On the Complexity of the MPA Problem in Probabilistic Networks

Hans L. Bodlaender[1], **Frank van den Eijkhof** [1], and **Linda C. van der Gaag**[1]

Abstract. The problem of finding the best explanation for a set of observations is studied within various disciplines of artificial intelligence. For a probabilistic network, finding the best explanation amounts to finding a value assignment to all the variables in the network that has highest posterior probability given the available observations. This problem is known as the *MPA*, or maximum probability assignment, *problem*. In this paper, we establish the computational complexity of the MPA problem and of various closely related problems. Among other results, we show that, while the MPA-p problem, where an assignment with probability at least p is to be found, is NP-hard, its fixed-parameter variant is solvable in linear time.

1 INTRODUCTION

The problem of finding the best explanation for a set of observations is being addressed in various disciplines of artificial intelligence. Within the disciplines of model-based diagnosis and abduction especially, the problem is studied from different perspectives. The best explanation can for example be defined as a minimum-cardinality explanation, that is, composed of a minimal number of hypotheses, or as an irredundant explanation, that is, an explanation that is minimal with respect to set inclusion. The problem of finding the best explanation is well studied in these disciplines and, for different types of explanation, the computational complexity of the problem has been established [1].

The problem is also studied within the discipline of probabilistic networks. A probabilistic network is a concise representation of a joint probability distribution on a set of statistical variables [2]. It is comprised of a graphical structure, encoding the variables and their interdependences, and an associated set of conditional probability distributions. The graphical structure and associated distributions uniquely define a joint probability distribution over the represented variables, thereby enabling the computation of any probability of interest. For a probabilistic network, the best explanation for a set of observations is defined as a value assignment to the network's variables that has highest posterior probability given the available observations, that is, the best explanation is a most likely one. The problem of finding such an explanation is known as the *MPA*, or maximum probability assignment, *problem*.

While the computational complexity of finding the best explanation has been established for different types of explanation within the disciplines of model-based diagnosis and abduction, few complexity results are yet available for the MPA problem in probabilistic networks. S. Shimony has proved the MPA problem to be NP-hard, building upon a transformation from the VERTEXCOVER problem [3]; A. Abdelbar and S. Hedetniemi have extended this result to approximation of the MPA problem [4]. In this paper, we once again address the computational complexity of the MPA problem, this time building upon a transformation from the 3-SATISFIABILITY problem. This transformation allows us to establish complexity results also for various other problems that are closely related to the MPA problem. More specifically, we establish NP-hardness of the MPA-p problem which is the problem, given a probability p, to find an assignment with probability at least p. We further show that, while the MPA-p problem is NP-hard, its *fixed-parameter* variant where an assignment with probability at least p is to be found for a *fixed* rational number p, is solvable in linear time.

The paper is organised as follows. In Section 2 we provide some preliminaries on probabilistic networks. In Section 3 we formally define the MPA problem and various related problems. In Section 4 we provide a general construct with which we establish the computational complexity of each of the problems defined in Section 3. In Section 5, we address the complexity of the fixed-parameter variant of the MPA-p problem and present a linear algorithm for solving it. The paper ends with our concluding observations in Section 6.

2 PRELIMINARIES

A *probabilistic network* is a representation of a joint probability distribution on a set of statistical variables. Before defining the concept of probabilistic network more formally, we introduce some notational conventions. Statistical variables are denoted by capital letters with a subscript, such as V_i. In the sequel, we assume all variables V_i to be binary, taking one of the values v_i and $\overline{v}_i$, representing V_i = *true* and V_i = *false* respectively; generalisation to variables with more than two discrete values is straightforward. A conjunction of values for a set of variables W is termed an *assignment* to W, written a_W; the assignment to the empty set of variables equals *true*. We say that two assignments a_V and a_W to the sets of variables V and W respectively, are *consistent* if a_V and a_W have the same values for the variables in $V \cap W$. In mathematical formulas, we will write W to express that the formula holds for all assignments to W.

