Pruning Rules for Learning Parsimonious Context Trees

Ralf Eggeling **Mikko Koivisto**
University of Helsinki, Department of Computer Science,
Helsinki Institute for Information Techology HIIT, Finland
{eggeling,mkhkoivi}@cs.helsinki.fi

Abstract

We give a novel algorithm for finding a parsimonious context tree (PCT) that best fits a given data set. PCTs extend traditional context trees by allowing context-specific grouping of the states of a context variable, also enabling skipping the variable. However, they gain statistical efficiency at the cost of computational efficiency, as the search space of PCTs is of tremendous size. We propose pruning rules based on efficiently computable score upper bounds with the aim of reducing this search space significantly. While our concrete bounds exploit properties of the BIC score, the ideas apply also to other scoring functions. Empirical results show that our algorithm is typically an order-of-magnitude faster than a recently proposed memory-intensive algorithm, or alternatively, about equally fast but using dramatically less memory.

1 INTRODUCTION

The conditional distribution of a response variable y, given some explanatory variables $x_1, x_2, \ldots, x_d$, is a key ingredient in common probabilistic models. Often the modeler's interest is in distributions that admit a compact, structured representation, thereby facilitating statistically efficient learning and computationally efficient inference, as well as easy human interpretation.

Examples of general-purpose model classes include decision trees (Breiman et al., 1984; Friedman & Goldszmidt, 1996), decision graphs (Oliver, 1993; Chickering et al., 1997), multi-linear functions (Chavira & Darwiche, 2005), and conditional independence trees (Su & Zhang, 2005). These models allow for representing *context-specific independence* (Boutilier et al., 1996): given a *context*, i.e., an assignment for a *subset* of the explanatory variables x_i, the response y becomes independent of the rest.

When the explanatory variables are equipped with a natural linear ordering, more specialized models of context-specific independence are justified. In particular, *context trees* (CTs) of depth d over an alphabet Ω (Rissanen, 1983; Bühlmann & Wyner, 1999) model the distribution of the next symbol y after a sequence $x_d x_{d-1} \cdots x_1$ assuming each context is an assignment $a_\ell \cdots a_1$ for the $\ell \leq d$ immediate predecessors $x_\ell \cdots x_1$ of y, where the length ℓ may vary with the context. While context trees excel in computational efficiency, their statistical efficiency decays when there are long-range dependencies (requiring long contexts) and when the alphabet is non-binary.

To address the shortcoming of CTs, Bourguignon & Robelin (2004) proposed *parsimonious context trees* (PCTs), in which a context is a sequence $C_\ell \cdots C_1$ of *sets* of symbols $C_i \subseteq \Omega$. The idea is that the conditional distribution of y is the same of all assignments $a_d \cdots a_1$ that *match* the context, that is, $a_i \in C_i$ for $i = 1, \ldots, \ell$. In effect, PCTs allow for a compact representation and statistically efficient learning even in the presence of long-range dependencies. PCTs have found applications particularly within computational biology (Seifert et al., 2012; Eggeling et al., 2015b), where modeling sequential data over discrete alphabets constitutes a recurring challenge.

From a computational point of view, learning PCTs (i.e., maximizing a given scoring function) is, however, very challenging for larger depth d and alphabet size $|\Omega|$. The dynamic programming (DP) algorithm of Bourguignon & Robelin (2004) avoids explicit enumeration of all PCTs. Yet, it has to explore all possible contexts $C_\ell \cdots C_1$, with $\emptyset \subset C_i \subseteq \Omega$, each of which is a potential node of an optimal PCT and corresponds to a subproblem of optimizing the subtree rooted at it. Recently, Eggeling et al. (2015a) enhanced the DP algorithm by observing that two contexts of the same length are equivalent (i.e., have identical optimal subtrees) if they are matched by exactly the same set of data points; thus the respective subproblem needs to be solved only once. While this *memoization* variant is effective in reducing the time requirement of PCT optimization in practice, it has the disadvantage of increased memory consumption.

Here, we investigate a new idea to expedite PCT optimization in practice. We present pruning rules, which allow us to ignore subproblems that are guaranteed to not contribute to an optimal PCT. To obtain such guarantees, we derive upper bounds specifically for the BIC score, similar in spirit to the bounds on local scores by Tian (2000) and de Campos & Ji (2011) for structure learning in Bayesian networks. While for Bayesian networks also *global* bounds, useful in branch-and-bound search, can be derived by structural relaxations (de Campos & Ji, 2011; Yuan & Malone, 2013), with no assumptions on local scores, for PCTs a separate global view is absent and utilizing concrete properties of a particular score seems necessary.

As our subproblems form a so-called AND/OR search space (Dechter & Mateescu, 2007; Nilsson, 1980), we cannot prune a subproblem based solely on the associated bound, but we need to combine the bounds of a multitude of subproblems; in essence, we have to consider all alternative partitions of a subproblem into smaller subproblems, all of which need to be solved. To this end, we propose an efficient treatment, which resembles a simple pruning rule popular in structure learning in Bayesian networks (Teyssier & Koller, 2005); however, while the latter concerns the "is subset of" relation, our rule concerns the "refines" relation on set partitions. We note that our algorithm is an instantiation of so-called *General Branch and Bound* (Nau et al., 1984), but not of the more special A* algorithm for OR spaces.

Our pruning technique is orthogonal to the memoization technique of Eggeling et al. (2015a). In particular, our technique can be employed with or without memoization, resulting in high or low memory consumption, respectively. Because the effectiveness of our ideas is data-dependent, we evaluate the proposed algorithms empirically on a wide selection of real-world data sets.

2 PRELIMINARIES

In this section, we revisit the definition of PCTs, the basic structure learning algorithm, and a recently proposed memoization technique for improving it.

2.1 PARSIMONIOUS CONTEXT TREES

A *parsimonious context tree* (PCT) $\mathcal{T}$ of depth d over an alphabet Ω is a rooted, balanced, node-labeled tree of depth d with the following additional property: For each node of depth $\ell < d$ the labels of its children partition Ω, i.e., the labels of the children are pairwise disjoint nonempty subsets of Ω whose union is Ω.

