertion follows by induction and Lemma 6. The final assertion now follows from the second by noting that if Λ, Γ are disjoint, then $I_P(\cup_{\alpha\in\Lambda} Y_\alpha; \cup_{\beta\in\Gamma} Y_\beta|S) \leq \sum_{\alpha\in\Lambda,\beta\in\Gamma} I_P(Y_\alpha; Y_\beta|S) \leq |V|^2(\epsilon_2 + (|V|+3)\epsilon_1)$. □

Let $\mathcal{CIP}$ be the set of conditional independence partitions of the form $\{(S, \pi_S(\epsilon_2 + (|V|+2)\epsilon_1))\}$, where $S \subseteq V$ has size at most k. There are at most $\binom{|V|}{\leq k} \in O(|V|^k)$ sets of size at most k. Then, we have the following result

Corollary 8. *We can generate the set $\mathcal{CIP}$ is time $O\left(|V|^{k+5}\right)$, making at most $O\left(|V|^{k+5}\right)$ oracle calls. Further, $\mathcal{CIP}$ satisfies the following with confidence at least $\left(1 - |V|^{k+5}\delta_1\right)$.*

1. *If $(S, \pi_S) \in \mathcal{CIP}$ and $\{A, B\}$ is any ϵ_2-partition of (V, S, P), then π_S refines $\{A, B\}$.*
2. *If $(S, \pi_S) \in \mathcal{CIP}$ and $\{A, B\}$ is any partition that π_S refines, then $\{A, B\}$ is an $|V|^3(\epsilon_2 + 3\epsilon_1)$-partition.*

5 Partitions and Tree-Decompositions

If $G = (V, E)$ is a triangulated graph of tree-width k, then G has a tree-decomposition $(T = (I, F), \{V_i\}_{i\in I})$ of width k such that $v_iv_j \in E$ if and only if $\{v_i, v_j\} \subseteq V_i$ for some $i \in I$. Conversely, given a tree-decomposition $(T = (I, F), \{V_i\}_{i\in I})$ of width k, we get a triangulated graph $G = (V, E)$ of tree-width k where E contains edges $v_iv_j \in E$ if and only if $\{v_i, v_j\} \subseteq V_i$ for some $i \in I$. For any edge $ij \in F$, let $V_{ij} = V_i \cap V_j$. Then it follows from a result in [5] that the minimal separators of G are precisely the sets of the form V_{ij}. Since T is a tree, the removal of any edge $ij \in F$ breaks up T into two subtrees T_A and T_B. Let $V_{ij}^A = V(T_A)$ and $V_{ij}^B = V(T_B)$. If P factorizes over G, then $\left\{V_{ij}^A, V_{ij}^B\right\}$ is a 0-partition for $(V_{V_{ij}}, V_{ij}, P)$. In fact, we have

$$P\left(\{X_v\}_{v\in V}\right) = \frac{\prod_{i\in I} P\left(\{X_v\}_{v\in V_i}\right)}{\prod_{V_iV_j\in F} P\left(\{X_v\}_{v\in V_{ij}}\right)}$$

If P satisfies the above condition, we say that P factorizes over the tree-decomposition $(T = (I, F), \{V_i\}_{i\in I})$. It is clear that finding a triangulated graph G of width k such that $P(\cdot)$ factorizes over G is equivalent to finding a tree-decomposition over which $P(\cdot)$ factorizes. Now, we can ask a related question : Given a tree-decomposition $(T = (I, F), \{V_i\}_{i\in I})$, find a probability distribution $\tilde{P}(\cdot)$ which factorizes over this tree-decomposition such that $D(P\|\tilde{P})$ is minimized. It is clear that when $P(\cdot)$ factorizes over T, then $P(\cdot)$ is the unique optimal solution. More generally, we have the following result.

Lemma 9. *Let P be a probability distribution, and $(T = (I, F), \{V_i\}_{i\in I})$ a tree-decomposition. Then the unique minimizer of $D(P\|\tilde{P})$ subject to the constraint that $\tilde{P}$ factorizes over T is given by*

$$\tilde{P}\left(\{X_v\}_{v\in V}\right) = \frac{\prod_{i\in I} P\left(\{X_v\}_{v\in V_i}\right)}{\prod_{V_iV_j\in F} P\left(\{X_v\}_{v\in V_{ij}}\right)}$$

Further,

$$D(P\|\tilde{P}) \leq \sum_{V_iV_j\in F} I_P(V_{ij}^A; V_{ij}^B|V_{ij})$$

As before, let $\mathcal{CIP} = \{(S, \pi_S(\epsilon_2 + (|V|+3)\epsilon_1)) : S \subseteq V \text{ with } |S| \leq k\}$. We say that the tree-decomposition $(T = (I, F), \{V_i\}_{i\in I})$ is compatible with $\mathcal{P}$ if $\pi_{V_{ij}}$ refines $\left\{V_{ij}^A, V_{ij}^B\right\}$. We then have the following result.

