New Complexity Results for MAP in Bayesian Networks

Cassio P. de Campos
Dalle Molle Institute for Artificial Intelligence
Galleria 2, Manno-Lugano, Switzerland
cassio@idsia.ch

Abstract

This paper presents new results for the (partial) maximum a posteriori (MAP) problem in Bayesian networks, which is the problem of querying the most probable state configuration of some of the network variables given evidence. It is demonstrated that the problem remains hard even in networks with very simple topology, such as binary polytrees and simple trees (including the Naive Bayes structure), which extends previous complexity results. Furthermore, a Fully Polynomial Time Approximation Scheme for MAP in networks with bounded treewidth and bounded number of states per variable is developed. Approximation schemes were thought to be impossible, but here it is shown otherwise under the assumptions just mentioned, which are adopted in most applications.

1 Introduction

A Bayesian network (BN) 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. One of the hardest inference problems in BNs is the *maximum a posteriori* (or MAP) problem, where one looks for states of some variables that maximize their joint probability, given some other variables as evidence (there may exist variables that are neither queried nor part of the evidence). This problem is known to be NP^{PP}-complete in the general case and NP-complete for polytrees [Park and Darwiche, 2004]. Thus, algorithms usually take large amount of time to solve MAP even in small networks. Approximating MAP in polytrees is also NP-hard. However, such hardness results are derived for networks with a large number of states per variable, which is not the most common situation in many practical problems. In this paper, a bounded number of states per variable is considered. It is proven that the problem remains hard even in binary polytrees and simple trees, using reductions from both the *satisfiability* and the *partition* problems, but it is also shown that there is a *Fully Polynomial Time Approximation Scheme* (FPTAS) whenever treewidth and number of states are bounded, so one may expect fast algorithms for MAP with a small approximation error under such assumptions. Fast algorithms for MAP may imply in fast algorithms for other related problems, for example inferences in decision networks and influence diagrams, besides the great interest in the MAP problem itself. Hence, this paper makes important steps in these directions.

2 Background

The reader is assumed to be familiar with basic notions of complexity theory and approximation algorithms (for example, see [Garey and Johnson, 1979; Vazirani, 2001]) and basic concepts of Bayesian networks [Pearl, 1988].

Definition 1 *A Bayesian network (BN) $\mathcal{N}$ is a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} = (\mathbf{V}_\mathcal{G}, \mathbf{E}_\mathcal{G})$ is a directed acyclic graph with nodes $\mathbf{V}_\mathcal{G}$ associated (in a one-to-one mapping) to random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ over discrete domains $\{\Omega_{X_1}, \ldots, \Omega_{X_n}\}$ and $\mathcal{P}$ is a collection of probability values $p(x_i|\pi_{X_i}) \in \mathcal{Q}$,[1] with $\sum_{x_i \in \Omega_{X_i}} p(x_i|\pi_{X_i}) = 1$, where $x_i \in \Omega_{X_i}$ is a category or state of X_i and $\pi_{X_i} \in \times_{X \in PA_{X_i}} \Omega_X$ a complete instantiation for the parents PA_{X_i} of X_i in $\mathcal{G}$. Furthermore, every variable is conditionally independent of its non-descendant non-parents given its parents.*

The joint probability distribution represented by a BN $(\mathcal{G}, \mathcal{X}, \mathcal{P})$ is obtained by $p(\mathbf{x}) = \prod_i p(x_i|\pi_{X_i})$, where $\mathbf{x} \in \Omega_\mathcal{X}$ and all states x_i, π_{X_i} (for every i) agree with $\mathbf{x}$. Now we introduce some notation. Singletons $\{X_i\}$ and $\{x_i\}$ are respectively denoted as X_i and x_i. Nodes of the graph and their associated random variables are used interchangeably. Uppercase letters are used for random variables and lowercase letters for their corresponding states. Bold letters are employed for vectors/sets. $z(\mathbf{X})$ denotes the product of the cardinality of the variables $\mathbf{X} \subseteq \mathcal{X}$, that is, $z(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} |\Omega_{X_i}|$ (with $z(\emptyset) = 1$). The input size $\text{S}(\mathcal{N})$ of a BN is given by the length of the bit string to specify all the local conditional probability distributions and the structure to describe the graph. Note that $\text{S}(\mathcal{N}) \geq n$ and $\text{S}(\mathcal{N}) \geq z_{max} = \max_{X_i \in \mathcal{X}} z(X_i)$.

The belief updating (BU) problem concerns the computation of $p(\mathbf{x}|\mathbf{e})$, for $\mathbf{x} \in \Omega_\mathbf{X}$ and $\mathbf{e} \in \Omega_\mathbf{E}$, with $\mathbf{X} \cup \mathbf{E} \subseteq \mathcal{X}$ and $\mathbf{X} \cap \mathbf{E} = \emptyset$. The complexity of solving BU is tightly related to the minimum treewidth of the network, or maximum number

[1] $\mathcal{Q}$ denotes the non-negative rational numbers defined by fractions of integers.

of variables in a single node of the most compact tree decomposition (for example, see [Darwiche, 2009]). In spite of that, some particular BNs deserve attention: $\mathcal{N} = (\mathcal{G}, \mathcal{X}, \mathcal{P})$ is called a *polytree* if the subjacent graph (dropping the direction of the arcs) of $\mathcal{G}$ has no cycles. A polytree $\mathcal{N}$ is further called a *tree* if each node of $\mathcal{G}$ has at most one parent. The treewidth of a tree is one. For trees and polytrees, BU is solvable in polynomial time [Pearl, 1988]. This is also true for any network of bounded treewidth. Finding a tree decomposition of *minimum treewidth* is hard, but it is enough to know a bound for the treewidth (which is not part of the input) to obtain it in polynomial time [Bodlaender, 1993].

The MAP problem is to find an instantiation $\mathbf{x}_{\text{opt}} \in \Omega_{\mathcal{X}^{\text{map}}}$, with $\mathcal{X}^{\text{map}} \subseteq \mathcal{X} \setminus \mathbf{E}$, such that its probability is maximized:

$$\mathbf{x}_{\text{opt}} = \operatorname*{argmax}_{\mathbf{x} \in \Omega_{\mathbf{X}}} p(\mathbf{x}|\mathbf{e}) = \operatorname*{argmax}_{\mathbf{x} \in \Omega_{\mathbf{X}}} p(\mathbf{x}, \mathbf{e}), \qquad (1)$$

because $p(\mathbf{e})$ (assumed to be non-zero[2]) is a constant with respect to the maximization. It is known that MAP queries are harder than BU queries (under the assumptions that P$\neq$NP and PP$\neq$NP$^{\text{PP}}$). It is proven that the general MAP problem is NP$^{\text{PP}}$-complete [Park and Darwiche, 2004]. However, such proof assumes a general case of the problem, while many practical BNs have some structural properties that might alleviate the complexity. Two of them are very important with respect to the complexity of the problem: the cardinality of the variables involved in the network and the minimum treewidth. The latter is considered in [Park and Darwiche, 2004], and the problem is shown to be NP-complete and not approximable by a polynomial approximation scheme even if the treewidth is bounded. In this paper both treewidth and cardinality of the network are exploited, which allows us to go further in the analysis.