We identify each node with the sequence of labels $V_\ell \cdots V_1$ of the nodes on the unique path from the node up to the root, denoted by $\mathbf{V}$ for short. We can also interpret the node $\mathbf{V}$ as the set of all sequences it represents. We say that a sequence $x_d \cdots x_1$ *matches* node $\mathbf{V}$, and denote it by $x_d \cdots x_1 \in \mathbf{V}$, if $x_i \in V_i$ for all positions $i = \ell, \ldots, 1$ (the remaining positions are ignored).

Given a PCT $\mathcal{T}$ and its node $\mathbf{V}$, we denote by $\mathcal{T}(\mathbf{V})$ the subtree of $\mathcal{T}$ rooted at $\mathbf{V}$. We say the subtree is *minimal* if it consists of a single chain of nodes down to a single leaf, thus all nodes labeled by Ω, and *maximal* if it consists only of nodes labeled with single symbols $a \in \Omega$, thus having $|\Omega|^{d-\ell(\mathbf{V})}$ leaves. Here, $\ell(\mathbf{V})$ denotes the depth of node $\mathbf{V}$.

To model the conditional distribution of the response variable y given a sequence $\mathbf{x}$, we equip each leaf $\mathbf{V}$ of a PCT with $|\Omega|$ parameters $\theta_{\mathbf{V}a}$, one parameter for each $a \in \Omega$. We interpret $\theta_{\mathbf{V}a}$ as the probability that y takes the value a given that $\mathbf{x}$ matches $\mathbf{V}$. In a data set $\mathbf{z} = (\mathbf{x}^t, y^t)_{t=1}^N$ we assume that, given $\mathbf{x}^t$, the response y^t is independent of the remainder of the data. Writing $\Theta_\mathcal{T}$ for the parameters of a PCT $\mathcal{T}$, we obtain the likelihood function

$$\mathcal{L}_\mathcal{T}(\Theta_\mathcal{T}) := \prod_{\text{leaf } \mathbf{V} \text{ of } \mathcal{T}} \prod_{a \in \Omega} \theta_{\mathbf{V}a}^{N_{\mathbf{V}a}}\,, \tag{1}$$

where $N_{\mathbf{V}a}$ denotes the count of the response a in data points where the explanatory variables match $\mathbf{V}$:

$$N_{\mathbf{V}a} := |\{t : \mathbf{x}^t \in \mathbf{V} \text{ and } y^t = a\}|\,. \tag{2}$$

We will further denote $N_\mathbf{V} := \sum_{a \in \Omega} N_{\mathbf{V}a}$.

2.2 BASIC STRUCTURE LEARNING

We consider a score-and-search approach to learn a PCT from a given data set. Suppose we are given a scoring function S that associates each PCT $\mathcal{T}$ with a real-valued score $S_\mathcal{T}$. The task is to find an optimal PCT,

$$\hat{\mathcal{T}} \in \operatorname*{argmax}_\mathcal{T} S_\mathcal{T}\,. \tag{3}$$

Aside from the fact that multiple PCTs may achieve the optimum, this task is practically equivalent to the task of finding the optimal score $S_{\hat{\mathcal{T}}}$. For convenience, we focus on the latter problem for the rest of the paper. An optimal tree $\hat{\mathcal{T}}$ can be constructed via, e.g., standard backtracking (see Section 2.3 for details).

Bourguignon & Robelin (2004) presented a dynamic programming algorithm for finding $S_{\hat{\mathcal{T}}}$. It relies on the mild assumption that the scoring function decomposes into a sum of leaf scores:

$$S_\mathcal{T} = \sum_{\text{leaf } \mathbf{V} \text{ of } \mathcal{T}} S(\mathbf{V})\,. \tag{4}$$

This property is fulfilled by the log-likelihood function (Eq. 1) and thus also by scoring functions that can be written as penalized maximum log-likelihood with a decomposable penalty term, such as AIC (Akaike, 1974), BIC (Schwarz, 1978), and the Bayesian marginal likelihood with a decomposable prior.

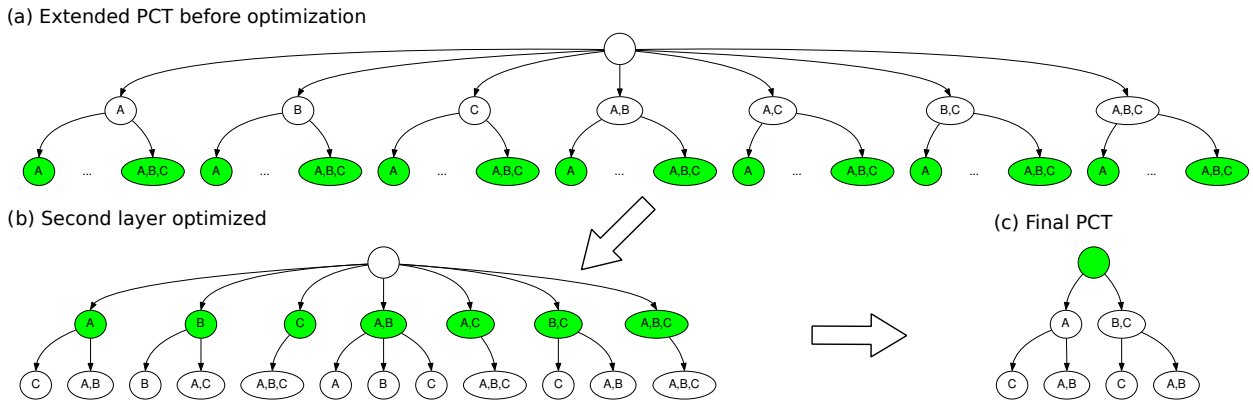

Figure 1: **Basic PCT optimization**. We show the bottom-up reduction of the extended PCT for a toy example of $d = 2$ and $\Omega = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. (a) Initially only the leaf scores of the extended PCT have an exact, optimal score assigned to them. (b) For each set of sibling leaves, the optimal valid selection of children is computed, the non-contributing siblings are discarded, and the winning score is propagated upwards to become the score of the parent. (c) The same principle is applied on the higher layer in order to select the optimal children of the root, obtaining a valid PCT with optimal score.