A probabilistic network now is a tuple $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is a directed acyclic graph and Γ is a set of conditional probability distributions. In the digraph G, each vertex V_i models a statistical variable. The set of arcs of G captures probabilistic independence. Two variables V_i and V_j are independent given a set of variables W, if either V_i or V_j is in W, or if every chain between V_i and V_j in G contains a variable from W with at least one emanating arc or a variable V_k with two incoming arcs such that neither V_k itself nor any of its descendants are in W. For a topological sort of the digraph G, that is, for an ordering $V_1, \ldots, V_n$ of the variables in

[1] Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. E-mail: {hansb,frankvde,linda}@cs.uu.nl

G with $i < j$ for every arc $(V_i, V_j) \in A(G)$, we now have that any variable V_i is independent of the preceding variables $V_1, \ldots, V_{i-1}$ given its set of parents $\pi(V_i)$. Associated with the digraph G is a set Γ of distributions: for each variable V_i are specified the conditional probability distributions $\Pr(V_i \mid \pi(V_i))$ that describe the influence of the various assignments to the variable's set of parents $\pi(V_i)$ on the probabilities of the values of V_i itself.

A probabilistic network $B = (G, \Gamma)$ uniquely defines a joint probability distribution $\Pr(V(G)) = \prod_{V_i \in V(G)} \Pr(V_i \mid \pi(V_i))$ that respects the independences portrayed by its digraph. Since it defines a unique probability distribution, a probabilistic network allows for the computation of any probability of interest over its variables [2].

3 DEFINITIONS

The concept of MPA underlies the problems that we address in this paper. We begin therefore by formally defining this concept.

Definition 3.1 *Let $B = (G, \Gamma)$ be a probabilistic network and let $\Pr$ be the joint probability distribution defined by B. Let O be the set of observed variables in B and let a_O denote the available observations; let $X = V(G) \backslash O$ be the set of unobserved variables in B. An* MPA *given a_O is an assignment a_X to X such that for all a'_X with $a'_X \neq a_X$, the property $\Pr(a'_X \wedge a_O) \leq \Pr(a_X \wedge a_O)$ holds. The probability of an MPA given a_O is termed the* MAP*, or maximum aposteriori probability, given a_O.*

In the sequel, we will often write MPA, and thus omit the available observations a_O from the notation, as long as ambiguity cannot occur. We now define the MPA problem and various closely related problems.

Definition 3.2 *Let $B = (G, \Gamma)$ be a probabilistic network and let $\Pr$ be its joint probability distribution. Let O, a_O and X be as above. We state the following problems, each having B, X and a_O as (part of) its input.*

a. The MPA *problem is the problem to find an MPA.*

b. The IS-AN-MPA *problem is the problem, given an assignment a_X to X, to decide whether a_X is an MPA.*

c. The IS-NOT-AN-MPA *problem is the problem, given an assignment a_X to X, to decide whether a_X is not an MPA.*

d. The MPA-p *problem is the problem, given a probability $0 \leq p \leq 1$, to find an assignment a_X to X with $\Pr(a_X \wedge a_O) \geq p$.*

e. The IS-AN-MPA-p *problem is the problem, given a probability p with $0 \leq p \leq 1$ and an assignment a_X to X, to decide whether $\Pr(a_X \wedge a_O) \geq p$.*

f. The $\exists$-MPA-p *problem is the problem, given a probability p with $0 \leq p \leq 1$, to determine whether there exists an assignment a_X to X with $\Pr(a_X \wedge a_O) \geq p$.*

g. The BETTER-MPA *problem is the problem, given an assignment a'_X to X, to find an assignment a_X to X such that $\Pr(a_X \wedge a_O) > \Pr(a'_X \wedge a_O)$.*

h. The MAP *problem is the problem to find the maximum probability p for which there exists an assignment a_X with $\Pr(a_X \wedge a_O) = p$.*

The IS-NOT-AN-MPA problem defined above is of interest just for intermediate steps in the proofs in the sequel. Note that the MPA-p, IS-AN-MPA-p and $\exists$-MPA-p problems have the probability p for their input, in addition to the parameters B, X and a_O. Further note that all problems mentioned above are defined for the set X of *all* unobserved variables in a given probabilistic network. To conclude, we note that all problems can be easily reformulated to include probabilities that are conditional on the observations a_O. The MPA-p problem, for example, can be reformulated as the problem of finding an assignment a_X to X with $\Pr(a_X \mid a_O) \cdot \Pr(a_O) \geq p$, where the probability $\Pr(a_O)$ is a constant with respect to all possible assignments a_X.