Definition 2 *Given a BN $\mathcal{N} = (\mathcal{G}, \mathcal{X}, \mathcal{P})$ such that the maximum cardinality of any variable is at most z and the minimum treewidth is at most w, $\mathbf{X} \subseteq \mathcal{X} \setminus \mathbf{E}$, a rational r and an instantiation $\mathbf{e} \in \Omega_{\mathbf{E}}$, Decision-MAP-$z$-$w$ is the problem of deciding if there is $\mathbf{x} \in \Omega_{\mathbf{X}}$ such that $p(\mathbf{x}, \mathbf{e}) > r$. MAP-$z$-$w$ is the respective optimization version.*

3 Complexity results

Hardness of MAP is known for polytrees where the maximum cardinality of a variable is $\Theta(\text{S}(\mathcal{N}))$ [Park and Darwiche, 2004]. More specifically, Decision-MAP-∞-w, where ∞ is used to indicate that there is no bound for z, is known to be NP-hard for $w = 2$, because the cardinality of the network used in the proof could be as large as the number of clauses in the SAT problem used in the reduction, and this number can be asymptotically as large as the SAT input size. It was also shown that Decision-MAP-z-∞ is NP-hard for $z = 2$. The next theorem strengthen those two results.

Theorem 3 *Decision-MAP-z-w is NP-hard even if $z = w = 2$.*

[2]If $p(\mathbf{e})$ is zero, then so is $p(\mathbf{x}, \mathbf{e})$, and the problem vanishes, as any $\mathbf{x}$ will be a maximizer for the problem.

Proof Hardness is shown using a reduction from *partition* problem, which is NP-hard [Garey and Johnson, 1979] and can be stated as follows: *given a set of m positive integers $s_1, \ldots, s_m$, is there a set $I \subset A = \{1, \ldots, m\}$ such that $\sum_{i \in I} s_i = \sum_{i \in A \setminus I} s_i$?* All the input is encoded using $b > 0$ bits. Denote $S = \frac{1}{2} \sum_{i \in A} s_i$ and call *even partition* a subset $I \subset A$ that achieves $\sum_{i \in I} s_i = S$. To solve *partition*, one may consider the rescaled problem (dividing every element by S), so as $v_i = \frac{s_i}{S} \leq 2$ are the elements and the goal is a partition I with sum equals to 1.

A binary polytree (so $z_{max} = 2$) with $3m + 1$ nodes, maximum number of parents per node equals to 2, and treewidth $w = 2$ is built in polynomial time. The binary nodes are $\mathbf{X} = \{X_1, \ldots, X_m\}$, $\mathbf{Y} = \{Y_0, Y_1, \ldots, Y_m\}$ and $\mathbf{E} = \{E_1, \ldots, E_m\}$. We denote by $\{x_i^T, x_i^F\}$ the states of X_i (similarly for Y_i and E_i). The structure of the network is presented in Figure 1. Each $X_i \in \mathbf{X}$ has no parents and uniform distribution, each E_i has X_i as sole parent, with probability values defined as $p(e_i^T|x_i^F) = 1$ and $p(e_i^T|x_i^T) = t_i$ (the values for e_i^F complement those of e_i^T), where t_i is obtained by evaluating 2^{-v_i} up to $4b + 3$ bits of precision and by rounding it up (if necessary), that is, $t_i = 2^{-v_i} + \text{error}_i$, where $0 \leq \text{error}_i < 2^{-(4b+3)}$. Clearly t_i can be computed in polynomial time and space in b (this ensures that the specification of the Bayesian network, which requires rational numbers, is polynomial in b). Furthermore, note that $2^{-v_i} \leq t_i \leq 2^{-v_i} + \text{error}_i < 2^{-v_i} + 2^{-(4b+3)} \leq 2^{-v_i + 2^{-4b}}$ (in short, this holds because 2^{-4b} in the exponent makes the value grow faster than the linear addition of $2^{-(4b+3)}$. Further details are omitted for simplicity).

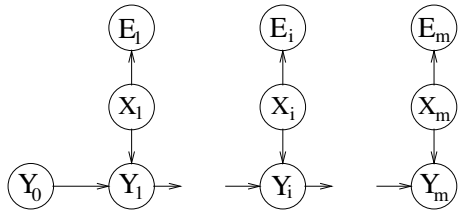

Figure 1: Network structure for the proof of Theorem 3.

Y_0 has no parents and $p(y_0^T) = 1$. For the nodes $Y_i \in \mathbf{Y}$, $1 \leq i \leq m$, the parents are X_i and Y_{i-1}, and the probability values are $p(y_i^T|y_{i-1}^T, x_i^T) = t_i$, $p(y_i^T|y_{i-1}^T, x_i^F) = 1$, and $p(y_i^T|y_{i-1}^F, x_i) = 0$ for both states $x_i \in \Omega_{X_i}$.

Note that with this specification and given the Markov condition of the network, we have (for any given $\mathbf{x} \in \Omega_{\mathbf{X}}$) $p(y_m^T|\mathbf{x}) = p(\mathbf{e}^{\mathbf{T}}|\mathbf{x}) = \prod_{i \in I} t_i$, where $I \subseteq A$ is the set of indices of the elements such that X_i is at the state x_i^T. Denote $\mathbf{t} = \prod_{i \in I} t_i$. Then $p(\mathbf{x}, \mathbf{e}^{\mathbf{T}}, y_m^F) = p(y_m^F|\mathbf{x}) p(\mathbf{x}, \mathbf{e}^{\mathbf{T}}) = p(\mathbf{x}) p(\mathbf{e}^{\mathbf{T}}|\mathbf{x}) \left(1 - p(y_m^T|\mathbf{x})\right) = \frac{1}{2^m} \mathbf{t}(1 - \mathbf{t})$. This is a concave quadratic function on $0 \leq \mathbf{t} \leq 1$ with maximum at 2^{-1} such that $\mathbf{t}(1-\mathbf{t})$ monotonically increases when $\mathbf{t}$ approaches one half (from both sides). If t_i was exactly 2^{-v_i}, then $\frac{1}{2^m} \mathbf{t}(1 - \mathbf{t}) = \frac{1}{2^m} 2^{-\sum_{i \in I} v_i} (1 - 2^{-\sum_{i \in I} v_i})$, which achieves the maximum of $\frac{1}{2^m} 2^{-1} (1 - 2^{-1})$ if and only if $\sum_{i \in I} v_i = 1$, that is, if there is an even partition. However 2^{-v_i} is used with

$4b+3$ bits of precision to ensure the computation is done in polynomial time, so the remainder of this proof addresses the question of how the numerical errors introduced in the definition of values t_i interfere in the main result. Hence, note that if I is not an even partition, then we know that one of the two conditions hold: (i) $\sum_{i\in I} s_i \leq S-1 \Rightarrow \sum_{i\in I} v_i \leq 1-\frac{1}{S}$, or (ii) $\sum_{i\in I} s_i \geq S+1 \Rightarrow \sum_{i\in I} v_i \geq 1+\frac{1}{S}$, because the original numbers s_i are integers. Consider these two cases.