To describe the algorithm, we denote by $S_{\mathcal{T}(\mathbf{V})}(\mathbf{V})$ the sum of the leaf scores in the subtree $\mathcal{T}(\mathbf{V})$ rooted at an inner node $\mathbf{V}$ of $\mathcal{T}$; for a leaf $\mathbf{V}$, we put $S_{\mathcal{T}(\mathbf{V})}(\mathbf{V}) := S(\mathbf{V})$. We have the recurrence

$$S_{\mathcal{T}(\mathbf{V})}(\mathbf{V}) = \sum_{\text{child } \mathbf{C} \text{ of } \mathbf{V}} S_{\mathcal{T}(\mathbf{C})}(\mathbf{C}) \,, \tag{5}$$

and, in particular, $S_{\mathcal{T}} = S_{\mathcal{T}}(\Lambda)$, where Λ is the root node of $\mathcal{T}$. Exploiting this recurrence, the algorithm of Bourguignon & Robelin (2004) optimizes the score over the subtrees rooted at $\mathbf{C}$, independently for each possible child node $\mathbf{C}$, and then selects a set of children that form an optimal partition of the parent node $\mathbf{V}$. For the base case of each leaf, the algorithm sets the optimal score $S_*(\mathbf{V}) := S(\mathbf{V})$, and for each inner node it computes the optimal score defined by

$$S_*(\mathbf{V}) := \max_{\substack{\{C_1,\ldots,C_r\} \\ \text{partition of } \Omega}} \left\{ S_*(C_1\mathbf{V}) + \cdots + S_*(C_r\mathbf{V}) \right\}. \tag{6}$$

It follows that the maximum score over all PCTs of depth d is obtained as

$$\max_{\mathcal{T}} S_{\mathcal{T}} = S_*(\Lambda) \,. \tag{7}$$

The algorithm computes the leaf scores of $(2^{|\Omega|} - 1)^d$ possible leaves and, in addition, finds an optimal set of children in $O(3^{|\Omega|})$ time for each of slightly more than $(2^{|\Omega|} - 1)^{d-1}$ inner nodes. Since the complexitity is a product of two terms which are both exponential in $|\Omega|$, PCTs are to date limited to applications of small to moderate alphabet size (Eggeling et al., 2015a).

2.3 INTERPRETATION AS REDUCTION OF EXTENDED PCT

The inner workings of the algorithm of Bourguignon & Robelin (2004) and the construction of the optimal PCT itself can be viewed as bottom-up reduction of a data structure called *extended PCT*, as illustrated in Figure 1. In contrast to a PCT, the sibling nodes in an extended PCT do not partition their parent node, but are labeled by all nonempty subsets of Ω. An extended PCT thus contains all possible PCTs as subtrees, as illustrated in Figure 1a. We denote the set of all nodes of an extended PCT by $\mathcal{V}$, and treat a PCT $\mathcal{T}$ as its proper subset, i.e., $\mathcal{T} \subset \mathcal{V}$.

Given an extended PCT, the base case of the algorithm requires the computation of leaf scores, and the task is now to reduce the extended PCT so that a PCT with maximal score remains. For each inner node it then (i) computes an optimal selection of children with the constraint that the node labels must form a partition of Ω, and (ii) removes all children and subtrees below that do not belong to this optimal partition.

Since an inner node can be optimized only if all of its children have already a score attached to them and are thus roots of valid PCT subtrees, the recursion leads essentially to a bottom-up reduction of the extended PCT beginning at the deepest *layer*, which comprises all nodes of the same depth, as displayed in Figure 1b. The optimization terminates once an optimal selection of children of the root node are computed; a possible final result is shown in Figure 1c.

2.4 MEMOIZATION

Eggeling et al. (2015a) proposed a *memoization* technique for speeding up PCT learning by exploiting regularities in the observed explanatory variables.

The core idea is to implement the algorithm of Bourguignon & Robelin (2004) in a top-down fashion and to store the result of subtree optimization for node $\mathbf{V}$ with the depth

$\ell(\mathbf{V})$ and the index set $I(\mathbf{V}) := \{t : \mathbf{x}^t \in \mathbf{V}\}$ as key, e.g., in a hash table. When the algorithm needs an optimal subtree (or its score) for another node $\mathbf{V}'$, the algorithm checks whether the key $(\ell(\mathbf{V}'), I(\mathbf{V}'))$ exists in the storage already and the associated result can be re-used. To work correctly, memoization requires that the score $S_*(\mathbf{V})$ depends on $\mathbf{V}$ only through $I(\mathbf{V})$ and $\ell(\mathbf{V})$. This property holds for many relevant scoring functions such as penalized maximum log-likelihood scores, but not for the Bayesian marginal likelihood with context-dependent pseudo-counts.

While memoization has shown to be rather effective particularly on highly structured data sets, it increases memory consumption. The original DP algorithm of Bourguignon & Robelin (2004) is relatively memory-efficient as only a small fraction of nodes of the extended PCT needs to be stored in memory at a given time, provided that the extended PCT is constructed, traversed, and reduced top-down in a depth-first manner. Memoization, in contrast, requires storing the scores and the associated data subset indices for all visited inner nodes of the extended PCT.

3 PRUNING RULES

We derive pruning rules, which utilize score upper bounds to decide at any given node whether we can avoid the explicit optimization over possible subtrees.

3.1 SCORING FUNCTION: BIC

In order to obtain effective upper bounds, we focus on the BIC score, derived from be Bayesian Information Criterion (Schwarz, 1978). It is an approximation of the Bayesian marginal likelihood and has been empirically shown to be suitable for PCT learning on real-world data due to its harsh penalty term, which favors sparse trees (Eggeling et al., 2014). The score can also be given a two-part-MDL interpretation (Rissanen, 1978). For a PCT $\mathcal{T}$, the BIC score is given by

$$S_{\mathcal{T}}^{\text{BIC}} = \ln \mathcal{L}_{\mathcal{T}}(\hat{\Theta}_{\mathcal{T}}(\mathbf{z})) - \frac{k}{2} \ln N \,, \tag{8}$$

where $\hat{\Theta}_{\mathcal{T}}(\mathbf{z})$ denotes the maximum-likelihood parameter estimate on $\mathbf{z}$ and k denotes the number of free parameters. Since k is proportional to the number of leaves and since the likelihood (Eq. 1) is a product of leaf terms, the BIC score decomposes into a sum of

$$S^{\text{BIC}}(\mathbf{V}) := \overbrace{\sum_a N_{\mathbf{V}a} \ln \frac{N_{\mathbf{V}a}}{N_{\mathbf{V}}}}^{L(\mathbf{V})} - \overbrace{\frac{1}{2}\big(|\Omega| - 1\big) \ln N}^{K} \,. \tag{9}$$

Here, $L(\mathbf{V})$ is the maximized log-likelihood of leaf $\mathbf{V}$ and K is the BIC penalty contribution of a single leaf, involving $|\Omega| - 1$ free parameters and a global sample size of N. We observe that the score of a leaf $\mathbf{V}$ does actually not depend on all data points in $\mathbf{z}$, but only on those that match $\mathbf{V}$.

