Properties of Bayesian Dirichlet Scores to Learn Bayesian Network Structures

Cassio P. de Campos[*]
Dalle Molle Institute for Artificial Intelligence
Galleria 2, Manno 6928
Switzerland

Qiang Ji[†]
Rensselaer Polytechnic Institute
110 8th Street, Troy, NY 12180
U.S.A.

Abstract

This paper addresses exact learning of Bayesian network structure from data based on the Bayesian Dirichlet score function and its derivations. We describe useful properties that strongly reduce the computational costs of many known methods without losing global optimality guarantees. We show empirically the advantages of the properties in terms of time and memory consumptions, demonstrating that state-of-the-art methods, with the use of such properties, might handle larger data sets than those currently possible.

Introduction

A Bayesian network is a probabilistic graphical model that relies on a structured dependency among random variables to represent a joint probability distribution in a compact and efficient manner. It is composed by a directed acyclic graph (DAG) where nodes are associated to random variables and conditional probability distributions are defined for variables given their parents in the graph. Learning the graph (or structure) of these networks from data is one of the most challenging problems. For complete data, best exact known methods take exponential time on the number of variables and are applicable to small settings (around 30 variables). Approximate procedures can handle larger networks, but usually they get stuck in local maxima.

In general terms, the problem is to find the best structure (DAG) according to some score function that depends on the data (Heckerman, Geiger, and Chickering 1995). There are methods based on other (local) statistical analysis (Spirtes, Glymour, and Scheines 1993), but they follow a completely different approach. The research on this topic is active (Chickering 2002; Teyssier and Koller 2005; Tsamardinos, Brown, and Aliferis 2006), mostly focused on complete data. Most exact ideas (where it is guaranteed to find the global best scoring structure) are based on dynamic programming (Koivisto, Sood, and Chickering 2004; Singh and Moore 2005; Koivisto 2006; Silander and Myllymaki 2006; Parviainen and Koivisto 2009), which spends time and memory proportional to $n \cdot 2^n$ (n is the number of variables). Such complexity forbids the use of those methods to a couple of tens of variables, mostly because of memory consumption. Recently an exact algorithm was proposed that uses a cache to store scores and applies a branch-and-bound idea to search over all possible graphs (de Campos, Zeng, and Ji 2009), obtaining good results.

We present some properties of the problem that bring a considerable improvement on many known methods using criteria that are based on the Bayesian Dirichlet (BD) score (Cooper and Herskovits 1992), which is the most common criteria for evaluating Bayesian network structures. As we see later, the mathematical derivations are more elaborate than those recently introduced for BIC and AIC criteria (de Campos, Zeng, and Ji 2009), and the reduction in the search space and cache size are less effective when priors are strong, but still relevant. This is expected, as the BIC score is known to penalize complex graphs more than BD scores do. We show that the search space can be reduced without losing the global optimality guarantee and that the memory requirements are small in many practical cases.

Bayesian networks

A Bayesian network represents a joint probability distribution over a collection of random variables. It can be defined as a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ is a directed acyclic graph (DAG) with $V_\mathcal{G}$ a collection of n nodes associated to random variables $\mathcal{X}$ (a node per variable), and $E_\mathcal{G}$ a collection of arcs; $\mathcal{P}$ is a collection of conditional probabilities $p(X_i|\mathit{PA}_i)$ where PA_i denotes the parents of X_i in the graph (PA_i may be empty), respecting the relations of $E_\mathcal{G}$. We assume that variables are categorical. In a Bayesian network every variable is conditionally independent of its non-descendants given its parents (Markov condition).

We use uppercase letters such as X_i, X_j to represent variables (or nodes of the graph, which are used interchanged), and x_i to represent a generic state of X_i, which has state space $\Omega_{X_i} = \{x_{i1}, x_{i2}, \ldots, x_{ir_i}\}$, where $r_i = |\Omega_{X_i}| \geq 2$ is the number of categories of X_i ($|\cdot|$ is the size of a set/vector). Bold letters are used to emphasize sets or vectors. $\mathbf{x} \in \Omega_\mathbf{X} = \times_{X \in \mathbf{X}} \Omega_X$, for $\mathbf{X} \subseteq \mathcal{X}$, is an instantiation for all the variables in $\mathbf{X}$. Furthermore, $r_{\Pi_i} = |\Omega_{\Pi_i}| = \prod_{X_t \in \Pi_i} r_t$ is the number of instantiations of the parent set Π_i of X_i, and $\boldsymbol{\theta} = (\theta_{ijk})_{\forall ijk}$ is the entire vector of param-

[*]Work partially supported by the project *Ticino in Rete* and Swiss NSF grant n. 200020-116674/1.

[†]Work partially supported by a grant from the U.S. Army Research Office under grant number W911NF-06-1-0331.

eters such that $\theta_{ijk} = p(x_{ik}|\boldsymbol{\pi}_{ij})$, where $i \in \{1,\ldots,n\}$, $j \in \{1,...,r_{\Pi_i}\}$, $k \in \{1,...,r_i\}$, and $\boldsymbol{\pi}_{ij} \in \Omega_{\Pi_i}$. Because of the Markov condition, the Bayesian network represents a joint probability distribution by the expression $p(\mathbf{x}) = p(x_1,\ldots,x_n) = \prod_i p(x_i|\boldsymbol{\pi}_i)$, for every $\mathbf{x} \in \Omega_{\mathcal{X}}$, where every x_i and $\boldsymbol{\pi}_i$ agree with $\mathbf{x}$.