If $\sum_{i\in I} s_i \geq S+1$, then $\mathbf{t} < \prod_{i\in I} 2^{-v_i+2^{-4b}}$ equals to

$$2^{\sum_{i\in I}(-v_i+2^{-4b})} \leq 2^{\frac{m}{2^{4b}}-(1+\frac{1}{S})} \leq 2^{-1-(\frac{1}{2^b}-\frac{1}{2^{3b}})} = l,$$

by using $S \leq 2^b$ and $m \leq b < 2^b$. On the other hand, if $\sum_{i\in I} s_i \leq S-1$, then $\mathbf{t} \geq \prod_{i\in I} 2^{-v_i}$ equals to

$$2^{-\sum_{i\in I} v_i} \geq 2^{-(1-\frac{1}{S})} = 2^{-1+\frac{1}{S}} \geq 2^{-1+\frac{1}{2^b}} = u.$$

Now suppose I' is an even partition. Then we know that the corresponding $\mathbf{t}'$ satisfies $2^{-1} \leq \mathbf{t}'$ and

$$\mathbf{t}' < \prod_{i\in I'} 2^{-v_i+2^{-4b}} = 2^{\sum_{i\in I'}(-v_i+2^{-4b})} \leq 2^{-1+\frac{1}{2^{3b}}} = a.$$

To complete the proof, we show that the distance between $\mathbf{t}'$ and 2^{-1} is always less than the distance between $\mathbf{t}$ and 2^{-1} of a non-even partition plus a gap, that is,

$$\begin{aligned} |\mathbf{t}'-2^{-1}| + 2^{-(3b+2)} &\leq a - 2^{-1} + 2^{-(3b+2)} \\ &< \min\{u-2^{-1}, 2^{-1}-l\} \leq |\mathbf{t}-2^{-1}|, \end{aligned} \quad (2)$$

which can be proved by analyzing the two elements of the min. The first term holds because

$$\begin{aligned} a+2^{-(3b+2)} - 2^{-1} &< a \cdot 2^{\frac{1}{2^{2b}}} - 2^{-1} \\ &= 2^{-1+\frac{2^{-b}+2^{-2b}}{2^b}} - 2^{-1} < 2^{-1+\frac{1}{2^b}} - 2^{-1} = u - 2^{-1}. \end{aligned}$$

The second comes from the fact that the function $h(b) = a + l + 2^{-(3b+2)} = 2^{-1+\frac{1}{2^{3b}}} + 2^{-1-(\frac{1}{2^b}-\frac{1}{2^{3b}})} + 2^{-(3b+2)}$ is less than 1 for $b = 1, 2$ (by inspection), it is a monotonic increasing function for $b \geq 2$ (the derivative is always positive), and it has $\lim_{b\to\infty} h(b) = 1$. Hence, we conclude that $h(b) < 1$, which implies

$$a + l + 2^{-(3b+2)} < 1 \iff a - 2^{-1} + 2^{-(3b+2)} < 2^{-1} - l.$$

This concludes that there is a gap of at least $2^{-(3b+2)}$ between the worst value of $\mathbf{t}'$ (relative to an even partition) and the best value of $\mathbf{t}$ (relative to a non-even partition), which will be used next to specify the threshold of the MAP problem. Now, set up $\mathbf{X}$ to be the MAP variables and $\mathbf{E} = \mathbf{e}$, $Y_m = y_m^F$ to be the evidence, so as the MAP decision becomes

$$\max_{\mathbf{x}\in\Omega_{\mathbf{X}}} p(\mathbf{x}, \mathbf{e^T}, y_m^F) > r = c \cdot \frac{1}{2^m}, \quad (3)$$

where c is defined as $a'\cdot(1-a')$, with a' equals a evaluated up to $3b+2$ bits and rounded up, which implies that $2^{-1} < a \leq a' < a+2^{-(3b+2)}$. By Eq. (2), a' is closer to one half than any $\mathbf{t}$ of a non-even partition, so the value c is certainly greater than any value that would be obtained by a non-even partition. On the other hand, a' is farther from 2^{-1} than a, so we can conclude that c separates even and non-even partitions, that is, $\mathbf{t}\cdot(1-\mathbf{t}) < c \leq a\cdot(1-a) < \mathbf{t}'\cdot(1-\mathbf{t}')$ for any $\mathbf{t}$ corresponding to a non-even partition and any $\mathbf{t}'$ of an even partition. Thus, a solution of the MAP problem obtains $p(\mathbf{x}, \mathbf{e^T}, y_m^F) > r$ if and only there is an even partition.[3] □

Corollary 4 *Decision-MAP is NP-complete when restricted to binary polytrees.*

Proof It follows directly from Theorem 3 and the well-known fact that Decision-MAP is in NP when w is fixed. □

The next theorem shows that the problem remains hard even in trees. The tree used for the proof is probably the simplest practical tree: a Naive Bayes structure, where there is a node called *class* with direct children called *features*. These features are independent of each other given the class. The simplicity of this tree makes the result stronger. Moreover, by reducing from the maximum-satisfiability problem, we show later that the inapproximability results before known for polytrees [Park and Darwiche, 2004] (when the maximum cardinality is not bounded) extend to the case of trees.

Theorem 5 *Decision-MAP-∞-w is NP-hard if $w = 1$ and the network topology follows the Naive Bayes structure.*

Proof We use a reduction from MAX-2-SAT. Let $X_1 \ldots, X_m$ be variables of a SAT problem with clauses $C_1, \ldots, C_{m'}$ written in 2CNF, that is, each clause is composed of a disjunction of two literals. Each literal belongs to $\Omega_{X_j} = \{x_j, \neg x_j\}$ for a given j. Without loss of generality, we assume that each clause involves exactly two distinct variables of $X_1 \ldots, X_m$. Let $b > 0$ be the number of bits to specify the MAX-2-SAT problem.

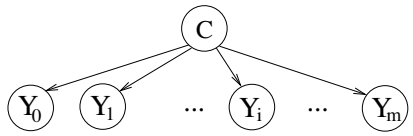

Figure 2: Network structure for the proof of Theorem 5.