3.2 UPPER BOUNDING THE BIC SCORE

Consider an inner node $\mathbf{V} \in \mathcal{V}$. To upper bound the BIC score over all possible subtrees rooted at $\mathbf{V}$, we upper bound the largest possible gain in the maximum-likelihood term on one hand, and lower bound the inevitable penalty due to increased model complexity on the other hand.

Consider first the likelihood term. We make use of the observation that every PCT is *nested* in the maximal PCT, which has $|\Omega|^d$ leaves, each representing a single realization of the explanatory variables. The same holds also locally for any PCT subtree below a particular node $\mathbf{V}$. Hence, the likelihood term is maximized by the maximal model, which partitions all realizations perfectly. We obtain the upper bound

$$\tilde{L}(\mathbf{V}) := \sum_{\mathbf{a} \in \Omega^{d-\ell(\mathbf{V})}} L(\mathbf{aV}) \,, \tag{10}$$

that is, we have $L(\mathbf{V}) \leq \tilde{L}(\mathbf{V})$. Here, $\mathbf{aV}$ denotes the leaf node obtained by extending the context of node $\mathbf{V}$ with a single realization $\mathbf{a}$ of the remaining explanatory variables. Note that $\tilde{L}(\mathbf{V})$ can be computed fast by summing over the sequences $\mathbf{a}$ that occur in the data points that match $\mathbf{V}$. Now we distinguish:

Case 1 The minimal subtree is optimal. In this case, we can compute the exact score directly, without recursion.

Case 2 The minimal subtree is not optimal. Thus an optimal subtree makes at least one split, and therefore has at least two leaves. In this case, the likelihood term is upper bounded by $\tilde{L}(\mathbf{V})$ (Eq. 10), while the penalty term is at least $2K$.

Combining the two cases yields the following bound.

Proposition 1 (Score upper bound)**.** *For an arbitrary inner node $\mathbf{V} \in \mathcal{V}$ it holds that*

$$S_*^{BIC}(\mathbf{V}) \leq S_0^{BIC}(\mathbf{V}) \,,$$

with

$$S_0^{BIC}(\mathbf{V}) := \max\{L(\mathbf{V}) - K, \tilde{L}(\mathbf{V}) - 2K\} \,.$$

We will use this upper bound twice to device two pruning rules in the next two sections.

3.3 STOPPING RULE

First, we consider a simple rule for pruning, which is an immediate consequence of the aforementioned upper bound. The idea can be phrased as follows:

> *Stop optimizing the subtree below a node when context-specific independence can be declared already without even entering the recursion.*

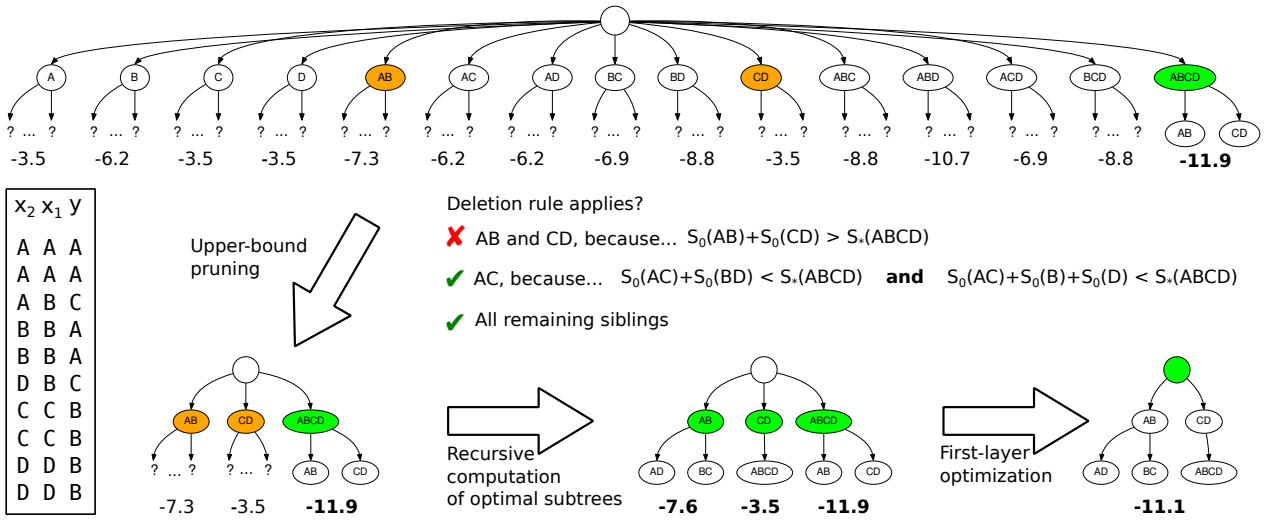

Figure 2: **Example of the deletion rule.** We consider a small data set of $N = 10$ for two explanatory variables over the alphabet $\Omega = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$ (bottom-left box). The root of each subtree for which exact BIC scores have been computed already is marked in green and corresponding score displayed in boldface below the subtree. The remaining other subtrees are associated with an upper bound on the BIC score. The root of a subtree which contributes to an upper-bounded partition score that is greater than the exact score of the maximal sibling node, and thus has to optimized explicitly, is displayed in orange.

Formally, we have the following.

Proposition 2 (Stopping rule)**.** *Let* $\mathbf{V} \in \mathcal{V}$ *with* $\ell(\mathbf{V}) < d$*, and let* $\mathcal{T}'$ *be the minimal subtree rooted at* $\mathbf{V}$*. If*

$$S_0^{BIC}(\mathbf{V}) = L(\mathbf{V}) - K\,,$$

then $S_*^{BIC}(\mathbf{V}) = S_{\mathcal{T}'}^{BIC}(\mathbf{V})$.