Given a complete multinomial dataset $D = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$, where $\mathbf{x}_u \in \Omega_{\mathcal{X}}$ is an instance of all the variables, the goal of structure learning is to find a graph $\mathcal{G}$ that maximizes a score function. In this paper, we consider the well-known Bayesian Dirichlet (BD) score, the BDe and the BDeu, and the K2 metric (Buntine 1991; Cooper and Herskovits 1992; Heckerman, Geiger, and Chickering 1995). The results refer to all criteria unless otherwise specified. As done before in the literature, we assume parameter independence and modularity (Heckerman, Geiger, and Chickering 1995).

$$BD: \quad s_D(\mathcal{G}) = \log\left(p(\mathcal{G}) \cdot \int p(D|\mathcal{G},\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}|\mathcal{G})d\boldsymbol{\theta}\right),$$

where the logarithmic is often used to simplify computations, $p(\boldsymbol{\theta}|\mathcal{G})$ is the prior of $\boldsymbol{\theta}$ for a given graph $\mathcal{G}$, assumed to be a Dirichlet with hyper-parameters $\boldsymbol{\alpha} = (\alpha_{ijk})_{\forall ijk}$ (which are assumed to be strictly positive):

$$p(\boldsymbol{\theta}|\mathcal{G}) = \prod_{i=1}^{n}\prod_{j=1}^{r_{\Pi_i}} \Gamma(\alpha_{ij}) \prod_{k=1}^{r_i} \frac{\theta_{ijk}^{\alpha_{ijk}-1}}{\Gamma(\alpha_{ijk})},$$

where $\alpha_{ij} = \sum_k \alpha_{ijk}$. Note that the hyper-parameters $(\alpha_{ijk})_{\forall ijk}$ depend on the graph $\mathcal{G}$, so a more detailed notation would be $\alpha_{ijk}^{\Pi_i}$, which unless necessary we omit.

The task is to look for the best structure $\mathcal{G}^* = \text{argmax}_{\mathcal{G}}\, s_D(\mathcal{G})$. From now on, the subscript D is omitted. We assume that there is no preference for any graph, so $p(\mathcal{G})$ is uniform and vanishes in the computations. Under the assumptions, it has been shown (Cooper and Herskovits 1992) that for multinomial distributions,

$$s(\mathcal{G}) = \log \prod_{i=1}^{n}\prod_{j=1}^{r_{\Pi_i}} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+n_{ijk})}{\Gamma(\alpha_{ijk})}, \quad (1)$$

where n_{ijk} indicates how many elements of D contain both x_{ik} and $\boldsymbol{\pi}_{ij}$, and $n_{ij} = \sum_k n_{ijk}$. As before, note that the values $(n_{ijk})_{\forall ijk}$ depend on the graph $\mathcal{G}$ (more specifically, they depend on the parent set Π_i of each X_i), so a more precise notation would be to use $n_{ijk}^{\Pi_i}$ instead of n_{ijk}. Although we avoid this heavy notation for simplicity (unless necessary in the context), we ask the reader to keep in mind that values n_{ijk} and α_{ijk} depend on the parent set of X_i.

The BDe score (Heckerman, Geiger, and Chickering 1995) assumes that $\alpha_{ijk} = \alpha^* \cdot p(\theta_{ijk}|\mathcal{G})$, where α^* is known as the Equivalent Sample Size (ESS), and $p(\theta_{ijk}|\mathcal{G})$ is the prior probability for $(x_{ik} \wedge \boldsymbol{\pi}_{ij})$ given $\mathcal{G}$. The BDeu score (Buntine 1991; Cooper and Herskovits 1992) assumes further that local priors are such that α_{ijk} becomes $\frac{\alpha^*}{r_{\Pi_i} r_i}$ and α^* is the only free hyper-parameter, while the K2 metric assumes that $\alpha_{ijk} = 1$ for all i, j, k.

An important property of all such criteria is that their functions are decomposable and can be written in terms of the local nodes of the graph, that is, $s(\mathcal{G}) = \sum_{i=1}^{n} s_i(\Pi_i)$, with $s_i(\Pi_i) =$

$$= \sum_{j=1}^{r_{\Pi_i}} \left(\log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+n_{ij})} + \sum_{k=1}^{r_i} \log \frac{\Gamma(\alpha_{ijk}+n_{ijk})}{\Gamma(\alpha_{ijk})} \right). \quad (2)$$

Equation (2) is used to compute the contribution of the node X_i with parent set Π_i to the global score of a graph.