Take a Naive Bayes shaped network. The idea is to build a network where the features represent the SAT variables and the class represents ways to satisfy the clauses, such that summing over the class states will produce the number of satisfying clauses. This is performed by considering the left and right literals of each clause in three conditions: (i) the clause is satisfied by its left literal (in this case, the value of the right literal is ignored), (ii) the clause is not satisfied by its left literal, but it is by its right literal, (iii) the clause is not satisfied by either of them. Therefore, Let C be the root of the Naive Bayes structure and $Y_1, \ldots, Y_m$ the binary features (as shown in Figure 2) such that $\Omega_{Y_j} = \{y_j^T, y_j^F\}$ for every j. Define the variable C to have $2m'$ states and uniform prior, that is, $p(c) = \frac{1}{2m'}$ for every $c \in \Omega_C$, where

[3] The conditional version of Decision-MAP could be used in the reduction too by including the term $\frac{1}{p(\mathbf{e^T}, y_m^F)}$ in r, which can be computed in polynomial time by a BN propagation in polytrees [Pearl, 1988], and it does not depend on the choice $\mathbf{x}$.

Ω_C equals to $\{c_{1L}, c_{1R}, c_{2L}, c_{2R}, \ldots, c_{m'L}, c_{m'R}\}$. Denote by L_i the literal of clause C_i with the smallest index, and by R_i the literal with the greatest index. Define the conditional probability functions of each Y_j given C as follows: $p(y_j^T|c_{iL}) = p(y_j^T|c_{iR}) = \frac{1}{2}$ if $L_i, R_i \notin \Omega_{X_j}$, that is, X_j does not appear in clause C_i, $p(y_j^T|c_{iR}) = 1$ if $(x_j = R_i) \vee (\neg x_j = L_i)$, $p(y_j^T|c_{iR}) = 0$ if $(x_j = L_i) \vee (\neg x_j = R_i)$, $p(y_j^T|c_{iL}) = 1$ if $(x_j = L_i)$, $p(y_j^T|c_{iL}) = 0$ if $(\neg x_j = L_i)$, $p(y_j^T|c_{iL}) = \frac{1}{2}$ if $R_i \in \Omega_{X_j}$. Define Y_0 as an extra feature such that $p(y_0^T|c_{iL}) = 1$ and $p(y_0^T|c_{iR}) = 1/2$ for every i. The probability values for y_j^F complement these numbers, that is, $p(y_j^F|c) = 1 - p(y_j^T|c)$ for every $c \in \Omega_C$. Hence, $\max_{y_0 \ldots y_m} p(y_0, y_1, \ldots, y_m) = \max_{y_1, \ldots y_m} p(y_0^T, y_1, \ldots y_m)$, because the vector $p(y_0^T|C)$ has $p(y_0^T|c) \geq p(y_0^F|c)$ for every $c \in \Omega_C$ (there is no reason to choose y_0^F in place of y_0^T as the probability value of the latter is always greater than that of the former for every given c). It is clear that the transformation is polynomial in b, as the network has $m+1$ nodes, with at most $2m'$ states (both m and m' are $O(b)$), and the probability values are always 0, $1/2$ or 1. By simple manipulations, we have

$$p(y_0^T, y_1, \ldots y_m) = \frac{1}{2m'} \frac{1}{2^{m-2}} \sum_i \left(p(y_{j_{iL}}|c_{iL}) p(y_{j_{iR}}|c_{iL}) \cdot \right.$$
$$\left. \cdot p(y_0^T|c_{iL}) + p(y_{j_{iL}}|c_{iR}) p(y_{j_{iR}}|c_{iR}) p(y_0^T|c_{iR})\right),$$

where j_{iL} and j_{iR}, with $j_{iL} < j_{iR}$, are the indices of the two variables that happen in clause C_i (the probability of all other variables Y_j that appeared in the product have led to the fraction $\frac{1}{2}$ because they do not happen in C_i and hence disappeared to form the constant $\frac{1}{2^{m-2}}$ that has been put outside the summation). Yet by construction, $p(y_0^T, y_1, \ldots y_m) =$

$$\frac{1}{2m'} \frac{1}{2^{m-1}} \sum_i \left(p(y_{j_{iL}}|c_{iL}) + p(y_{j_{iL}}|c_{iR}) p(y_{j_{iR}}|c_{iR})\right).$$

Note that $p(y_{j_{iL}}|c_{iL}) = 1$ if and only if $p(y_{j_{iL}}|c_{iR}) = 0$ (and $p(y_{j_{iL}}|c_{iL}) = 0$ if and only if $p(y_{j_{iL}}|c_{iR}) = 1$), and $p(y_{j_{iL}}|c_{iL}) = 1$ if and only if the instantiation of the Boolean variable in L_i satisfies clause C_i. In this case, $p(y_{j_{iL}}|c_{iR}) = 0$, and the sum $p(y_{j_{iL}}|c_{iL}) + p(y_{j_{iL}}|c_{iR}) p(y_{j_{iR}}|c_{iR})$ equals to 1. On the other hand, if the Boolean variable in L_i does not make clause C_i satisfiable, then $p(y_{j_{iL}}|c_{iL}) + p(y_{j_{iL}}|c_{iR}) p(y_{j_{iR}}|c_{iR}) = p(y_{j_{iR}}|c_{iR})$, that is, it is one if and only if the Boolean in R_i satisfies C_i. Because we sum over all clauses, $p(y_0^T, y_1, \ldots y_m) = \frac{k}{2^m m'} \iff k$ clauses are satisfiable. Hence, solving $\max_{y_0 \ldots y_m} p(y_0, y_1, \ldots, y_m)$ is the same as solving MAX-2-SAT, and the reduction of the decision version easily follows. □

We show next a stronger inapproximability result than previously stated in the literature, because we make use of trees instead of polytrees. Recall that an approximation algorithm for a maximization problem where the exact maximum value is $M > 0$ is said to achieve a ratio $\alpha > 1$ from the optimal if the resulting value is guaranteed to be greater than or equal to $\frac{M}{\alpha}$. We demonstrate that approximating MAP is NP-hard even if the network topology is as simple as a tree. This leaves no hope of approximating MAP in polynomial time when the number of states per variable is not bounded.

Theorem 6 *It is NP-hard to approximate MAP-∞-w, with $w = 1$, to any ratio $\alpha = 2^{S(\mathcal{N})^\varepsilon}$, for fixed $0 \leq \varepsilon < 1$.*

Proof We show that it is possible to reduce MAX-2-SAT to the approximate version of MAP-∞-1 in polynomial time and space in the size of input. The idea is similar to the repeated construction used in [Park and Darwiche, 2004]. We build q copies of the network of Theorem 5 (superscripts are added to the variables to distinguish the copies as follows: the nodes of the t-th copy are named $C^t, Y_0^t, \ldots, Y_m^t$) and link them by a common binary parent D of all the C^t nodes (as shown in Fig.3), with states $\{d^T, d^F\}$. We define $p(d^T) = 1$ and $p(c^t|d^T)$ remains uniform as before, for every node C^t.