Analogous to the problems defined above in which we build on *maximum* probabilities, we define the MINPA, IS-A-MINPA, IS-NOT-A-MINPA, MINPA-p, IS-A-MINPA-p, $\exists$-MINPA-p, BETTER-MINPA, and MINAP problems that build on *minimum* probabilities. Note that these problems have trivial solutions if the probability distribution involved is not strictly positive (a joint probability distribution $\Pr$ on a set V of statistical variables is *strictly positive* if, for all assignments a_V to V, $\Pr(a_V) = 0$ implies $a_V \equiv$ *false*).

4 COMPLEXITY OF THE PROBLEMS

We address the computational complexity of the various problems that we have defined in the previous section. More specifically, we show for each of these problems that it is polynomially solvable, NP-complete, coNP-complete, or NP-hard. For further reading on these and other complexity classes, we refer to [5]. In Section 4.1 we briefly address the probability representation that underlies our complexity results. Section 4.2 describes a method for constructing a probabilistic network from an instance of the 3-SATISFIABILITY problem. In Section 4.3 we then use this construct to establish the computational complexity of the problems introduced in Section 3.

4.1 Probability representation

An ill-chosen representation of the probabilities of a probabilistic network may obscure the true complexity of the problems defined before. To illustrate this observation, we consider a representation in which for each probability a (Turing machine) program exists that produces the next decimal digit in polynomial time. With this representation we can show for most problems from Definition 3.2 that they are undecidable. We consider, as an example, the IS-AN-MPA problem for a probabilistic network B with a single vertex. Let M be a Turing machine with an empty input tape; from M we construct a program as indicated above. For integers i, we let $\text{RUN}(M, i)$ be true if and only if running M did not halt before step $i+1$; we further define

$$r = 4 \cdot 10^{-1} + \sum_{i>1, \text{RUN}(M,i)} 9 \cdot 10^{-i}$$

If M halts in step 4, for example, we find $r = 0.499$. In general, if M halts, we have for its output r that $r < 0.5$. If M does not halt, then $r = 0.49999\cdots = 0.5$. From M, we can build a Turing machine that creates a next digit of r in a constant number of steps. Now consider the probabilistic network $B = (\{X\}, \{\Pr(x) = r\})$. It is readily seen that the assignment $a_X = x$ is an MPA for B if and only if M does not halt. With this representation, therefore, the IS-AN-MPA problem is as least as hard as the HALTING problem, which is known to be undecidable.

In the remainder of this paper we assume that any probability specified in the set Γ of a probabilistic network is given as a fraction of two integers, both of which are expressed in finite binary notation. We find this assumption to be reasonable, as a network's probabilities are often assessed by experts or are estimated as proportions

from data. With our representation, computing the probability of a given assignment to all variables and comparing two probabilities, for example, can be performed in polynomial time, which renders testing whether a specific problem from Definition 3.2 belongs to a given complexity class to be of polynomial complexity.

4.2 A construct for our proofs

In this section, we describe a method for constructing a probabilistic network from a given instance of the well-known 3-SATISFIABILITY problem; this construct constitutes the basis for several of the proofs presented in Section 4.3. Before describing the construct in detail, we formally define the 3-SATISFIABILITY problem. Let $U = \{X_1, \ldots, X_n\}$, $n \geq 1$, be a set of Boolean variables. Let $C = \{C_1, \ldots, C_m\}$, $m \geq 1$, be a set of clauses, where each clause is the disjunction of exactly three literals of the form X_i or $\overline{X}_i$, $X_i \in U$. The 3-SATISFIABILITY problem (U, C) now is the problem to assign truth values to the variables in U such that $\bigwedge_{C_i \in C} C_i \equiv \mathit{true}$. The 3-SATISFIABILITY problem is known to be NP-complete [5].