Lemma 10. *Suppose that $(T = (I, F), \{V_i\}_{i\in I})$ is a tree-decomposition that is compatible with $\mathcal{CIP}$. If $\tilde{P}$ is chosen as in Lemma 9, then $D(P\|\tilde{P}) \leq |V|^4(\epsilon_2 + 3\epsilon_1)$.*

Proof. Let i be any interior vertex of T. Since $\mathcal{V}_i$ is a partition that is refined by π_{V_i}, by Corollary 8, the maximum possible loss of information induced by i is given by $|V|^3(\epsilon_2 + 3\epsilon_1)$. Therefore, $D(P\|\tilde{P}) \leq |I| \cdot |V|^3(\epsilon_2 + 3\epsilon_1) \leq |V|^4(\epsilon_2 + 3\epsilon_1)$. □

Therefore, if we restrict our search to tree-decompositions that are compatible with $\mathcal{CIP}$, then we are guaranteed to find a graphical model $(\tilde{G}, \tilde{P})$ such that $D(P\|\tilde{P}) \leq |I| \cdot |V|^3 (\epsilon_2 + 3\epsilon_1) \leq |V|^4 (\epsilon_2 + 3\epsilon_1)$. Therefore, by choosing ϵ_1 and ϵ_2 appropriately, we can make the error arbitrarily small. What we have not mentioned so far is how we determine a tree-decomposition that is compatible with $\mathcal{CIP}$. To do this, we use a simple variation of a dynamic programming algorithm proposed in [4]. This variant is shown as Algorithm 2. To prove that this algorithm works as advertised, we need to show that any tree-decomposition produced is compatible with $\mathcal{CIP}$, and that if there is a tree-decomposition compatible with $\mathcal{CIP}$, this algorithm will find a tree-decomposition. The essential idea behind the algorithm is that $\mathcal{M}$ contains the list of all subsets of vertices such that there is a tree-decomposition for a graph on these vertices of width at most k which is compatible with $\mathcal{CIP}$. Clearly any set of at most $k+1$ vertices satisfy this requirement (corresponding to a tree-decomposition consisting of just one vertex). Therefore, initially $\mathcal{M}$ contains all sets of size at most $k+1$. We associate with each element of M, a tree-decomposition, and a designated set S such that $S = V_{ij}$ for some $ij \in F$ if $F \neq \phi$, or $S \subseteq V_i$ otherwise. It then follows by induction that the algorithm will find a tree-decomposition compatible with $\mathcal{CIP}$ if one exists. Since $\mathcal{CIP}$ is $O(n^{k+1})$, it follows that this algorithm runs in time $O(n^{k+1})$. Hence we have the following result.

$\mathcal{S} \leftarrow \{(S, A) : (S, \pi_S) \in \mathcal{CIP}$ and $A \in \pi_S$ with $|A| > 1\}$;
$\mathcal{M} \leftarrow \{(S, A) : S, A \subseteq V$ with $|S \cup A| \leq k + 1\}$;
for $(S, A) \in \mathcal{M}$ **do**
  $I \leftarrow \{1\}$; $V_1 \leftarrow S \cup A$; $T \leftarrow (I, \phi)$;
  $\mathsf{treeDecomp}(S, A) \leftarrow (S, T, \{V_1\})$;
end
for $(S, A) \in \mathcal{S}$ *in order of increasing* $|S \cup A|$ **do**
  for $v \in S \cup A$ **do**
    $\mathcal{R} \leftarrow \{(L, B) \in \mathcal{M} : L \subseteq S \cup \{v\}$ and $L \cup B \subseteq A \cup S\}$;
    if $\cup_{(L,B)\in\mathcal{R}}(L \cup B) = A \cup S$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \cup \{(S, A)\}$;
      Order $\mathcal{R}$ arbitrarily : $\mathcal{R} = \{(L_1, B_1), \ldots, (L_m, B_m)\}$;
      $(R_i, T_i, \{V_j\}_{j\in J_i}) \leftarrow \mathsf{treeDecomp}(L_i, B_i)$; // Assume J_i disjoint. ;
      $I \leftarrow \cup_{i=1}^m J_i \cup \{r\}$;
    end
  end
end
if *there is a* $(S, \pi_S) \in \mathcal{CIP}$ *such that* $(S, A) \in \mathcal{M}$ *for every* $A \in \pi_S$ **then**
  Exit("A tree-decomposition was found");
else
  Exit("No tree-decomposition possible");
end