Properties of the score functions

Local scores need to be computed many times to evaluate the candidate graphs when we look for the best graph. Because of decomposability, we can avoid to compute such functions several times by creating a cache that contains $s_i(\Pi_i)$ for each X_i and each parent set Π_i. Note that this cache may have an exponential size on n, as there are 2^{n-1} subsets of $\{X_1,\ldots,X_n\} \setminus \{X_i\}$ to be considered as parent sets. This gives a total space and time of $O(n \cdot 2^n \cdot v)$ to build the cache, where v is the worse case asymptotic time to compute the local score function at each node.[1] Instead, we describe a collection of results that are used to obtain much smaller caches in many practical cases. First, Lemma 1 is quite simple but very useful to discard elements from the cache of each node X_i. It was previously stated in (Teyssier and Koller 2005) and (de Campos, Zeng, and Ji 2009), among others.

Lemma 1. *Let X_i be a node of $\mathcal{G}'$, a candidate DAG for a Bayesian network where the parent set of X_i is Π_i'. Suppose $\Pi_i \subset \Pi_i'$ is such that $s_i(\Pi_i) > s_i(\Pi_i')$. Then Π_i' is not the parent set of X_i in the optimal DAG $\mathcal{G}^*$.*

Proof. This fact comes straightforward from the decomposability of the score functions. Take a graph $\mathcal{G}$ that differs from $\mathcal{G}'$ only on the parent set of X_i, where it has Π_i instead of Π_i'. Note that $\mathcal{G}$ is also a DAG (as $\mathcal{G}$ is a subgraph of $\mathcal{G}'$ built from the removal of some arcs, which cannot create cycles) and $s(\mathcal{G}) = \sum_{j \neq i} s_j(\Pi_j') + s_i(\Pi_i) > \sum_{j \neq i} s_j(\Pi_j') + s_i(\Pi_i') = s(\mathcal{G}')$. Hence any DAG $\mathcal{G}'$ with parent set Π_i' for X_i has a subgraph $\mathcal{G}$ with a better score than that of $\mathcal{G}'$, and thus Π_i' is not the optimal parent configuration for X_i in $\mathcal{G}^*$. $\square$

Unfortunately Lemma 1 does not tell us anything about supersets of Π_i', that is, we still need to compute scores for all the possible parent sets and later verify which of them can be removed. This would still leave us with $\Omega(n \cdot 2^n \cdot v)$ time and space requirements (although the space would be reduced after applying the lemma). In the follow we describe how to avoid all such computations. First note that, in the BD score, $s_i(\Pi_i) =$

$$= \sum_{j \in J_i} \left(\log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+n_{ij})} + \sum_{k \in K_{ij}} \log \frac{\Gamma(\alpha_{ijk}+n_{ijk})}{\Gamma(\alpha_{ijk})} \right),$$

[1]Note that the time to compute a single local score might be large depending on the number of parents.

where $J_i \doteq J_i^{\Pi_i} = \{1 \leq j \leq r_{\Pi_i} : \; n_{ij} \neq 0\}$, because $n_{ij} = 0$ implies that all terms cancel each other. In the same manner, $n_{ijk} = 0$ implies that the terms of the internal summation cancel out, so let $K_{ij} \doteq K_{ij}^{\Pi_i} = \{1 \leq k \leq r_i : \; n_{ijk} \neq 0\}$ be the indices of the categories of X_i such that $n_{ijk} \neq 0$. Let $K_i^{\Pi_i} \doteq \cup_j K_{ij}^{\Pi_i}$ be a vector with all indices corresponding to non-zero counts for Π_i (note that the symbol $\cup$ was used as a concatenation of vectors, as $K_i^{\Pi_i}$ may have repetitions). Note that the counts n_{ijk} (and consequently $n_{ij} = \sum_k n_{ijk}$) are completely defined if we know the parent set Π_i. Rewrite the score as follows:

$$s_i(\Pi_i) = \sum_{j \in J_i} \left(f(K_{ij}, (\alpha_{ijk})_{\forall k}) + g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) \right),$$

with $f(K_{ij}, (\alpha_{ijk})_{\forall k}) = \log \Gamma(\alpha_{ij}) - \sum_{k \in K_{ij}} \log \Gamma(\alpha_{ijk})$ and $g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) =$

$$- \log \Gamma(\alpha_{ij} + n_{ij}) + \sum_{k \in K_{ij}} \log \Gamma(\alpha_{ijk} + n_{ijk}).$$

We do not need K_{ij} as argument of $g(\cdot)$ because the set of non-zero n_{ijk} is known from the counts $(n_{ijk})_{\forall k}$ that are already available as arguments of $g(\cdot)$. To achieve the desired theorem that will be able to reduce the computational time to build the cache, some intermediate results are necessary.