Given an instance (U, C) of the 3-SATISFIABILITY problem, we construct a probabilistic network $B_{(U,C)} = (G, \Gamma)$ as follows. For each Boolean variable X_i, we add a vertex X_i to the vertex set $V(G)$. For each clause $C_j \in C$, we add a vertex C_j to $V(G)$. For all vertices $X_i, C_j \in V(G)$, we add an arc (X_i, C_j) to the arc set $A(G)$ of the digraph if and only if the Boolean variable X_i occurs in a literal of the clause C_j. We further add two auxiliary vertices Y and D; the purpose of these vertices will be explained presently. To conclude we add arcs (Y, C_j), for all $C_j \in V(G)$, as well as the arc (Y, D) to $A(G)$. Figure 1 depicts the digraph that is thus constructed from an example instance.

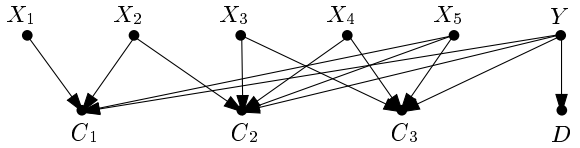

Figure 1. The digraph constructed from the instance $(U, C) = (\{X_1, \ldots, X_5\}, \{\{X_1, \overline{X_2}, X_5\}, \{\overline{X_2}, X_3, \overline{X_4}\}, \{\overline{X_3}, X_4, X_5\}\})$ of the 3-SATISFIABILITY problem.

To conclude the construction of the network $B_{(U,C)}$, we associate with the digraph G a set Γ of probability distributions. For each (root) vertex $X_i \in V(G)$, we set $\Pr(x_i) = \frac{1}{2}$. We further set $\Pr(y) = \frac{1}{2}$ for the vertex Y. For each vertex $C_j \in V(G)$, we observe that its set of parents $\pi(C_j)$ consists of Y and the three vertices X_i that occur in the literals of the clause C_j; we set

$$\Pr(c_j \mid a_{\pi(C_j)}) = \begin{cases} \frac{3}{4} & \text{if } C_j \text{ is satisfied by } a_{\pi(C_j)\setminus\{Y\}} \\ \frac{3}{4} & \text{if } a_{\pi(C_j)} \text{ has } Y = \mathit{true} \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

With this definition, the following property holds: given an assignment a_X to all vertices X_i, we have for a vertex C_j corresponding with a satisfied clause, that $\Pr(c_j) = \frac{3}{4}$; for a vertex C_j that corresponds with an unsatisfied clause, we have the lower probability $\Pr(c_j) = \frac{1}{2}$, that is, provided that $a_Y \neq y$. The conditional probability distributions for vertex C_j thus in essence capture satisfaction of the clause C_j. With

$$\Pr(d \mid a_Y) = \begin{cases} \frac{1}{2} & \text{if } a_Y = y \\ \frac{3}{4} & \text{otherwise} \end{cases}$$

for the vertex $D \in V(G)$, we now enforce easy identification of an assignment a, with $a_Y = y$, that has the following property: if the instance (U, C) of the 3-SATISFIABILITY problem does not have a solution, then a is an MPA for the constructed network; otherwise, a is the second-best assignment. The following proposition, given without proof due to space limitations, summarises the main properties of the thus constructed network $B_{(U,C)}$.

Lemma 4.1 *Let (U, C) be an instance of the* 3-SATISFIABILITY *problem with the set of Boolean variables $U = \{X_1, \ldots, X_n\}$, $n \geq 1$, and the set of clauses $C = \{C_1, \ldots, C_m\}$, $m \geq 1$. Let $B_{(U,C)} = (G, \Gamma)$ be the probabilistic network that results from the construction described above. Then, the following properties hold for the probability distribution* Pr *defined by $B_{(U,C)}$.*

a. The probability of any assignment $a_{V(G)}$ to $V(G)$ equals $\Pr(a_{V(G)}) = (\frac{1}{2})^{n+1+i} \cdot (\frac{3}{4})^{j} \cdot (\frac{1}{4})^{m+1-i-j}$, for some i, j, with $i \geq 0$, $j \geq 0$, $m + 1 - i - j \geq 0$.