Algorithm 2: Finding partial k-trees given a set of conditionally independent partitions ($\mathcal{CIP}$)

Theorem 11. *For any* $\epsilon, \delta > 0$*, if we set* $\delta_1 = \frac{\delta}{|V|^5}$*, and* $\epsilon_2 = \epsilon_1 = \frac{\min(\epsilon,\alpha)}{4|V|^4}$*, the algorithm described will, in time* $O(n^{k+1})$*, produce a graphical model* $(\tilde{G}, \tilde{P})$ *such that* $D(P\|\tilde{P}) \leq \epsilon$ *with confidence* δ*.*

6 Conclusions

We have shown that by identifying approximate conditional independencies in a graphical model, we can identify the potential separators in the underlying graph. We have shown that this yields a polynomial time algorithm to properly efficiently PAC-learn a strongly connected graphical models of bounded tree-width. Interesting avenues for future research include investigating if this technique can be turned a robust learning scheme for larger classes of graphical models, and to see if this algorithm can be extended to classes other than bounded tree-width models.

References

[1] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees", IEEE Transactions on Information Theory, v. 14, Pages 462–467

[2] M. Grötschel, L. Lovász and A. Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In *Combinatorica*, v. 1, Pages 169–197, 1981.

[3] L. Lovász. "Submodular functions and convexity". In *Mathematical Programming – The State of the Art*, A. Bachem, M. Grötchel and B. Korte, eds., Springer-Verlag, Pages 235–257, 1983.

[4] S. Arnborg, D. G. Corneil and A. Proskurowski. "Complexity of finding embeddings in a k-tree". In SIAM J. Alg. Disc. Meth., v. 8, No. 2, April 1987

[5] C. W. Ho and R. C. T. Lee. "Counting clique trees and computing perfect elimination schemes in parallel", In Information Processing Letters v. 31, Pages 61-68, 1989.

[6] H. L. Bodlaender, "A tourist guide through treewidth", In Acta Cybernetica, 1993.

[7] K-U. Höffgen. "Learning and Robust Learning of Product Distributions." In COLT '93.

[8] P. Dagum and M. Luby "Approximating probabilistic inference in belief networks is NP-hard", Artificial Intelligence, v. 60, Pages 141–153, 1993.

[9] M. Queyranne. "Minimizing symmetric submodular functions", in *Math. Programming*, 82 (1998), Pages 3–12.

[10] D. Chickering, "Learning Bayesian networks is NP-complete", in Fisher, D. and Lenz, H. editors, *Learning from Data"*, Pages 121–130, Springer-Verlag

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

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

[13] S. Dasgupta, "Learning polytrees", In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Pages 134–141, 1999.

[14] D. Karger and N. Srebro. "Learning Markov networks: Maximum bounded tree-width graphs." In *Symposium on Discrete Algorithms*, 2001, Pages 391-401.

[15] C. Meek, "Finding a path is harder than finding a tree", JAIR v. 15, Pages 383–389, 2001.

[16] M. Narasimhan and J. Bilmes, "Efficient proper PAC-learning of bounded tree-width graphical models", Technical Report UWEETR-2004-0009, University of Washington, 2004.

A Proof of Lemma 4

An ordered pair (t, u) called a pendent pair for (V, f) if $\{u\} \in \arg\min_{\{U \subseteq V : u \in U,\, t \notin U\}} f[u]$. Algorithm 3 is a subroutine used in Queyranne's algorithm. When f is symmetric and submodular, the subroutine returns a pendant pair (t, u). We consider the case when f is approximately symmetric and submodular. By this we mean that there is an $\epsilon > 0$ such that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B) - \epsilon$, and $|f(A) - f(V \setminus A)| \leq \epsilon$ for every $A, B \subseteq V$. We then have the following equivalent to Lemma 1 in [9], which is proven using a variant of their argument.