Lemma 2. *For any Π_i, $(\alpha_{ijk})_{\forall ijk} > 0$, and integers $(n_{ijk})_{\forall ijk} \geq 0$, we have that $g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) \leq c$ if $n_{ij} \geq 1$, where $c = -\log \Gamma(v) \approx 0.1214$ and $v = \operatorname{argmax}_{x>0} - \log \Gamma(x) \approx 1.4616$. Furthermore, $g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) \leq -\log \alpha_{ij} + \log \alpha_{ijk} - f(K_{ij}, (\alpha_{ijk})_{\forall k})$ if $|K_{ij}| = 1$.*

Proof. We use the relation $\Gamma(x + \sum_k a_k) \geq \Gamma(x+1) \prod_k \Gamma(a_k)$, for $x \geq 0$, $\forall_k a_k \geq 1$ and $\sum_k a_k \geq 1$ (note that it is valid even if there is a single element in the summation), which comes from the Beta function inequality:

$$\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \leq \frac{x+y}{xy} \implies \Gamma(x+1)\Gamma(y+1) \leq \Gamma(x+y+1),$$

where $x, y > 0$, and then we use $y + 1 = \sum_t a_t$ (which is possible because $\sum_t a_t > 1$ and thus $y > 0$), to obtain:

$$\Gamma(x + \sum_t a_t) \geq \Gamma(x+1)\Gamma(\sum_t a_t) \geq \Gamma(x+1) \prod_t \Gamma(a_t), \tag{3}$$

(the last step is due to $a_t \geq 1$ for all t, so the same relation of the Beta function can be overall applied, because $\Gamma(x+1)\Gamma(y+1) \leq \Gamma(x+y+1) \leq \Gamma(x+1+y+1)$). Now,

$$\frac{\Gamma(\alpha_{ij} + n_{ij})}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} = \frac{\Gamma(\sum_{1 \leq k \leq r_i} (\alpha_{ijk} + n_{ijk}))}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} =$$

$$= \frac{\Gamma(\sum_{k \notin K_{ij}} \alpha_{ijk} + \sum_{k \in K_{ij}} (\alpha_{ijk} + n_{ijk}))}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} \geq$$

$\geq \Gamma(1 + \sum_{k \notin K_{ij}} \alpha_{ijk})$, by renaming $x = \sum_{k \notin K_{ij}} \alpha_{ijk}$ and $a_k = \alpha_{ijk} + n_{ijk}$ (we have that $\sum_{k \in K_{ij}} (\alpha_{ijk} + n_{ijk}) \geq n_{ij} \geq 1$ and each $a_k \geq 1$). Thus $g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) =$

$$-\log \frac{\Gamma(\alpha_{ij} + n_{ij})}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} \leq -\log \Gamma(1 + \sum_{k \notin K_{ij}} \alpha_{ijk}).$$

Because $v = \operatorname{argmax}_{x>0} - \log \Gamma(x)$ and $c = -\log \Gamma(v)$,

$$-\log \Gamma(1 + \sum_{k \notin K_{ij}} \alpha_{ijk}) \leq c.$$

If $|K_{ij}| = 1$, then let $K_{ij} = \{k\}$. We have $n_{ij} \geq 1$ and

$$g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) = -\log \frac{\Gamma(\alpha_{ij} + n_{ij})}{\Gamma(\alpha_{ijk} + n_{ij})}$$

$$= -\log \left(\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ijk})} \prod_{t=0}^{n_{ij}-1} \frac{(\alpha_{ij} + t)}{(\alpha_{ijk} + t)} \right)$$

$$= -f(K_{ij}, (\alpha_{ijk})_{\forall k}) - \log \frac{\alpha_{ij}}{\alpha_{ijk}} - \sum_{t=1}^{n_{ij}-1} \log \frac{(\alpha_{ij} + t)}{(\alpha_{ijk} + t)}$$

$$\leq -\log \alpha_{ij} + \log \alpha_{ijk} - f(K_{ij}, (\alpha_{ijk})_{\forall k}).$$

□

Lemma 3. *For any Π_i, $(\alpha_{ijk})_{\forall ijk} > 0$, and integers $(n_{ijk})_{\forall ijk} \geq 0$, we have that $g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) \leq 0$ if $n_{ij} \geq 2$.*

Proof. If $n_{ij} \geq 2$, we use the relation $\Gamma(x + \sum_k a_k) \geq \Gamma(x+2) \prod_k \Gamma(a_k)$, for $x \geq 0$, $\forall_k a_k \geq 1$ and $\sum_k a_k \geq 2$. This inequality is obtained in the same way as in Lemma 2, but using a tighter Beta function bound:

$$\mathcal{B}(x,y) \leq \frac{x+y}{xy} \left(\frac{(x+1)(y+1)}{x+y+1} \right)^{-1} \Rightarrow$$

$$\Rightarrow \Gamma(x+2)\Gamma(y+2) \leq \Gamma(x+y+2),$$

and the relation follows by using $y + 2 = \sum_t a_t$ and the same derivation as before. Now,

$$\frac{\Gamma(\alpha_{ij} + n_{ij})}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} = \frac{\Gamma(\sum_{1 \leq k \leq r_i} (\alpha_{ijk} + n_{ijk}))}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} =$$

$$= \frac{\Gamma(\sum_{k \notin K_{ij}} \alpha_{ijk} + \sum_{k \in K_{ij}} (\alpha_{ijk} + n_{ijk}))}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} \geq$$