b. The probability of the assignment $a_{V(G)}$ with $V_i = \mathit{true}$ for all $V_i \in V(G)$ equals $\Pr(a_{V(G)}) = (\frac{1}{2})^{n+2} \cdot (\frac{3}{4})^{m}$.

c. The probability of the assignment $a_{V(G)}$ with $Y = \mathit{true}$ and $V_i = \mathit{false}$ for all $V_i \in V(G)\setminus\{Y\}$ equals $\Pr(a_{V(G)}) = (\frac{1}{2})^{n+2} \cdot (\frac{1}{4})^{m}$.

d. There exists an assignment $a_{V(G)}$ to $V(G)$ with probability $\Pr(a_{V(G)}) = (\frac{1}{2})^{n+1} \cdot (\frac{3}{4})^{m+1}$ if and only if (U, C) has a solution.

e. There exists an assignment $a_{V(G)}$ to $V(G)$ with probability $\Pr(a_{V(G)}) = (\frac{1}{2})^{n+1} \cdot (\frac{1}{4})^{m+1}$ if and only if (U, C) has a solution.

4.3 The complexity results

Building upon the construct presented in the previous section, we establish the computational complexity for each of the problems defined in Section 3.

Proposition 4.2 *The* IS-NOT-AN-MPA *problem is NP-complete.*

Proof. We begin by showing that the IS-NOT-AN-MPA problem belongs to the complexity class NP. Given the input assignment a_X, we non-deterministically select another assignment a'_X to the set X of unobserved variables. Checking whether $\Pr(a_X \wedge a_O) < \Pr(a'_X \wedge a_O)$, that is, checking whether a_X is not an MPA, can be performed in polynomial time.

To prove NP-hardness, we construct a transformation from the 3-SATISFIABILITY problem. Let (U, C) be an instance of this problem and let $B_{(U,C)}$ be the probabilistic network that is constructed from (U, C) as described in Section 4.2. Now, let $a_{V(G)}$ be the assignment to $V(G)$ with $V_i = \mathit{true}$ for all $V_i \in V(G)$. From Lemma 4.1.*b.*, we know that $\Pr(a_{V(G)}) = (\frac{1}{2})^{n+2} \cdot (\frac{3}{4})^{m}$. If (U, C) does not have a solution, then $a_{V(G)}$ is an MPA by the properties *a.*, *b.*, and *d.* from Lemma 4.1. If (U, C) does have a solution, then we have from Lemma 4.1.*d.* that there is an assignment different from $a_{V(G)}$ with a larger probability, that is, $a_{V(G)}$ is not an MPA. We conclude that (U, C) has a solution if and only if $a_{V(G)}$ is not an MPA for the constructed network. As $B_{(U,C)}$ and $a_{V(G)}$ can be computed from (U, C) in polynomial time, we have a polynomial-time transformation from 3-SATISFIABILITY to the IS-NOT-AN-MPA problem, from which we conclude NP-hardness of the latter. □

Analogous to the proof of the previous lemma, we can show that also the IS-NOT-A-MINPA problem is NP-complete.

As we have argued before in Section 4.1, our representation of the probabilities of a probabilistic network enables us to compute the probability of a given assignment to all variables in the network in polynomial time; comparing two probabilities has the same complexity. From this observation we have the following proposition; a similar property holds for the IS-A-MINPA-p problem.

Proposition 4.3 *The* IS-AN-MPA-p *problem can be solved in polynomial time.*

Building upon the previous two propositions, we now state complexity results for the remaining six problems from Definition 3.2.

Proposition 4.4 *The following properties hold.*

a. The IS-AN-MPA *problem is coNP-complete.*

b. The $\exists$-MPA-p *problem is NP-complete.*

c. The MPA, MPA-p, BETTER-MPA, *and* MAP *problems are NP-hard.*

Proof. Due to space limitations, we restrict our proof to the first two properties.

a. We observe that the IS-AN-MPA problem is the complement of the IS-NOT-AN-MPA problem. From Proposition 4.2 and the definition of coNP-completeness, we have that the IS-AN-MPA problem is coNP-complete.

b. We begin by showing that the $\exists$-MPA-p problem belongs to the class NP. Given the input probability p, we non-deterministically select an assignment a_X to the set X of unobserved variables. From Proposition 4.3 we have that checking whether $\Pr(a_X \wedge a_O) \geq p$ takes polynomial time.