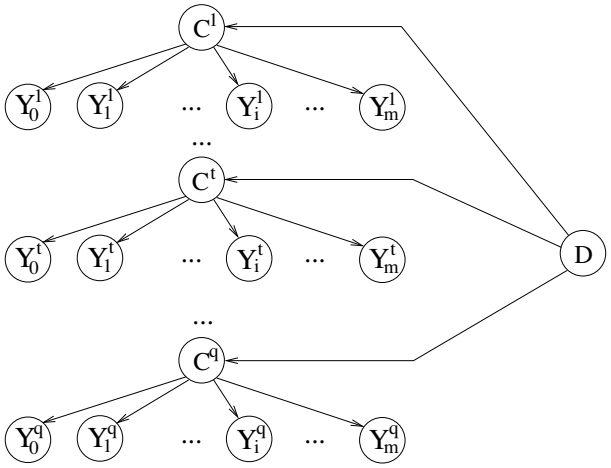

Figure 3: Network structure for the proof of Theorem 6.

By construction, we have $p(y_0^1, \ldots y_m^1, \ldots, y_0^q, \ldots y_m^q) =$

$$\sum_d \prod_{t=1}^q p(y_0^t, y_1^t, \ldots y_m^t|d) p(d) = \prod_{t=1}^q p(y_0^t, y_1^t, \ldots y_m^t|d^T),$$

and hence each copy has independent computations given d^T. By the argument in the proof of Theorem 5 for each copy, we obtain $p(y_0^1, \ldots y_m^1, \ldots, y_0^q, \ldots y_m^q) = \prod_{t=1}^q p(y_0^t, \ldots y_m^t) = \prod_{t=1}^q \frac{k}{2^m m'} = \left(\frac{k}{2^m m'}\right)^q$ if and only if k clauses are satisfiable in the MAX-2-SAT problem. Suppose we want to decide if at least $1 < k' \leq m'$ clauses are satisfiable (the restriction of $k' > 1$ does not lose generality). Using the approximation over this new network with ratio α, if at least k' clauses are satisfiable, then we must have $p(y_0^1, \ldots y_m^1, \ldots, y_0^q, \ldots y_m^q) \geq \frac{1}{\alpha} \left(\frac{k'}{2^m m'}\right)^q$. On the other hand, if it is not possible to satisfy k' clauses, then we know that $p(y_0^1, \ldots y_m^1, \ldots, y_0^q, \ldots y_m^q) \leq \left(\frac{k'-1}{2^m m'}\right)^q$. Now we need to show that it is possible to pick q such that $\left(\frac{k'-1}{2^m m'}\right)^q < \frac{1}{\alpha} \left(\frac{k'}{2^m m'}\right)^q$ and such that q is polynomially bounded. The proof concludes by choosing a q such that $q > (\log(2) k' f'(b)^\varepsilon)^{\frac{1}{1-\varepsilon}}$, which is polynomial in the

input size (b is the input size of MAX-2-SAT and f' is the size of one of the Naive Bayes structures) and implies $q > \log(2)k'q^{\varepsilon} \cdot f'(b)^{\varepsilon} \implies q > k'\log(2^{(q \cdot f'(b))^{\varepsilon}}) \implies q > k'\log(2^{\mathrm{S}(\mathcal{N})^{\varepsilon}}) \implies q > k'\log(\alpha)$. By Taylor expansion we have $\log\left(\frac{k'}{k'-1}\right) \geq \frac{1}{k'}$ (for any $k' > 1$) and thus $q > k'\log(\alpha) \implies q > \frac{\log(\alpha)}{\log\left(\frac{k'}{k'-1}\right)} \implies (k'-1)^q < \frac{1}{\alpha}(k')^q \implies \left(\frac{k'-1}{2^m m'}\right)^q < \frac{1}{\alpha}\left(\frac{k'}{2^m m'}\right)^q$. □

Theorem 7 shows that, even with bounded cardinality and very simple trees, it is hard to solve MAP. The construction that is used in the proof is trick, aiming at Eq. (4), which has the product of values t_i from the partition problem (as in Theorem 3) appearing in a competing way in the equation.

Theorem 7 *Decision-MAP-z-w is NP-hard even if $z = 5$ and $w = 1$ and the network topology follows a Hidden Markov Model (HMM) structure.*

Proof This proof uses the exact same rescaled *partition* problem of the proof of Theorem 3, as well as the idea of approximating exponentials to guarantee that the reduction is polynomial (refer there for the definition of the problem).

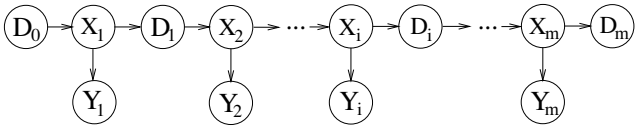

Figure 4: Structure for the proof of Theorem 7. It is not exactly an HMM structure, but it would be trivial to include another child to every D_i node without changing the results.

We construct (in polynomial time) a tree with $3m + 1$ nodes: $\mathbf{X} = \{X_1, \ldots, X_m\}$, $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$ and $\mathbf{D} = \{D_0, D_1, \ldots, D_m\}$, such that $\Omega_{X_i} = \{x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}\}$ has 5 states, $\Omega_{Y_i} = \{y_i^T, y_i^F\}$ is binary, and $\Omega_{D_i} = \{d_i^T, d_i^F, d_i^*\}$ is ternary. The structure of the network is presented in Figure 4. The probability functions are defined by Table 1 (except for D_0, which has uniform distribution).

First, we show that $p(\mathbf{y}) = p(y_1, \ldots, y_m) = \frac{1}{2^m}$ for any configuration $\mathbf{y} \in \Omega_{\mathbf{Y}}$. By construction, we have $p(y_i|d_{i-1}) = \sum_{x_i} p(y_i|x_i)p(x_i|d_{i-1}) = \frac{1}{2}$ for any value of y_i, d_{i-1}. Now,

$$\begin{aligned} p(y_1, \ldots, y_m) &= \sum_{d_{m-1}} p(y_m|d_{m-1})p(d_{m-1}, y_1, \ldots, y_{m-1}) \\ &= 1/2 \cdot p(y_1, \ldots, y_{m-1}). \end{aligned}$$

Applying the same idea successively, we obtain $p(y_1, \ldots, y_m) = \frac{1}{2^m}$. Using a similar procedure, we can obtain the values for $p(d_m^T, \mathbf{y})$ and $p(d_m^F, \mathbf{y})$ as follows:

$$p(d_m^T, \mathbf{y}) = \begin{cases} t_i \cdot \frac{1}{2} \cdot p(d_{m-1}^T, y_1, \ldots, y_{m-1}) & \text{if } y_m = y_m^T, \\ 1 \cdot \frac{1}{2} \cdot p(d_{m-1}^T, y_1, \ldots, y_{m-1}) & \text{if } y_m = y_m^F, \end{cases}$$

and applying successively, we get $p(d_m^T, \mathbf{y}) = \frac{2^{-m}}{3}\prod_{i \in I} t_i$, where $I \subset A$ is the set of indices of elements such that Y_i is

Table 1: Probability values used in the proof of Theorem 7.