The correctness can be shown by contradiction. Assume $\mathcal{T}'$ does not yield the optimal score. Then $L(\mathbf{V}) - K < S_*^{\text{BIC}}(\mathbf{V})$. But since $S_*^{\text{BIC}}(\mathbf{V}) \leq S_0^{\text{BIC}}(\mathbf{V})$, this violates the premise.

3.4 DELETION RULE

The idea of our second rule is to identify a node with so low a score that the node cannot appear in any optimal PCT. In an idealized form, the rule reads:

> *Delete a child node if the best set of children it belongs to is worse than some other set of children (to which the node does not belong).*

As we wish to delete as many potential child nodes as possible and not compute their optimal scores, we cannot assume the optimal scores of the sibling nodes are available. Thus, to make the rule concrete, we resort to upper bounds on the scores. Likewise, we need to lower bound the optimal score among the valid sets of children. While, in principle, various lower bounding schemes would be possible, we have chosen to use a particularly simple bound: the optimal score of the Ω-labeled child. We next describe the bounds and the rule more formally.

Consider a node $\mathbf{V}$. To efficiently check whether a child node $C\mathbf{V}$ can be deleted, we need an upper bound on the score obtained by a partition of $\mathbf{V}$ that includes $C\mathbf{V}$. To this end, we associate any set function $f : 2^\Omega \to \mathbb{R}$ with another function $f' : 2^\Omega \to \mathbb{R}$ defined by letting $f'(\emptyset) := 0$ and, for all $\emptyset \subset B \subseteq \Omega$,

$$f'(B) := \max_{\substack{\{C_1,\ldots,C_r\} \\ \text{partition of } B}} \left\{ f(C_1) + \cdots + f(C_r) \right\}. \qquad (11)$$

A folklore dynamic programming algorithm computes f' for a given f in $O(3^{|\Omega|})$ time, based on the recurrence

$$f'(B) = \max_{\emptyset \subset C \subseteq B} \left\{ f(C) + f'(B \setminus C) \right\}. \qquad (12)$$

Eggeling et al. (2015a) made use of this observation to compute the optimal score over all partitions of the alphabet, given by $f'(\Omega)$ with a suitable function f. Here, we apply it to score upper bounds, and we also use several of the values $f'(B)$ in order to stay within $O(3^{|\Omega|})$ for computing all upper bounds required:

Proposition 3 (Deletion rule)**.** *Let* $\mathbf{V} \in \mathcal{V}$ *with* $\ell(\mathbf{V}) < d$*, and let* $\emptyset \subset C \subset \Omega$*. Let* $f(C') := S_0(C'\mathbf{V})$ *for all* $\emptyset \subset C' \subseteq \Omega$*. If*

$$S_0(C\mathbf{V}) + f'(\Omega \setminus C) \;<\; S_*(\Omega\mathbf{V})\,,$$

then the node $C\mathbf{V}$ *does not belong to any optimal PCT.*

The correctness can be shown by contradiction. Suppose $C\mathbf{V}$ did belong to an optimal PCT even though the premise was fulfilled. Then $S_*(C\mathbf{V}) + f'(\Omega \setminus C) \geq S_*(\Omega\mathbf{V})$, leading to $S_0(C\mathbf{V}) < S_*(C\mathbf{V})$, which violates the property of S_0 being an an upper bound of S_*.

We illustrate the deletion rule by a small toy example in Figure 2. Here, we focus on the first layer, where only for the maximal node an optimal PCT subtree is computed and thus an exact score is available already, whereas for the other siblings only upper-bounded scores exist. Based on those scores, we compute the upper-bounded scores of every possible partition and observe and observe that only one partition, consisting of the two child nodes AB and CD, has an upper-bounded score that is greater than the exact score of the maximal sibling node ABCD. All other siblings can thus not contribute to an optimal partition of child nodes, as even the best partition they contribute to has an upper-bounded score smaller than what is already achieved. Hence, the corresponding subtrees do not need to be optimized explicitly and can be deleted.

The deletion rule does invest a certain amount of effort that the obtained savings need to compensate before the rule becomes effective: In the worst case, we need to compute the optimal partition of children twice for each inner node, once with the upper-bounded scores for excluding subtrees from further optimization, and once with the exact scores. As a positive side note, we observe that, while we focus on BIC upper bounds in this work, the deletion rule is in principle independent of the used scoring function, as long as an effective upper bound $S_0 \leq S_*$ can be specified.

3.5 LOOKAHEAD

The upper bound of Section 3.2 can be computed directly for a given node without entering the recursion. However, we can tighten the bound, if we do enter the recursion for one or more steps, in effect, performing a lookahead on the data. To this end, for all nodes $\mathbf{V}$ and number of steps $q = 1, \ldots, d - \ell(\mathbf{V})$, define

$$S_q(\mathbf{V}) := \max_{\substack{\{C_1,\ldots,C_r\} \\ \text{partition of } \Omega}} \sum_{i=1}^{r} S_{q-1}(C_i\mathbf{V}) \,, \qquad (13)$$

with $S_0(\mathbf{V})$ being the base case of a flat upper bound.

Proposition 4 (Lookahead bound)**.** *For all $\mathbf{V} \in \mathcal{V}$ and $q = 1, \ldots, d - \ell(\mathbf{V})$, it holds that*

$$S_*(\mathbf{V}) \leq S_q(\mathbf{V}) \leq S_0(\mathbf{V}) \,. \qquad (14)$$

Using the lookahead bound with large q does constitute a substantial computational effort. If $q = d - \ell(\mathbf{V})$, the bound matched the optimal score and, in essence, is obtained by traversing through all possible PCT subtrees. Hence, the choice of q is critical in order to obtain a trade-off between gained savings and additional invested effort in relation to the flat bound.

One possibility to cope with that issue is to dynamically increase q, i.e., to first test whether pruning on flat upper bounds S_0 is possible. If this is not the case, q is increased by one, up to the maximal value of $d - \ell(\mathbf{V})$. However, in this work we refrain from exploring this procedure in full, as in our preliminary studies one-step lookahead ($q = 1$) turned out to perform the best in the vast majority of cases.