To prove NP-hardness, we construct a transformation from the 3-SATISFIABILITY problem. Let (U, C) be an instance of this problem and let $B_{(U,C)}$ be the probabilistic network that is constructed from (U, C) as described before. Now, let $p = (\frac{1}{2})^{n+1} \cdot (\frac{3}{4})^{m+1}$. From Lemma 4.1.*d*., we have that there is an assignment with probability at least p if and only if (U, C) has a solution. As $B_{(U,C)}$ and p can be computed from (U, C) in polynomial time, we have a polynomial-time transformation from 3-SATISFIABILITY to the $\exists$-MPA-p problem, from which we conclude NP-hardness of the latter. □

Analogous to the previous proposition, we can formulate similar complexity results for the MINPA, IS-A-MINPA, $\exists$-MINPA-p, MINPA-p, BETTER-MINPA, and MINAP problems. Note that Lemma 4.1.*e*. provides for the proof of NP-completeness of the $\exists$-MINPA-p problem.

5 FIXED-PARAMETER VARIANTS

In the previous section, we have shown that the $\exists$-MPA-p problem is NP-complete and that the MPA-p problem is NP-hard. We have studied these problems in a variant where the probability p is part of their input. In this section, we again address the two problems, this time in a variant where the probability p is no longer part of the input but a *fixed* rational number. For further reading on the fixed-parameter complexity of problems, we refer the reader to [6]. We show that the fixed-parameter variants of the two problems mentioned above can be solved in linear time.

To prove our complexity results for the two fixed-parameter problems, we begin by considering the special situations where the probability p equals either zero or one. As any assignment to a network's variables has a probability at least 0, the fixed-parameter variant of the $\exists$-MPA-p problem for $p = 0$ can be decided in the affirmative, even without inspecting the problem's input. For $p = 1$, a simple linear-time algorithm is readily constructed to test if there is a (necessarily unique) assignment with probability 1. We now proceed by constructing an algorithm for the $\exists$-MPA-p problem, for $0 < p < 1$, that uses a branch-and-bound technique for searching the space of all possible assignments to a network's variables. The algorithm constructs a search tree by recursively assigning appropriate values to the variables in the network, in the order of a topological sort of its digraph. Branches of the tree are pruned dynamically with the help of a simple criterion that verifies whether or not it is possible to extend the assignment associated with the branch to an assignment to all variables with a large enough probability. We first describe the algorithm and then address its correctness and running time.

Our algorithm uses the recursive procedure E-MPA-P. This procedure takes for its input a probabilistic network B, the set O of observed variables along with their assignment a_O, an array A, a probability p, and an integer i. The array $A[1 \cdots n]$ is used to store values for the variables in the network; the element $A[i]$ stores a value for the variable V_i. We assume, without loss of generality, that in the first call to the procedure the elements of the array A are initialised at *true*. The input parameter i denotes the level in the search tree that is currently being investigated; at level i, the search process has fixed the values for the variables $V_1, \ldots, V_i$. In the first call to the procedure, the parameter i is initialised at 0. The following pseudocode now summarises our algorithm.

procedure E-MPA-P(B, O, a_O, A, p, i)
  if $p > 1$ **then return** *false* **endif**;
  if $i = n$ **then return** *true* **endif**;
  compute $q = \Pr(v_{i+1} \mid V_1 = A[1], \ldots, V_i = A[i])$;
  if $q \neq 0$ and ($V_{i+1} \notin O$ or ($V_{i+1} \in O$ and $V_{i+1} =$ *true*))
  then $A[i+1]$:= *true*;
    if E-MPA-P($B, O, a_O, A, p/q, i+1$)
    then return *true*
    endif;
  endif;
  if $q \neq 1$ and ($V_{i+1} \notin O$ or ($V_{i+1} \in O$ and $V_{i+1} =$ *false*))
  then $A[i+1]$:= *false*;
    if E-MPA-P($B, O, a_O, A, p/(1-q), i+1$)
    then return *true*
    endif;
  endif;
  return *false*.