$p(Y_i\|X_i)$	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}
y_i	1	1	0	0	1/2
$\neg y_i$	0	0	1	1	1/2

$p(X_i\|D_{i-1})$	d_{i-1}^T	d_{i-1}^F	d_{i-1}^*
x_{i1}	1/2	0	0
x_{i2}	0	1/2	0
x_{i3}	0	1/2	0
x_{i4}	1/2	0	0
x_{i5}	0	0	1

$p(D_i\|X_i)$	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}
d_i^T	t_i	0	0	1	0
d_i^F	0	1	t_i	0	0
d_i^*	$1-t_i$	0	$1-t_i$	0	1

at the state y_i^T (the ratio $\frac{1}{3}$ comes from the uniform $p(D_0)$). Likewise,

$$p(d_m^F, \mathbf{y}) = \begin{cases} 1 \cdot \frac{1}{2} \cdot p(d_{m-1}^F, y_1, \ldots, y_{m-1}) & \text{if } y_m = y_m^T, \\ t_i \cdot \frac{1}{2} \cdot p(d_{m-1}^F, y_1, \ldots, y_{m-1}) & \text{if } y_m = y_m^F, \end{cases}$$

and again $p(d_m^F, \mathbf{y}) = \frac{2^{-m}}{3}\prod_{i \in A \setminus I} t_i$. Therefore,

$$\begin{aligned} \max_{\mathbf{y}} p(d_m^*, \mathbf{y}) &= \max_{\mathbf{y}} \left(p(\mathbf{y}) - p(d_m^T, \mathbf{y}) - p(d_m^F, \mathbf{y})\right) \\ &= \frac{1}{2^m} - \min_{\mathbf{y}} \left(p(d_m^T, \mathbf{y}) + p(d_m^F, \mathbf{y})\right) \\ &= \frac{1}{2^m}\left(1 - \frac{1}{3}\min_{\mathbf{y}}\left(\prod_{i \in I} t_i + \prod_{i \in A \setminus I} t_i\right)\right). \end{aligned} \tag{4}$$

Consider that $t_i = 2^{-v_i}$. The function $\prod_{i \in I} t_i + \prod_{i \in A \setminus I} t_i = 2^{-\sum_{i \in I} v_i} + 2^{-\sum_{i \in A \setminus I} v_i}$ is convex and achieves its minimum when $2^{-\sum_{i \in I} v_i} = 2^{-\sum_{i \in A \setminus I} v_i} \iff \sum_{i \in I} v_i = \sum_{i \in A \setminus I} v_i = 1$. Thus, using $\mathbf{Y}$ as the MAP variables and d_m^* as the evidence, we obtain $\max_{\mathbf{y}} p(d_m^*, \mathbf{y}) = \frac{2}{3}\frac{1}{2^m}$ if and only if there is an even partition. This is still flaw in one respect: the specification of the network depends on computing t_i, for each $i \in A$, which can be done only to a certain precision (we can only use a number of places that is polynomial in b). The derivation follows in a similar way as done in Theorem 3, but using $6b + 3$ bits of precision. □

Corollary 8 *Decision-MAP is NP-complete when the graph is restricted to a tree and variables have bounded cardinality.*

Proof It follows directly from Theorem 7 and the fact that Decision-MAP is in NP when w is fixed. □

Despite the hardness results, we obtain a fully polynomial-time approximation scheme when cardinality and treewidth are bounded. For this purpose, algorithms for BU can be seen as a way to represent and compute a polynomial function on the probability values that define the network. The idea to approximate MAP is that we can iteratively use an inference algorithm for BU, but computing the polynomial for every

possible configuration of the query variables $\mathcal{X}^{\text{map}}$. As there are too many instantiation of $\mathcal{X}^{\text{map}}$, we check at every intermediate step of the algorithm for options $\mathbf{x}^{\text{map}}$ that are close to each other, so that the error of using one or another is below some threshold. In this case, we discard options as soon as possible and keep only a polynomial number of them, that are enough to approximate well the result.

Theorem 9 *MAP-z-w has a FPTAS for any fixed z and w.*

Proof We construct an approximation algorithm A that, for a given $\varepsilon > 0$, is polynomial in $\mathrm{S}(\mathcal{N})$ and in $\frac{1}{\varepsilon}$ such that the value $p^A(\mathbf{x}^{\text{map}}, \mathbf{e})$ obtained by A is at least $\frac{p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e})}{1+\varepsilon}$, where $p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e})$ stands for the optimal solution value of MAP-z-w, that is, $\mathbf{x}^{\text{map}}_{\text{opt}} = \text{argmax}_{\mathbf{x}^{\text{map}}} p(\mathbf{x}^{\text{map}}, \mathbf{e})$. We know that $p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e}) > 0$ because $\sum_{\mathbf{x}^{\text{map}}} p(\mathbf{x}^{\text{map}}, \mathbf{e}) = p(\mathbf{e}) > 0$ and thus the value achieving the maximum cannot be zero ($p(\mathbf{e}) > 0$ does not lose generality, as discussed before).

Take a binary tree decomposition $(\{\mathbf{C}_1, \ldots, \mathbf{C}_{n'}\}, T)$ with treewidth (at most) w, such that $n' \leq 2n$ and $\mathbf{C}_j$ indicates both a node of the tree T and the set of network variables associated to that node, which can be obtained in polynomial time [Bodlaender, 1993; Shenoy, 1996]. Let $w' = w + 1$ be a bound for the maximum number of variables in a single node of the decomposition. Elect a node and assume all edges of T point towards the opposite direction of it. Without loss of generality, let $\mathbf{C}_1$ be this node and $\mathbf{C}_1, \ldots, \mathbf{C}_{n'}$ be a topological order with respect to this tree. Let $\mathbf{C}_{j_p} = \text{PA}_{\mathbf{C}_j}$ be the only parent of $\mathbf{C}_j$ in the tree and $\Lambda_{\mathbf{C}_j}$ be the children of $\mathbf{C}_j$.