3.6 FINAL ALGORITHM

We combine the presented ideas using pseudo code. Consider first the task of computing, for all nonempty subsets $B \subseteq \Omega$, the maximum total score over all partitions of B, in other words, the set function f' for a given set function f, as defined in Eq. 11. The procedure MAX-PART given below completes this task based on the recurrence in Eq. 12.

```
MAX-PART(f)
1  g[∅] ← 0
2  for each ∅ ⊂ B ⊆ Ω in quasi-lexicographical order
3       do g[B] ← −∞
4          for each ∅ ⊂ C ⊆ B
5               do g[B] ← max{g[B], f[C]+g[B\C]}
6  return g
```

The main algorithm, given below as procedure MAX-PCT, calls MAX-PART(f) both with exact scores and with upper bounds, as specified by the argument f. The call MAX-PCT($\mathbf{V}$) returns the optimal score $S_*(\mathbf{V})$. We thus obtain the maximum score over all PCTs of depth d by calling MAX-PCT(Λ).

```
MAX-PCT(V)
 1  score ← L(V) − K
 2  if ℓ(V) < d and score < S_0^BIC(V)
 3     then s[Ω] ← MAX-PCT(ΩV)
 4          for each ∅ ⊂ B ⊆ Ω
 5               do u[C] ← S_q^BIC(CV)
 6          u' ← MAX-PART(u)
 7          for each ∅ ⊂ B ⊆ Ω
 8               do s[C] ← −∞
 9                  if u[C] + u'[Ω \ C] > s[Ω]
10                    then s[C] ← MAX-PCT(CV)
11          s' ← MAX-PART(s)
12          score ← s'[Ω]
13  return score
```

3.7 INCORPORATING MEMOIZATION

While we omitted it for the sake of simplicity in the pseudocode, incorporating the memoization technique of Eggeling et al. (2015a) into the proposed algorithm is straightforward.

We can add a test that checks whether the index set $I(\mathbf{V})$ has already occurred with some other node at the same depth directly when entering the function. If the test is positive, the score is re-used and the rest of the function is skipped. If the test is negative, the score is stored in a hash data structure at the end of the function with the current data subset (index set) and the depth of the node as key.

4 CASE STUDIES

In the empirical part of this work, we evaluate the effect of the proposed pruning techniques using a Java-implemententation based on the Jstacs library (Grau et al., 2012). We focus on the metric of *visited nodes* in the extended PCT, which includes all created nodes, included those used only for look-ahead computations. In addition, we also measure the elapsed *running time*. We consider the problem of modeling DNA binding sites of regulatory proteins such as transcription factors, which constitutes one established application of PCTs.

4.1 DATA AND PREDICTION STUDY

A data set of DNA binding sites consists of short sequences of the same length over the alphabet $\Omega = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ that are considered to be recognized by the same DNA-binding protein. We use 25 data sets, which all show at least some degree of statistical dependence among sequence positions, from the publicly available data base JASPAR (Mathelier et al., 2013). The data sets differ in sequence length L from 10 to 21 symbols, in the sample size N from 156 to 3629 sequences, and in the strength of the signal.

To exhibit the general properties of this data, we perform a prediction experiment based on the model class and learning framework of Eggeling et al. (2014), which makes a position-specific use of PCTs. We perform a leave-one-out cross-validation for different PCT depths $d \in \{0, \ldots, 6\}$ and compute the mean log predictive probability over the test sequences from all iterations. We aggregate the resulting 7×25 mean log predictive probabilities in two different ways and visualize the results in Figure 3. First, we check for each data set, which d yields the highest predictive probability, and in case that several depths share rank one (which can happen if all subtrees below a certain depth are minimal), we note the smallest d. Second, we average the mean log predictive probabilities for each d over all 25 data sets.

We observe that all data sets contain indeed statistical dependencies to some degree, since $d = 0$ is never the best choice, and increasing d yields – on average – an increased predictive performance, though the magnitude of improvement decreases gradually. This indicates that (i) the used BIC score for PCT optimization works reasonably well in order to avoid overfitting, and (ii) the data sets have a non-trivial structure so that the optimization problem is hard.

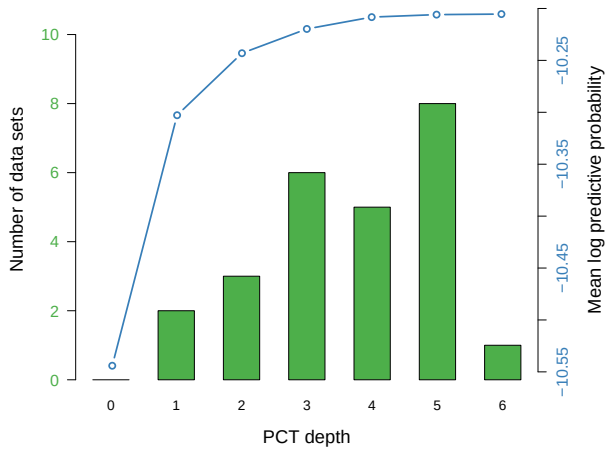

Figure 3: **Prediction study**. We show the results of a leave-one-out cross validation. Green bars indicate for how many data sets a certain PCT depth d is the smallest depth that yields an optimal prediction. The blue line shows the average predictive performance over all data sets as a function of d.

4.2 ONE DATA SET IN DETAIL

Next, we consider a single data set and investigate the effectiveness of the pruning rule in relation to the basic algorithm and to the memoization technique of Eggeling et al. (2015a). We focus on the DNA-binding protein CTCF (Kim et al., 2007), where the corresponding JASPAR data set consists of $N = 908$ sequences of length $L = 19$.

Figure 4a shows for this data set the *sequence logo* (Schneider & Stephens, 1990), which is a common visualization of the marginal distribution of the individual sequence positions. For each position, the four possible symbols are scaled relative to the marginal probability $\mathbf{p} = (p_\mathrm{A}, p_\mathrm{C}, p_\mathrm{G}, p_\mathrm{T})$ and the height of the symbol stack is scaled by $2 - \mathrm{H}(\mathbf{p})$ (which is often called *information content*), with H denoting the Shannon entropy in bits. Figure 4b shows the fraction of the visited nodes in the extended PCT of depth 5 for the pruning technique, the memoization technique, and the combination of both in relation to the basic algorithm.