Lemma 12. *Suppose that $f : 2^V \to \mathbb{R}$ is approximately symmetric and submodular. Then for every $1 \leq i \leq n-1$, $y \notin W_i$ and $X \subseteq W_{i-1}$, we have*

$$f[W_i] + f[y] \leq f[W_i \setminus X] + f[X \cup y] + (i-1)\epsilon$$

Data : A set V and an f-valued oracle
Result : A pendent pair (t, u)
$v_1 \leftarrow$ arbitrary x; $W_1 \leftarrow \{v_1\}$
for $i = 1 : n-1$ **do**
  $\forall u \in V \setminus W_i$ **do** $\mathsf{key}[u] \leftarrow f[W_i \cup u] - f[u]$;
  $v_{i+1} \leftarrow \arg\min_{u \in V \setminus W_i} \mathsf{key}[u]$
  $W_{i+1} \leftarrow W_i \cup v_{i+1}$
end
return(v_{n-1}, v_n)

Algorithm 3: PendentPair

Proof. We show this by induction. For $i = 1$, $W_{i-1} = \phi$, and so we only need check that $f[W_i]+f[y] \leq f[W_i]+f[y]$, which is clearly true. Suppose that the result holds for all $1 \leq i < k$. Consider any $S \subseteq W_{k-1}$. Let j be the smallest integer such that $S \subseteq W_{j-1}$. If $j = k$, then $v_{k-1} \in S$ and $W_{k-1} \setminus S \subseteq W_{k-2}$. Therefore,

$$\begin{aligned} f[W_k \setminus S] + f[S \cup u] &\overset{(A)}{=} f[(W_{k-1} \setminus S) \cup v_k] + f[S \cup u] \\ &\overset{(B)}{\geq} f(W_{k-1}) + f(v_k) - f(S) + f(S \cup u) - (k-2)\epsilon \\ &\overset{(C)}{\geq} f(W_{k-1} \cup u) + f(v_k) - (k-1)\epsilon \\ &\overset{(D)}{\geq} f(W_k) + f(u) - (k-1)\epsilon \end{aligned}$$

Where (A) follows because $W_k \setminus S = (W_{k-1} \setminus S) \cup v_k$, (B) follows from the inductive hypothesis by taking $y = v_k \notin W_{k-1}$ and $X = W_{k-1} \setminus S$, (C) follows from approximate submodulartity by taking $A = W_{k-1}$ and $B = S \cup u$. Then $A \cup B = W_{k-1} \cup u$ and $A \cap B = S$. To see (D), note that at step k, $v_k \in V \setminus W_{k-1}$ is selected so that $v_k = \arg\min_{u \in V \setminus W_{k-1}} (f[W_{k-1} \cup u] - f[u])$. Therefore, for $u \notin W_k$, we have $f[W_{k-1} \cup u] - f[u] \geq f[W_{k-1} \cup v_k] - f[v_k] = f[W_k] - f[v_k]$.

If $j < k$, then $v_{j-1} \in S$ and none of $v_j, \ldots, v_k$ are in S. We have

$$\begin{aligned} f[W_k \setminus S] + f[S + u] &= f[(W_{j-1} \setminus S) \cup (W_k \setminus W_{j-1})] + f[S \cup u] \\ &\overset{(F)}{\geq} f[(W_{j-1} \setminus S) \cup (W_k \setminus W_{j-1})] \\ &\quad + f[W_j] - f[W_j \setminus S] + f[u] - (j-2)\epsilon \\ &\overset{(G)}{\geq} f[W_k] + f[u] - (j-2)\epsilon \end{aligned}$$

Since $j < k$, we may use the inductive hypothesis for $y = u$, and $X = S \subseteq W_{j-1}$ to get (F). Then we use approximate submodularity by letting $A = (W_{j-1} \setminus S) \cup (W_k \setminus W_{j-1})$ and $B = W_j$. Then $A \cup B = W_k$ and $A \cap B = W_j \setminus S$. to get (G). Therefore, the claim holds by induction. □

Theorem 13. *Suppose that f is approximately symmetric and submodular. Then the pendent pair (t, u) returned Algorithm 3 satisfies*

$$f[u] \leq \min_{\{\, U \subseteq V \,:\, u \in U,\, t \notin U \,\}} f[U] + \frac{(n+1)\epsilon}{2}$$