$\geq \Gamma(2 + \sum_{k \notin K_{ij}} \alpha_{ijk})$, obtained by renaming $x = \sum_{k \notin K_{ij}} \alpha_{ijk}$ and $a_k = \alpha_{ijk} + n_{ijk}$, as we know that $\sum_{k \in K_{ij}} (\alpha_{ijk} + n_{ijk}) \geq n_{ij} \geq 2$ and each $a_k \geq 1$. Finally,

$$g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) = -\log \frac{\Gamma(\alpha_{ij} + n_{ij})}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} \leq$$

$$\leq -\log \Gamma(2 + \sum_{k \notin K_{ij}} \alpha_{ijk}) \leq 0,$$

because $\Gamma(2 + \sum_{k \notin K_{ij}} \alpha_{ijk}) \geq 1$. □

Lemma 4. *Given two parent sets Π_i^0 and Π_i for a node X_i such that $\Pi_i^0 \subset \Pi_i$, if $s_i(\Pi_i^0) >$*

$$\sum_{\substack{j \in J_i^{\Pi_i}: \\ |K_{ij}^{\Pi_i}| \geq 2}} f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + \sum_{\substack{j \in J_i^{\Pi_i}: \\ |K_{ij}^{\Pi_i}| = 1}} \log \frac{\alpha_{ijk'}^{\Pi_i}}{\alpha_{ij}^{\Pi_i}}, \tag{4}$$

then Π_i is not the optimal parent set for X_i.

Proof. Using the results of Lemmas 2 and 3, $s_i(\Pi_i) =$

$$\sum_{j \in J_i} \Big(f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + g((n_{ijk}^{\Pi_i})_{\forall k}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) \Big) \leq$$

$$\sum_{j \in J_i \colon |K_{ij}^{\Pi_i}| \geq 2} \Big(f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + g((n_{ijk}^{\Pi_i})_{\forall k}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) \Big)$$

$$+ \sum_{j \in J_i^{\Pi_i} \colon |K_{ij}^{\Pi_i}| = 1} \Big(-\log \alpha_{ij}^{\Pi_i} + \log \alpha_{ijk'}^{\Pi_i} \Big) \leq$$

$$\sum_{j \in J_i^{\Pi_i} : |K_{ij}^{\Pi_i}| \geq 2} f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + \sum_{j \in J_i^{\Pi_i} : |K_{ij}^{\Pi_i}| = 1} \log \frac{\alpha_{ijk'}^{\Pi_i}}{\alpha_{ij}^{\Pi_i}},$$

which by the assumption of this lemma, is less than $s_i(\Pi_i^0)$. Thus, we conclude that the parent set Π_i^0 has better score than Π_i, and the desired result follows from Lemma 1. $\square$

Lemma 5. *Given the BDeu score, $(\alpha_{ijk})_{\forall ijk} > 0$, and integers $(n_{ijk})_{\forall ijk} \geq 0$ such that $\alpha_{ij} \leq 0.8349$ and $|K_{ij}| \geq 2$ for a given j, then $f(K_{ij}, (\alpha_{ijk})_{\forall k}) \leq -|K_{ij}| \cdot \log r_i$.*

Proof. Using $\alpha_{ijk} \leq \alpha_{ij} \leq 0.8349$ (for all k), we have

$$f(K_{ij}, (\alpha_{ijk})_{\forall k}) = \log \Gamma(\alpha_{ij}) - |K_{ij}| \log \Gamma(\frac{\alpha_{ij}}{r_i}) =$$

$$\log \Gamma(\alpha_{ij}) - |K_{ij}| \log \Gamma(\frac{\alpha_{ij}}{r_i} + 1) + |K_{ij}| \log \frac{\alpha_{ij}}{r_i} =$$

$$\log \Gamma(\alpha_{ij}) - |K_{ij}| \log \frac{\Gamma(\frac{\alpha_{ij}}{r_i} + 1)}{\alpha_{ij}} - |K_{ij}| \log r_i =$$

$$|K_{ij}| \log \frac{\Gamma(\alpha_{ij})^{1/|K_{ij}|} \alpha_{ij}}{\Gamma(\frac{\alpha_{ij}}{r_i} + 1)} - |K_{ij}| \log r_i.$$

Now, $\Gamma(\alpha_{ij})^{1/|K_{ij}|} \alpha_{ij} \leq \Gamma(\frac{\alpha_{ij}}{r_i} + 1)$, because $r_i \geq 2$, $|K_{ij}| \geq 2$ and $\alpha_{ij} \leq 0.8349$. It is interesting to point out that 0.8349 is in fact a bound for α_{ij} that ensures this last inequality to hold when $r_i = |K_{ij}| = 2$, which is the worst case scenario (greater values of them make the left-hand side decrease and the right-hand side increase). $\square$

Theorem 1. *Given the BDeu score and two parent sets Π_i^0 and Π_i for a node X_i such that $\Pi_i^0 \subset \Pi_i$ and $\alpha_{ij}^{\Pi_i} \leq 0.8349$ for every j, if $s_i(\Pi_i^0) > -|K_i^{\Pi_i}| \log r_i$ then neither Π_i nor any superset $\Pi_i' \supset \Pi_i$ are optimal parent sets for X_i.*