We observe that pruning is effective in particular at positions where the marginal distribution of the response variable is far from uniform, with clear examples being position 10 and position 13. In fact, the correlation coefficient between the information content of the response variable and the common logarithm of fraction of visited nodes using the pruning rule in relation to the basic algorithm amounts -0.821. For memoization, a similar effect occurs, but here marginal distributions of the explanatory variables are the deciding factor: the largest saving occurs at position 14, where highly informative positions 10 and 13 are in the context.

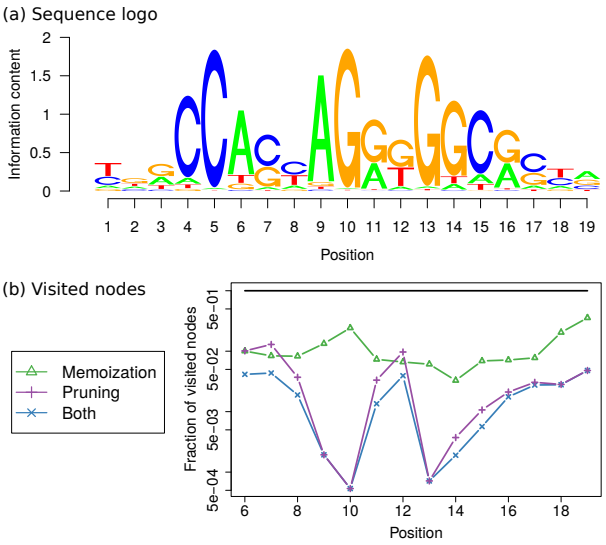

Figure 4: **Detailed result for CTCF data set.** Subfigure (a) shows the position-specific marginal nucleotide frequencies for each position in the sequence. Subfigure (b) shows the fraction of visited nodes required for PCT optimization ($d = 5$) for every sequence position $j = 6, \ldots, 19$, and for the three algorithmic variants in relation to the basic algorithm.

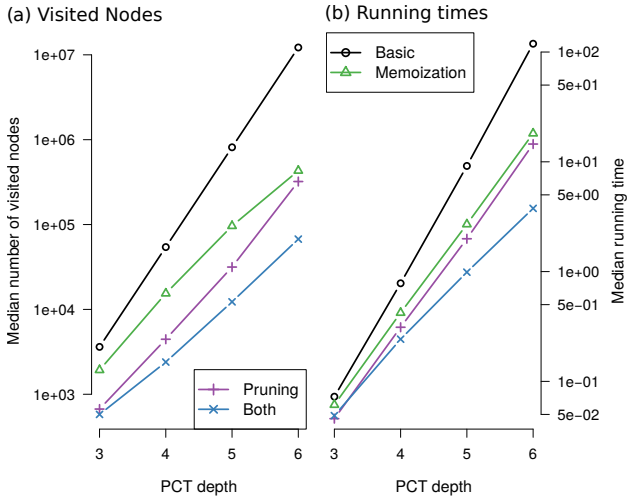

Figure 5: **Aggregated results for all four algorithmic variants.** For each data set, we average the visited nodes and running times over all sequence positions. We then plot (a) the median number of visited nodes and (b) median running times over all data sets.

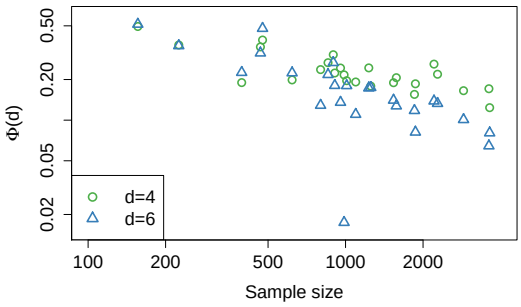

Figure 6: **Effect of sample size on savings ratio for pruning algorithm.** For all 25 data sets, for $d = \{4, 6\}$, we plot $\Phi(d)$ of Eq. 15 against the sample size N.

We summarize that the empirical performance does indeed satisfy the theoretical expectations: While memoization exploits regularities in the realizations of the explanatory variables, the pruning rules exploit regularity in the response variable, and the effect of the latter is often, though not always, greater. Combining both techniques generally resembles the sole application of pruning, albeit additional savings due to memoization do occur.

4.3 BROADER STUDY

We now take a broader view by considering all data sets. Given a PCT depth d, we (i) average the number of visited nodes and the absolute running times for each data set over all sequence positions, and (ii) take the median of average visited nodes and average running times over all data sets. The results are displayed for $d = 3, \ldots, 6$ in Figure 5, and confirm the observations made for a single data set in the previous section: Pruning outperforms memoization, yet the combination of both techniques yields the largest overall effect.

However, we observe a striking divergence between savings in terms of visited nodes in the extended PCT as displayed in Figure 5a and the improvements in terms of running times as displayed in Figure 5b. Whereas the savings in the first case are up to two orders of magnitude, the absolute running time yields a smaller improvement of roughly one order of magnitude for PCTs of $d = 5$.

In order to investigate this issue further, we measure how strong the runtime reduction deviates from the reduction in terms of visited nodes. Let $\mathrm{VN}(d)$ denote the number of visited nodes in the extended PCT for finding an optimal PCT of depth d using the pruning technique, and let $\mathrm{RT}(d)$ denote the corresponding running time. Let further $\mathrm{VN}_{\mathrm{basic}}(d)$ and $\mathrm{RT}_{\mathrm{basic}}(d)$ denote the same quantities for the basic algorithm. We define

$$\Phi(d) = \frac{\mathrm{VN}(d) \times \mathrm{RT}_{\mathrm{basic}}(d)}{\mathrm{VN}_{\mathrm{basic}}(d) \times \mathrm{RT}(d)}, \tag{15}$$

which yields a value of 1, if the savings in visited nodes are translated one-to-one into savings in running times, and a value smaller than 1, if savings in terms of visited nodes are larger. Next, we plot $\Phi(4)$ and $\Phi(6)$ for each data set against the sample size (Figure 6).