Note that the probability q computed by the algorithm is readily obtained from the set of probability distributions Γ of the probabilistic network under consideration, since we have that $\Pr(V_i \mid V_1, \ldots, V_{i-1}) = \Pr(V_i \mid \pi(V_i))$ for all variables V_i.

Our algorithm correctly solves the $\exists$-MPA-p problem, as is stated in the following lemma.

Lemma 5.1 *Let the procedure* E-MPA-P *be as defined above. For each i, the (recursive) call* E-MPA-P(B, O, a_O, A, p, i) *returns the value true if and only if there is an assignment $a_{V(G)}$ to $V(G)$ that satisfies the following three conditions:*

a. $\Pr(a_{V(G)}) \geq p \cdot \prod_{j=1,\ldots,i} \Pr\left(V_j = A[j] \mid \bigwedge_{k=1,\ldots,j-1} V_k = A[k]\right)$;

b. for all $j = 1, \ldots, i$, $a_{V(G)}$ is consistent with $V_j = A[j]$;

c. $a_{V(G)}$ is consistent with a_O.

Proof. To prove the property stated in the lemma we use downwards induction on i. As the lemma is clearly correct for $p > 1$, we assume in the remainder of the proof that $p \leq 1$.

Now consider, for our induction basis, the situation where $i = n$. The call to E-MPA-P then outputs the value *true*. We show that an assignment $a_{V(G)}$ to $V(G)$ exists that satisfies the three conditions stated in the lemma. Let $a_{V(G)}$ be an assignment such that the second and third condition are satisfied, that is, $a_{V(G)}$ is consistent with a_O and, for all $j = 1, \ldots, n$, $a_{V(G)}$ is consistent with $V_j = A[j]$. Then, from the distribution Pr defined by the network we have that

$$\Pr(a_{V(G)}) = \prod_{j=1,\ldots n} \Pr\left(V_j = A[j] \mid \bigwedge_{k=1,\ldots,j-1} V_k = A[k]\right)$$

With $p \leq 1$, we conclude that an assignment satisfying the three conditions exists.

Now consider the situation where $i < n$. Suppose that there is an assignment $a_{V(G)}$ to $V(G)$ that satisfies the three conditions stated in the lemma. We show that the call E-MPA-P(B, O, a_O, A, p, i) returns the value *true*. We distinguish between V_{i+1} having the value *true* in $a_{V(G)}$, and V_{i+1} having the value *false*. If $V_{i+1} =$ *true* in $a_{V(G)}$, then the procedure finds the value $q = \Pr(v_{i+1} \mid V_1 = A[1], \ldots, V_i = A[i]) > 0$. In the recursive call E-MPA-P$(B, O, a_O, A, p/q, i + 1)$, we then have that

$$\Pr(a_{V(G)}) \geq p \cdot \prod_{j=1,\ldots,i} \Pr\left(V_j = A[j] \mid \bigwedge_{k=1,\ldots,j-1} V_k = A[k]\right)$$

$$= (p/q) \cdot \prod_{j=1,\ldots i+1} \Pr\left(V_j = A[j] \mid \bigwedge_{k=1,\ldots,j-1} V_k = A[k]\right)$$

Using induction, we know that this recursive call will return *true*. Similarly, when $a_{V(G)}$ has $V_{i+1} =$ *false*, the alternative recursive call from the procedure will return the value *true*. Conversely, it is readily shown, again by induction, that if either of these recursive calls returns *true*, then an assignment $a_{V(G)}$ satisfying the conditions from the lemma exists. □

The following proposition now states that the fixed-parameter variants of the MPA-p and ∃-MPA-p problems are solvable in linear time. The proof of the properties builds upon the algorithm outlined above.