Let $\mathbf{X}_j^{\text{last}} = \mathbf{C}_j \setminus \mathbf{C}_{j_p}$ be the set of nodes of $\mathcal{G}$ in $\mathbf{C}_j$ that do not appear in the parent $\mathbf{C}_{j_p}$ (they also do not appear in any other node towards $\mathbf{C}_1$ because of the properties of a tree decomposition). To solve BU, we can use the following recursion, which is processed from $j = n'$ to 1: $p(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j}) =$

$$\sum_{\Omega_{\mathbf{X}_j^{\text{last}} \setminus \mathbf{X}'}} \prod_{X_i \in \mathbf{X}_j^{\text{proc}}} p(x_i|\pi_{X_i}) \prod_{\mathbf{C}_{j'} \in \Lambda_{\mathbf{C}_j}} p(\mathbf{u}_{\mathbf{C}_{j'}}|\mathbf{v}_{\mathbf{C}_{j'}}), \quad (5)$$

and $\mathbf{X}_j^{\text{proc}} = \{X_i \in (\mathbf{C}_j \setminus \bigcup_{\mathbf{C}_{j'} \in \Lambda_{\mathbf{C}_j}} \mathbf{U}_{\mathbf{C}_{j'}}) : (X_i \cup \text{PA}_{X_i}) \cap \mathbf{X}_j^{\text{last}} \neq \emptyset\}$ is the set of variables whose local probability functions were not processed yet (but need to be) in order to sum out over $\mathbf{X}_j^{\text{last}} \setminus \mathbf{X}'$ ($\mathbf{X}'$ are the variables whose states are fixed). Furthermore, $\mathbf{U}_{\mathbf{C}_j}$ is composed of elements of $\mathbf{C}_j$ and descendants that are also present in the parent $\mathbf{C}_{j_p}$ and whose local probability distributions were already taken into account (they do appear in the left side of the conditioning bar in a potential in $\mathbf{C}_j$ or in its descendants), and finally $\mathbf{V}_{\mathbf{C}_j}$ are the variables that already appeared in the right side of the conditioning bar (but not in the left side nor they were summed out). Note that Eq. (5) is computed for every $\mathbf{u}_{\mathbf{C}_j}$ and every $\mathbf{v}_{\mathbf{C}_j}$. Each step j can be seen as one bucket in the bucket elimination algorithm (apart from the fact that we might sum out more than one variable at once – this can also be implemented in the bucket elimination), or the processing of a node in a join tree algorithm. In fact, the values $p(\mathbf{u}_{\mathbf{C}_{j'}}|\mathbf{v}_{\mathbf{C}_{j'}})$ that come from previous computations represent the information from the children of $\mathbf{C}_j$, with domains $\Omega_{\mathbf{U}_{\mathbf{C}_{j'}} \cup \mathbf{V}_{\mathbf{C}_{j'}}}$ that come from independent subtrees, and are used to construct $p(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j})$ over $\Omega_{\mathbf{U}_{\mathbf{C}_j} \cup \mathbf{V}_{\mathbf{C}_j}}$. The recursion is evaluated for each j, and finally $p(\mathbf{u}_{\mathbf{C}_1}) = p(\mathbf{x}')$, where $\mathbf{x}'$ are the fixed states of $\mathbf{X}'$. The running time, in number of additions and multiplications, is at most $\sum_j (1 + |\Lambda_{\mathbf{C}_j}|) \cdot z(\mathbf{C}_j) \in O(n \cdot z_{max}^w) \in O(n)$, as both z_{max} and $w = \max_j |\mathbf{C}_j|$ are bounded.

To solve the MAP problem, the evidence $\mathbf{e}$ is considered part of $\mathbf{x}'$, and Eq. (5) is computed for every distinct instantiation of the MAP variables that have already appeared in a given subtree. The tree is iteratively processed until the root is reached. We use the notation $p_{\mathbf{x}_\mathbf{C}^{\text{map}}}(\mathbf{u}|\mathbf{v}) = p(\mathbf{u}|\mathbf{v})$ such that $\mathbf{u}$ and $\mathbf{v}$ agree with $\mathbf{x}_\mathbf{C}^{\text{map}}$, where $\mathbf{x}_\mathbf{C}^{\text{map}} \in \Omega_{\mathbf{X}_\mathbf{C}^{\text{map}}}$ (and $\mathbf{X}_\mathbf{C}^{\text{map}} \subseteq \mathcal{X}^{\text{map}}$), to indicate that the MAP variables that appear in a subtree rooted at $\mathbf{C}$ are fixed to the value $\mathbf{x}_\mathbf{C}^{\text{map}}$ and hence this can be interpreted as having distinct potentials $p_{\mathbf{x}_\mathbf{C}^{\text{map}}}$ for distinct instantiations $\mathbf{x}_\mathbf{C}^{\text{map}}$. Note that $p_{\mathbf{x}_\mathbf{C}^{\text{map}}}$ has $d = z((\mathbf{U} \cup \mathbf{V}) \setminus (\mathbf{E} \cup \mathbf{X}_\mathbf{C}^{\text{map}}))$ elements as arguments (the part of $\mathbf{u}, \mathbf{v}$ corresponding to $\mathbf{e}$ and $\mathbf{x}_\mathbf{C}^{\text{map}}$ are fixed). With this approach, we can rewrite Eq. (5) to introduce the MAP variables just by replacing p with $p_{\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}}$, where $\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}$ is restricted to the variables that already appeared in the computation (and further supposing that $\mathbf{X}_{\mathbf{C}_\mathbf{j}}^{\text{map}}$ is never summed out). This algorithm is correct, but it eventually has to process an exponential number of potentials $p_{\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}}$ (one for each $\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}$), because as the algorithm goes on, as the set of already processed MAP variables increases, until the MAP variables are again altogether in the root.

To overcome this situation, we devise a criterion to keep only potentials that are reasonably apart. Notice now that $1 \geq p(\mathbf{e}) \geq p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e}) > 2^{-g(\mathrm{S}(\mathcal{N}))}$, where g is a polynomial function, because $p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e})$ is obtained from a sequence of additions and multiplications through Eq. (5) over numbers of the input. The same argument holds for every intermediate probability value that is computed through Eq. (5): we have that $p_{\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}}(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j}) > 0 \Rightarrow p_{\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}}(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j}) > 2^{-g(\mathrm{S}(\mathcal{N}))}$, for a given polynomial function g. Hence, let g be a polynomial function that satisfies such condition for every number involved in the calculations. Because each $p_{\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}}(\mathbf{U}_{\mathbf{C}_j}|\mathbf{V}_{\mathbf{C}_j})$ can be represented by a vector in the dimension $d = z((\mathbf{U}_{\mathbf{C}_j} \cup \mathbf{V}_{\mathbf{C}_j}) \setminus (\mathbf{E} \cup \mathbf{X}_{\mathbf{C}_\mathbf{j}}^{\text{map}}))$, the idea is to show that we can fix an upper bound to the number of candidates $p_{\mathbf{x}_{\mathbf{C}_\mathbf{j}}^{\text{map}}}(\mathbf{U}_{\mathbf{C}_j}|\mathbf{V}_{\mathbf{C}_j})$ that are propagated from one step to the next. Following the ideas of [Papadimitriou and Yannakakis, 2000], we divide the hypercube $[2^{-g(\mathrm{S}(\mathcal{N}))}, 1]^d$ into a lattice of hypercubes such that, in each coordinate, the ratio of the largest to the smallest value is $1 + \frac{\varepsilon}{2w'n'}$, which produces a number of hypercubes bounded by

$$O\left(\left(\frac{\log 2^{g(\mathrm{S}(\mathcal{N}))}}{\frac{\varepsilon}{2w'n'}}\right)^d\right) \in O\left(\left(\frac{2w'n' \cdot g(\mathrm{S}(\mathcal{N}))}{\varepsilon}\right)^{z^{w'}}\right), \quad (6)$$

where the log appears because the lattice is created from 1 to

$2^{-g(\mathrm{S}(\mathcal{N}))}$, successively dividing the coordinate by $1+\frac{\varepsilon}{2w'n'}$ (a bin for the exact zero probability is also allocated). In each of these smaller hypercubes, we keep at most one vector, so Eq. (6) bounds the number of potentials $p_{\mathbf{x}^{\text{map}}_{\mathbf{C_j}}}$ that are processed in each step j. This procedure is carried out through the steps, so we have a polynomial time procedure both in $\mathrm{S}(\mathcal{N})$ and in $\frac{1}{\varepsilon}$, because the running time is less than $O(n)$ times Eq. (6) to the power of 2 (from the decomposition).