We observe a correlation between Φ and sample size: for small data sets the discrepancy between running time and visited nodes is smaller than for larger data sets, which is beneficial since the statistical efficiency of PCTs is relevant in particular for small data settings. The one outlier in the plot at sample size close to 1000 where the $\Phi(6)$ is clearly off belongs to a highly-regular data set where pruning is so effective that reading the data and initializing all data structures becomes the dominating factor, albeit requiring only few milliseconds in absolute terms. Ignoring that outlier and extrapolating Φ in Figure 6 to a hypothetical sample size of zero, it appears that savings in running times and visited nodes could match.

To explain this behavior, we have to reconsider the theoretical expectations of the running time of the basic algorithm (Section 2.2). It assumes the work to be performed in each inner node to be dominated by the alphabet partitioning problem and thus constant for a fixed Ω. For small $|\Omega|$, however, data-management operations, such as determining $I(C\mathbf{V})$ given $I(\mathbf{V})$, become a significant factor. Unlike alphabet partitioning, data-management is not equally demanding within each node: Nodes close to the root match on average more data points than nodes close to the leaves. Moreover, assuming the siblings in the extended PCT are ordered quasi-lexicographically as in the example of Figure 1a, we observe that the right half of the extended PCT (or any subtree of it) matches on average more data points than the left half. However, the subtrees of the extended PCT not traversed explicitly due to the pruning or memoization technique are predominantly the subtrees that match comparably few data points, which explains why the savings in visited nodes do not directly translate into the same saving factors for running times.

5 CONCLUDING REMARKS

We have investigated a bound-and-prune approach to finding a maximum-score parsimonious context tree (PCT). Specifically, we derived local score upper bounds for the BIC score, with an option for a few-step lookahead, and we presented two pruning rules: a stopping rule and a deletion rule.

Empirical results on DNA binding site data showed that pruning alone, which essentially exploits regularities in the response variable, is slightly more effective than the memoization technique of Eggeling et al. (2015a), which exploits regularities in the explanatory variables. While the combination of pruning and memoization runs an about order-of-magnitude faster, it partially inherits the large memory requirements of memoization; however, pruning does reduce the memory requirement, too, as a smaller number of subproblems need to be solved explicitly. These finding suggest that pruning should always be included, and memoization can be added to gain further speedups provided that memory consumption is not an issue.

While we have restricted our attention to learning PCTs and to the BIC score, we believe many of the presented ideas are applicable more generally. First, the BIC score can, in principle, be replaced by any other scoring function for which good upper bounds similar to Proposition 1 can be established. Second, the techniques should easily extend to learning decision trees with many categorical explanatory variables. Third, the bound-and-prune approach might be effective also in expediting other algorithms that are based on recursive set partitioning, like the one by Kangas et al. (2014) for learning chordal Markov networks.

Acknowledgements

The authors thank the anonymous reviewers for valuable suggestions to improve the presentation. This work was supported by the Academy of Finland, Grant 276864 "Supple Exponential Algorithms".

References

Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

Bourguignon, P.-Y. and Robelin, D. Modèles de Markov parcimonieux: sélection de modele et estimation. In *Proceedings of the 5e édition des Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM)*, 2004.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-specific independence in Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

Breiman, L., Friedman, J.H., Olshen, R., and Stone, C.J. *Classification and Regression Trees*. Wadsworth, 1984.

Bühlmann, P. and Wyner, A.J. Variable length Markov chains. *Annals of Statistics*, 27:480–513, 1999.

Chavira, M. and Darwiche, A. Compiling Bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

Chickering, D.M., Heckerman, D., and Meek, C. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings. of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.

de Campos, C.P. and Ji, Q. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

Dechter, R. and Mateescu, R. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.

Eggeling, R., Roos, T., Myllymäki, P., and Grosse, I. Robust learning of inhomogeneous PMMs. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Eggeling, R., Koivisto, M., and Grosse, I. Dealing with small data: On the generalization of context trees. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015a.

Eggeling, R., Roos, T., Myllymäki, P., and Grosse, I. Inferring intra-motif dependencies of DNA binding sites from ChIP-seq data. *BMC Bioinformatics*, 16:375, 2015b.

Friedman, N. and Goldszmidt, M. Learning Bayesian networks with local structure. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

Grau, J., Keilwagen, J., Gohr, A., Haldemann, B., Posch, S., and Grosse, I. Jstacs: A Java framework for statistical analysis and classification of biological sequences. *Journal of Machine Learning Research*, 13:1967–1971, 2012.

Kangas, K., Koivisto, M., and Niinimäki, T. Learning chordal Markov networks by dynamic programming. In *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.

Kim, T.E., Abdullaev, Z.K., Smith, A.D., Ching, K.A., Loukinov, D.I., Green, R.D., Zhang, M.Q., Lobanenkov, V.V., and Ren, B. Analysis of the vertebrate insulator protein CTCF-binding sites in the human genome. *Cell*, 128:1231–1245, 2007.

Mathelier, A., Zhao, X., Zhang, A.W., Parcy, F., Worstley-Hunt, R., Arenillas, D.J., Buchman, S., Chen, C., Chou, A., Ienasescu, H., Lim, J., Shyr, C., Tan, G., Zhou, M., Lenhard, B., Sandelin, A., and Wasserman, W.W. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Research*, 42(D1):D142–D147, 2013.

Nau, D.S., Kumar, V., and Kanal, L. General branch and bound, and its relation to A* and AO*. *Artificial Intelligence*, 23(1):29–58, 1984.

Nilsson, N.J. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.

Oliver, J. Decision graphs – an extension of decision trees. In *Proceedings of the 4th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 1993.

Rissanen, J. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

Rissanen, J. A universal data compression system. *IEEE Transactions on Information Theory*, 29(5):656–664, 1983.

Schneider, T.D. and Stephens, R.M. Sequence logos: A new way to display consensus sequences. *Nucleic Acids Research*, 18(20):6097–6100, 1990.

Schwarz, G.E. Estimating the dimension of a model. *Annals of Statistics*, 2:461–464, 1978.

Seifert, M., Gohr, A., Strickert, M., and Grosse, I. Parsimonious higher-order hidden Markov models for improved array-CGH analysis with applications to Arabidopsis thaliana. *PLOS Computational Biology*, 8(1):e1002286, 2012.

Su, J. and Zhang, H. Representing conditional independence using decision trees. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, 2005.

Teyssier, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

Tian, J. A branch-and-bound algorithm for MDL learning in Bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.

Yuan, C. and Malone, B. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.