Proof. We have that $s_i(\Pi_i^0) > -|K_i^{\Pi_i}| \log r_i =$

$$\sum_{j \in J_i^{\Pi_i} \colon |K_{ij}^{\Pi_i}| \geq 2} -|K_{ij}^{\Pi_i}| \log r_i + \sum_{j \in J_i^{\Pi_i} \colon |K_{ij}^{\Pi_i}| = 1} - \log r_i,$$

which by Lemma 5 is greater than or equal to

$$\sum_{j \in J_i^{\Pi_i} \colon |K_{ij}^{\Pi_i}| \geq 2} f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + \sum_{j \in J_i^{\Pi_i} \colon |K_{ij}^{\Pi_i}| = 1} - \log r_i.$$

Now, Lemma 4 suffices to show that Π_i is not a optimal parent set. To show the result for any superset $\Pi_i' \supset \Pi_i$, we just have to note that $|K_i^{\Pi_i'}| \geq |K_i^{\Pi_i}|$ (because the overall number of non-zero counts can only increase when we include more parents), and $\alpha_{ij'}^{\Pi_i'}$ (for all j') are all less than 0.8349 (because the αs can only decrease when more parents are included), thus we can apply the very same reasoning. $\square$

Theorem 1 provides a bound to discard parent sets without even inspecting them because of the non-increasing monotonicity of the employed bounding function when we increase the number of parents. Thus, the idea is to check the validity of Theorem 1 every time the score of a parent set Π_i of X_i is about to be computed by taking the best score of any subsets and testing it against the theorem. Whenever possible, we discard Π_i and do not even look into all its supersets. This result allows us to stop computing scores much earlier than previous results do (Singh and Moore 2005), reducing the number of computations to build and store the cache. Note that the constraint $\alpha_{ij} \leq 0.8349$ is not too restrictive, because as parent sets grow, as ESS is divided by larger numbers (it is an exponential decrease of the αs). Hence, the values α_{ij} become quickly below the threshold. Π_i is also checked against Lemma 1 (which is stronger in the sense that instead of a bounding function, the actual scores are directly compared). However it cannot help us to avoid analyzing the superset of Π_i. As we see in the experiments, the practical size of the cache after the application of the properties is small even for considerably large networks, and both Lemma 1 and Theorem 1 help reducing the cache size, while Theorem 1 also help to reduce computations.

Experiments

We perform experiments to show the benefits of the reduced cache and search space.[2] We use data sets available at the UCI repository (Asuncion and Newman 2007). Lines with missing data are removed and continuous variables are discretized over the mean into binary variables. The data sets are: *adult* (15 variables and 30162 instances), *breast* (10 variables and 683 instances), *car* (7 variables and 1728 instances) *letter* (17 variables and 20000 instances), *lung* (57 variables and 27 instances), *mushroom* (23 variables and 1868 instances), *nursery* (9 variables and 12960 instances), Wisconsin Diagnostic Breast Cancer or *wdbc* (31 variables and 569 instances), *zoo* (17 variables and 101 instances). The number of categories per variables varies from 2 to dozens in some cases (we refer to UCI for further details).

Table 1 presents the time in seconds (first block) and number of steps in local score evaluations (second block) for the cache construction, after applying Lemma 1 and Theorem 1. The last block shows the percentage of the full cache that was discarded by each of them. For instance, the cell "56; 29" indicates that 56% of the full cache was discarded by applying Lemma 1, while 29% was not even evaluated (because of Theorem 1). Each column presents the results for a distinct data set. In different lines we show results for ESS equals to 0.1, 1, 10 and 100. Note that discarding elements by Lemma 1 does not reduce the number of steps to build the

[2]The software is available online through the web address *http://www.ecse.rpi.edu/∼cvrl/structlearning.html*

Table 1: Time and number of steps (local score evaluations) used to build the cache. The last block contains the percentage of the potential full cache that was discarded by applying *Lemma1* and *Theorem1*, respectively (only *Theorem1* reduces the number of steps). Results for BDeu score with ESS varying from 0.1 to 100 are presented.

	ESS	adult	breast	car	letter	lung	mushroom	nursery	wdbc	zoo
	0.1	89.3	0.0	0.0	429.4	2056.3	357.9	0.7	2891.8	1.7
Time	1	91.6	0.0	0.0	440.4	1398.7	278.7	0.7	2692.7	1.7
(in seconds)	10	91.6	0.0	0.0	438.1	1098.0	268.9	0.7	2763.7	1.7
	100	53.8	0.0	0.0	93.3	1428.8	318.8	0.6	3272.8	0.8
	0.1	$2^{17.4}$	$2^{10.5}$	$2^{8.8}$	$2^{20.1}$	$2^{30.8}$	$2^{24.0}$	$2^{11.2}$	$2^{27.9}$	$2^{19.8}$
Number of	1	$2^{17.4}$	$2^{10.5}$	$2^{8.8}$	$2^{20.1}$	$2^{30.2}$	$2^{23.6}$	$2^{11.2}$	$2^{27.8}$	$2^{19.7}$
Steps	10	$2^{17.4}$	$2^{10.4}$	$2^{8.8}$	$2^{20.1}$	$2^{29.8}$	$2^{23.5}$	$2^{11.2}$	$2^{27.9}$	$2^{19.6}$
	100	$2^{16.5}$	$2^{10.4}$	$2^{8.8}$	$2^{17.9}$	$2^{30.2}$	$2^{23.7}$	$2^{11.1}$	$2^{28.0}$	$2^{17.9}$
Worst-case		$2^{17.9}$	$2^{12.3}$	$2^{8.8}$	$2^{20.1}$	$2^{31.1}$	$2^{26.5}$	$2^{11.2}$	$2^{28.4}$	$2^{20.1}$
% of reduction	0.1	56; 29	15; 72	95; 0	99; 0	61; 22	8; 83	97; 0	69; 27	48; 20
implied by *Lemma1*;	1	56; 29	14; 72	94; 0	99; 0	38; 46	7; 87	97; 0	65; 33	47; 23
and % of reduction	10	56; 29	13; 74	86; 0	97; 0	24; 59	6; 88	95; 0	65; 32	41; 30
obtained from *Theorem1*	100	31; 63	6; 73	34; 0	7; 77	24; 49	7; 86	67; 4	68; 25	9; 77