Proposition 5.2 *For every fixed rational number $p \in [0, 1]$, there exist algorithms for solving the* MPA-p *and* ∃-MINPA-p *problems that use $O(n)$ time on a probabilistic network with n variables.*

Proof. As mentioned before, the property stated in the proposition trivially holds for $p = 0$ and $p = 1$. We now show that, for every fixed rational number p with $0 < p < 1$, the E-MAP-P procedure for solving the ∃-MPA-p problem takes linear time. To this end, we investigate the search tree constructed by the procedure. A leaf of this tree is called *true*, if it returns the value *true*; otherwise, it is called *false*. A node of the search tree is called *branching*, if it has two children, none of which is a false leaf. With each node x, we associate the value of p from the corresponding call to the E-MPA-P procedure; this value is denoted p_x. Similarly, q_x denotes the value of q during the call. Let r be the root of the tree.

For a child y of a node x in the search tree, we have that $p_y \geq p_x$, as either $p_y = p_x/q_x$ and $0 < q_x \leq 1$, or $p_y = p_x/(1 - q_x)$ and $0 \leq q_x < 1$. For a branching node x, we have that $0 < q_x < 1$, $p_x/q_x \leq 1$ and $p_x/(1 - q_x) \leq 1$, from which we have $p_x \leq q_x$, $p_x \leq 1 - q_x$ and $q_x \leq 1 - p_x$. Now, suppose that y is a child of the branching node x. We have that either $p_y = p_x/q_x \geq p_x/(1-p_x) \geq p_x/(1 - p_r)$, or $p_y = p_x/(1 - q_x) \geq p_x/(1 - p_x) \geq p_x/(1 - p_r)$. With induction, it follows that, if there are α branching nodes on the path from the root r to a node z, then $p_z \geq p_r/(1 - p_r)^\alpha$. If this node z is a true leaf z, then $1 \geq p_z \geq p_r/(1 - p_r)^\alpha$, from which we conclude that $\alpha \leq \log p_r / \log(1 - p_r)$, which is a constant for a fixed rational p_r.

From the above observations we have that, if the probability p in the ∃-MINPA-p problem is a fixed rational number, then the number of branching nodes on a path from the root of the search tree to a true leaf is bounded by a constant, and hence the total number of branching nodes is bounded by a constant. The number of nodes in the tree, therefore, is linear in n where n is the number of variables of the input network. Every node of the tree corresponds to a recursive call to the E-MPA-P procedure in which the computations to be performed require constant time. For every fixed rational number p, therefore, the total time of running the algorithm thus is $O(2^{\log p/log(1-p)} \cdot n)$, that is, linear in the number of variables of the input network.

We observe that the MPA-p problem can be solved by the E-MPA-P procedure after a minor modification: instead of returning the value *true*, the procedure must return the corresponding assignment. The above proof therefore easily extends to the fixed-parameter variant of the MPA-p problem. □

6 CONCLUSIONS

We have addressed the computational complexity of various problems for probabilistic networks. We have shown, for example, that the ∃-MPA-p problem is NP-complete and that the MPA-p problem is NP-hard. In addition, we have established the computational complexity of various other problems that are closely related to these two problems. Our results show that MPA-related problems are hard to solve, indicating that only heuristic algorithms can serve to feasibly solve them in general. Special instances of the problems, however, may be more readily solvable. This observation holds, for example, for probabilistic networks with certain restricted topologies [7]. Here we have shown that also the fixed-parameter variants of the two problems mentioned above are easy to solve: we have shown that these problems can be solved in linear time.

Acknowledgement. This research has been (partly) supported by the Netherlands Organisation for Scientific Research (NWO).

REFERENCES

[1] T. Bylander, D. Allemang, M.C. Tanner, and J.R. Josephson, 'The computational complexity of abduction', *Artificial Intelligence*, **49**, 25 – 60, (1991).
[2] F.V. Jensen, *Bayesian Networks and Decision Graphs*, Statistics for Engineering and Information Science, Springer-Verlag, New York, 2001.
[3] S.E. Shimony, 'Finding MAPs for belief networks is NP-hard', *Artificial Intelligence*, **68**, 399 – 410, (1994).
[4] A.M. Abdelbar and S.M. Hedetniemi, 'Approximating MAPs for belief networks is NP-hard and other theorems, *Artificial Intelligence*, **102**, 21 – 38, (1998).
[5] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.
[6] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer, 1998.
[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*, Morgan Kaufmann, Palo Alto, 1988.