Proof. Use Lemma 12 with $i = n-1$ and $y = v_n$. Then by approximate symmetry, we must have $|f[v_n] - f[W_{n-1}]| \leq \epsilon$, and $|f[W_{n-1} \setminus X] - f[X \cup v_n]| \leq \epsilon$. Hence $f[v_n] + f[W_{n-1}] \leq 2f[v_n] + \epsilon$, and $f[W_{n-1} \setminus X] + f[X \cup v_n] \leq 2f[X + v_n] + \epsilon$. Therefore,

$$\begin{aligned} 2f[v_n] - \epsilon &\leq f[v_n] + f[W_{n-1}] \\ &\leq f[W_{n-1} \setminus X] + f[X \cup v_n] + (n-1)\epsilon \\ &\leq 2f[X \cup v_n] + n\epsilon \end{aligned}$$

Therefore, $f[v_n] \leq f[W_{n-1} \setminus X] + \frac{(n+1)\epsilon}{2}$. This holds for every $X \subseteq W_{n-1} = V \setminus \{v_n\}$. The result follows. □

Therefore, Algorithm 3 returns an approximate pendent pair using $O(n^2)$ time. Therefore, this theorem is the approximate equivalent to Theorem 2 of [9]. We now consider two cases. If $\{u\}$ is an approximate minimizer of f, then we are done. So we suppose tha this is not the case. Then if $U \subseteq V$, with $u \in U$ and $v \notin U$, then by Theorem 13, we must have $f[u] \leq f[U] + \frac{(n+1)\epsilon}{2}$. Therefore, U cannot be an approximate minimizer either. Hence we only need consider sets U with $\{u, v\} \subseteq U$. Therefore, we consider the modified problem of finding the optimal solution $\arg\min_{U \subseteq V_1} f_1[U]$, where $V_1 = V \setminus \{u\}$ and

$$f_1[U] = \begin{cases} f[U] & \text{if } v \notin U \\ f_1[U \cup \{u\}] & \text{if } v \in U \end{cases}$$

This may be thought of as the result of merging v and u into a single element, as we may do since either both u and v are in U or neither are. The approximate symmetry and approximate submodularity of f_1 follows from the approximate symmetry and submodularity of f. Therefore, we may use a simple recursive algorithm to determine the approximate minimal minimizer of f as shown in Algorithm 4. While it is very straightforward to think of this

Data : A set V and an f-valued oracle
Result : An (f approximate minimal) proper subset $U \subset V$
$(v, u) \leftarrow \mathsf{PendentPair}(V, f, x)$ for x arbitrary;
$V_1 \leftarrow V \setminus \{u\}$;
$f_1 \leftarrow$ merged oracle for f;
$U_1 \leftarrow \mathsf{OptimalSet}(V_1, f_1)$;
if $f[u] \leq f_1[U]$ **then**
  $U \leftarrow \{u\}$
else
  $U \leftarrow \begin{cases} U \text{ if } v \notin U \\ U \cup \{u\} \text{ if } v \in U \end{cases}$
end
return(U)

Algorithm 4: OptimalSet (Recursive Version)

algorithm recursively, we can also use a non-iterative algorithm as shown in Algorithm 5. The iterative algorithm is essentially the same as the algorithm presented in [9]. Therefore, the following theorem, which is the equivalent of

Data : A set V and an f-valued oracle
Result : An (f approximate minimal) proper subset $U \subset V$
$(V_0, f_0) \leftarrow (V, f)$;
for $i \leftarrow 0$ *to* $n-2$ **do**
  $(v_i, u_i) \leftarrow \mathsf{PendentPair}(V_i, f_i)$;
  $s_i \leftarrow f_i[u_i]$;
  $U_i \leftarrow$ unmerged version of $\{u_i\}$;
  $V_{i+1} \leftarrow V_i \setminus \{u_i\}$; $f_{i+1} \leftarrow$ merged oracle for f_i;
end
Pick $i \in \arg\min \{s_i\}_{i=0}^{n-2}$;
return(U_i)

Algorithm 5: OptimalSet (Iterative Version)

Theorem 3 of [9] follows.

Theorem 14. *Algorithm* OptimalSet *returns a set U which satisfies $f[U] \leq f[U_1] + (n+1)\epsilon$ for all $U_1 \subset V$.*