It remains to show that the resulting $p^A(\mathbf{x}^{\text{map}}, \mathbf{e})$ is at least $\frac{p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e})}{1+\varepsilon}$. Each value $p_{\mathbf{x}^{\text{map}}_{\mathbf{C_j}}}(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j})$ is obtained from a sum of multiplications, with at most $|\Lambda_{\mathbf{C}_j}|+1$ terms each. Hence, the approximation at any step satisfies $p^A_{\mathbf{x}^{\text{map}}_{\mathbf{C_j}}}(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j}) >$

$$\sum_{\Omega_{\mathbf{X}^{\text{last}}_j \setminus (\mathbf{E}\cup\mathbf{X}^{\text{map}}_{\mathbf{C_j}})}} \prod_{X_i\in\mathbf{X}^{\text{proc}}_j} p(x_i|\pi_{X_i}) \cdot \prod_{\mathbf{C}_{j'}\in\Lambda_{\mathbf{C}_j}} \frac{p_{\mathbf{x}^{\text{map}}_{\mathbf{C_j}}}(\mathbf{u}_{\mathbf{C}_{j'}}|\mathbf{v}_{\mathbf{C}_{j'}})}{(1+\frac{\varepsilon}{2w'n'})^{l_{\mathbf{C}_{j'}}}}$$

$> \frac{p_{\mathbf{x}^{\text{map}}_{\mathbf{C_j}}}(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j})}{(1+\frac{\varepsilon}{2w'n'})^{l_{\mathbf{C}_j}}}$, as $l_{\mathbf{C}_j}$, the the number of variables that appear in nodes of the subtree of T rooted at $\mathbf{C}_j$, is equal to $l_{\mathbf{C}_j} = |\mathbf{C}_j| + \sum_{\mathbf{C}_{j'}\in\Lambda_{\mathbf{C}_j}} l_{\mathbf{C}_{j'}}$. In the root of T, we have that $p^A_{\mathbf{x}^{\text{map}}_{\mathbf{C_1}}}(\mathbf{u}_{\mathbf{C}_1}) > \frac{p_{\mathbf{x}^{\text{map}}_{\mathbf{C_1}}}(\mathbf{u}_{\mathbf{C}_1})}{(1+\frac{\varepsilon}{2w'n'})^{w'n'}} = \frac{p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e})}{(1+\frac{\varepsilon}{2w'n'})^{w'n'}}$, as $w'n' \geq l_{\mathbf{C}_1}$ (there are less than w' elements per node $\mathbf{C}_j$). Two points are worth mentioning: (i) the bin created for the value zero in the lattice guarantees that if some intermediate value $p^A_{\mathbf{x}^{\text{map}}_{\mathbf{C_j}}}(\mathbf{u}_{\mathbf{C}_j}|\mathbf{v}_{\mathbf{C}_j})$ is zero, then it matches the exact value and the approximation does not degenerate; (ii) the fact that the tree decomposition is binary ensures that the computations in each step are polynomial. Finally, $p^A(\mathbf{x}^{\text{map}}, \mathbf{e}) > \frac{p(\mathbf{x}^{\text{map}}_{\text{opt}}, \mathbf{e})}{1+\varepsilon}$, because of the inequality $(1+\frac{\varepsilon}{r})^r \leq 1+2\varepsilon$, which is valid for any $0 \leq \varepsilon \leq 1$ and integer $r > 0$ (it is convex in ε). □

It follows from Theorem 9 that MAP-z-w has a FPTAS in any network with bounded treewidth and cardinality, including polytrees. At first, this result seems to contradict past results, where it is stated that approximating MAP is hard even in polytrees [Park and Darwiche, 2004]. But we assume a bound for the cardinality of the variables, which is the most common situation in practical BNs, while previous results work with a more general class of networks and do not assume the bound. Currently, the FPTAS is theoretical, because the number of hypercubes that are used to divide the vector space (given by Eq. (6)) is huge, that is, the division is so granulated that the number of discarded potentials (belonging to the same hypercube) is very small and might not be computationally attractive. However, it brings the possibility of developing other techniques that might efficiently solve the MAP in practical applications.

4 Conclusion

This paper closes a few theoretical questions for the MAP problem in Bayesian networks. We address the following complexity results:

- MAP remains hard in simple *binary* polytrees with at most two parents per node, strengthening previous results by using a binary version of a polytree.
- MAP remains hard in trees with no bound on maximum cardinality but network topology as simple as a Naive Bayes structure.
- MAP in trees without a bound in the maximum cardinality does not admit a polynomial approximation scheme, again strengthening previous results.
- MAP remains hard in trees with bounded maximum cardinality and network topology as simple as a Hidden Markov Model structure, which implies hardness for trees with bounded cardinality per variable.

Altogether these new complexity proofs strongly indicate that MAP problems are hard even when the underlying structure of the BN is very simple. It is also shown that an approximation method with theoretical guarantees is possible, but it is necessary to work on reducing the hidden constants and exponents of it. This is a point to be addressed in future work, as well as trying to devise an efficient pseudo-polynomial time algorithm (in fact, both ideas are correlated). Another possibility is to study (theoretically and empirically) how to select potentials from the sets in order to further reduce their sizes, which may produce very good approximation results in.

Acknowledgments

This work has been partially supported by the Swiss NSF grant n. 200020_134759 / 1 and the *Computational Life Sciences – Ticino in Rete* project. I thank Denis Mauá and the anonymous reviewers for their suggestions.

References

[Bodlaender, 1993] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proc. of STOC*, pages 226–234, 1993. ACM.

[Darwiche, 2009] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge Univ. Press, 2009.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Papadimitriou and Yannakakis, 2000] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proc. of FOCS*, page 86–92, 2000. IEEE Computer Society.

[Park and Darwiche, 2004] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *J. Artif. Intell. Res.*, 21:101–133, 2004.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Shenoy, 1996] Prakash P. Shenoy. Binary join trees. In *Proc. of UAI*, pages 492–499, 1996.

[Vazirani, 2001] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.