cache, because they needed to be processed anyway, while Theorem 1 does imply in a smaller number of steps. The line *worst-case* presents the number of steps to build the cache without using Theorem 1. As we see through the log-scale in which they are presented, the reduction in number of steps has not been exponential, but still computational significant in many cases. For instance, all the experiments were run using less than 2GB of memory. Even a factor of five could make hard to run it because of the memory usage.

The benefits of the application of these results imply in performance gain for many algorithms in the literature to learn Bayesian network structures, as long as they only need to work over the (already precomputed) small cache. In Table 2 we present the final cache characteristics, where we find the most attractive results, for instance, the small cache sizes when compared to the worst case. The first block contains the maximum number of parents per node (averaged over the nodes). The worst-case is the total number of nodes in the data set minus one, apart from *lung* (where we have set a limit of at most six parents) and *wdbc* (with at most eight parents). The second block shows the cache size for each data set and distinct values of ESS. We also show the results of the BIC score and the worst-case values for comparison. We see that the actual cache size is orders of magnitude smaller than the worst case situation. It is also possible to analyze the search space reduction implied by these results by looking the implications to the search space of structure learning. We must point out that by search space we mean all the possible combinations of parent sets for all the nodes. Eventually some of these combinations are not DAGs, but are still being counted. However, there are two considerations: (i) the precise counting problem is harder to solve (in order to give the exact search space size), and (ii) most structure learning algorithms run over more than only DAGs, because they need to look at the graphs (and thus combinations of parents) to decide if they are acyclic or not. In these cases, the actual search space is not simply the set of possible DAGs, even though the final solution will be a DAG.

An expected but important point to emphasize is the correlation of the prior with the time and memory to build the cache. As larger ESS (and thus the prior towards the uniform) as slower and more memory consuming is the method. That is explained by the fact that smoothing the different parent sets by the stronger prior makes harder to see large differences in scores, and consequently the properties that would reduce the cache size are less effective. The two largest data sets in terms of number of variables (*lung* and *wdbc*) were impossible to be processed without setting up other limits such as maximum number of parents or maximum number of free parameters in the node (we have not used any limit for the other data sets). We used an upper limit of six parents per node for *lung* and eight for *wdbc*. This situation deserves further study so as to clarify whether it is possible to run these computations on large data sets and large ESS. It might be necessary to find tighter bounds if at all possible, that is, stronger results than Theorem 1 to discard unnecessary score evaluations earlier in the computations.

Conclusions

This paper describes novel properties of the Bayesian Dirichlet score functions to learn Bayesian network structure from data. Such properties allow the construction of a cache with all possible local scores of nodes and their parents with reduced memory consumption, which can later be used by searching algorithms. For instance, memory consumption was a bottleneck for some algorithms in the literature, see for example (Parviainen and Koivisto 2009). This implies in a considerable reduction of the search space of graphs without losing the global optimal structure, that is, it is ensured that the overall best graph remains in the reduced space. In fact the reduced memory and search space potentially benefits most structure learning methods in the literature, for example dynamic programming, branch and bound, some types of local search, simulated annealing, genetic programming, etc, and it makes possible to the very same algorithms to find the global solution in instances before not handled, just by working over the reduced space.

Table 2: Final cache characteristics: maximum number of parents (average by node), actual cache size, and (approximate) search space implied by the cache. Worst-cases are presented for comparison (those marked with a star are computed using the constraint on the number of parents that was applied to *lung* and *wdbc*). Results with ESS from 0.1 to 100 are presented.

	ESS	adult	breast	car	letter	lung	mushroom	nursery	wdbc	zoo
	0.1	2.1	1.0	0.7	4.5	0.1	4.1	1.2	1.3	1.4
Maximum Number	1	2.4	1.0	1.0	5.2	0.4	4.4	1.7	1.7	1.9
of Parents	10	3.3	1.0	1.9	5.9	3.0	4.8	2.1	3.1	3.4
	100	5.9	2.0	3.1	6.0	6.0	7.0	3.2	6.0	6.0
Worst-case		14.0	9.0	6.0	16.0	6.0*	22.0	8.0	8.0*	16.0
	0.1	$2^{4.2}$	$2^{1.5}$	$2^{1.1}$	$2^{8.2}$	$2^{0.2}$	$2^{8.5}$	$2^{1.9}$	$2^{3.6}$	$2^{3.3}$
Final Size	1	$2^{4.8}$	$2^{1.9}$	$2^{1.6}$	$2^{9.0}$	$2^{0.8}$	$2^{8.9}$	$2^{2.4}$	$2^{4.9}$	$2^{4.4}$
of the Cache	10	$2^{6.3}$	$2^{3.3}$	$2^{3.0}$	$2^{10.5}$	$2^{10.7}$	$2^{9.8}$	$2^{3.5}$	$2^{12.1}$	$2^{8.9}$
	100	$2^{9.1}$	$2^{5.5}$	$2^{5.4}$	$2^{13.3}$	$2^{19.2}$	$2^{13.5}$	$2^{6.2}$	$2^{19.6}$	$2^{13.1}$
	BIC	$2^{9.3}$	$2^{4.7}$	$2^{4.5}$	$2^{15.3}$	$2^{11.5}$	$2^{13.0}$	$2^{5.6}$	$2^{12.9}$	$2^{10.9}$
Worst-case ($n2^{n-1}$)		$2^{17.9}$	$2^{12.3}$	$2^{8.8}$	$2^{20.1}$	$2^{31.1*}$	$2^{26.5}$	$2^{11.2}$	$2^{28.4*}$	$2^{20.1}$
	0.1	$2^{54.1}$	$2^{13.3}$	$2^{6.3}$	$2^{129.0}$	$2^{8.2}$	$2^{175.7}$	$2^{11.6}$	$2^{90.3}$	$2^{39.3}$
Implied Search	1	$2^{62.1}$	$2^{17.1}$	$2^{8.3}$	$2^{144.8}$	$2^{33.1}$	$2^{186.0}$	$2^{15.4}$	$2^{132.7}$	$2^{60.3}$
Space (approx.)	10	$2^{91.6}$	$2^{33.2}$	$2^{20.6}$	$2^{176.1}$	$2^{612.0}$	$2^{221.8}$	$2^{27.3}$	$2^{375.1}$	$2^{150.7}$
	100	$2^{136.1}$	$2^{55.2}$	$2^{37.7}$	$2^{226.1}$	$2^{1092.9}$	$2^{310.5}$	$2^{55.9}$	$2^{606.1}$	$2^{221.8}$
	BIC	2^{71}	2^{23}	2^{10}	2^{188}	2^{330}	2^{180}	2^{17}	2^{216}	2^{111}
Worst-case		2^{210}	2^{90}	2^{42}	2^{272}	2^{1441*}	2^{506}	2^{72}	2^{727*}	2^{272}

We show through experiments with public data sets that requirements of memory are considerably smaller., as well as the resulting reduced search space.

There is certainly much further to be done. The most important question is whether the bound of Theorem 1 can be improved or not. We are actively working on this question. Furthermore, the experimental analysis can be extended to further clarify the understanding of the problem, for instance how the ESS affects the results. The comparison of structures and what define them to be good is an important topic. For example, accuracy of the generated networks can be evaluated with real data. On the other hand, that does not ensure that we are finding the true links of the underlying structure, but a somehow similar graph that produces a close joint distribution. For that, one could use generated data and compare the structures against the one data were generated from it. Besides that, considerably large data sets will hardly be handled by exact methods, so an study on how the properties may help fast approximate methods is also a desired goal.

References

Asuncion, A., and Newman, D. 2007. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Buntine, W. 1991. Theory refinement on Bayesian networks. In *Conf. on Uncertainty in AI*, 52–60.

Chickering, D. M. 2002. Optimal structure identification with greedy search. *J. of Machine Learning Research* 3:507–554.

Cooper, G. F., and Herskovits, E. 1992. A bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9:309–347.

de Campos, C. P.; Zeng, Z.; and Ji, Q. 2009. Structure learning of Bayesian networks using constraints. In *Int. Conf. on Machine Learning*, 113–120.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20(3):197–243.

Koivisto, M.; Sood, K.; and Chickering, M. 2004. Exact bayesian structure discovery in bayesian networks. *J. of Machine Learning Research* 5:549–573.

Koivisto, M. 2006. Advances in exact bayesian structure discovery in bayesian networks. In *Conf. on Uncertainty in AI*, 241–248.

Parviainen, P., and Koivisto, M. 2009. Exact structure discovery in bayesian networks with less space. In *Conf. on Uncertainty in AI*, 2009.

Silander, T., and Myllymaki, P. 2006. A simple approach for finding the globally optimal bayesian network structure. In *Conf. on Uncertainty in AI*, 445–452.

Singh, A. P., and Moore, A. W. 2005. Finding optimal bayesian networks by dynamic programming. Technical report, Carnegie Mellon University. CMU-CALD-05-106.

Spirtes, P.; Glymour, C.; and Scheines, R. 1993. *Causation, Prediction and Search*. Springer-Verlag.

Teyssier, M., and Koller, D. 2005. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Conf. on Uncertainty in AI*, 584–590.

Tsamardinos, I.; Brown, L. E.; and Aliferis, C. 2006. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning* 65(1):31–78.