

Robustifying the Viterbi Algorithm

Cedric De Boom, Jasper De Bock, Arthur Van Camp, and Gert de Cooman

SYSTeMS Research Group, Ghent University,
Technologiepark 914, 9052 Zwijnaarde, Belgium
{Cedric.DeBoom, Jasper.DeBock, Arthur.VanCamp, Gert.deCooman}@UGent.be

Abstract. We present an efficient algorithm for estimating hidden state sequences in imprecise hidden Markov models (iHMMs), based on observed output sequences. The main difference with classical HMMs is that the local models of an iHMM are not represented by a single mass function, but rather by a set of mass functions. We consider as estimates for the hidden state sequence those sequences that are maximal. In this way, we generalise the problem of finding a state sequence with highest posterior probability, as is commonly considered in HMMs, and solved efficiently by the Viterbi algorithm. An important feature of our approach is that there may be multiple maximal state sequences, typically for iHMMs that are highly imprecise. We show experimentally that the time complexity of our algorithm tends to be linear in this number of maximal sequences, and investigate how this number depends on the local models.

Keywords: imprecise hidden Markov model, Viterbi algorithm, maximality, hidden state sequence, robustness

1 Introduction

The popularity of Bayesian networks has increased rapidly over the last decades, and their power has been illustrated in numerous applications. Nevertheless, some of the assumptions they are based on are rather severe and, in some cases, even unreasonable. For example, in order to specify a Bayesian network, one has to quantify its local probability mass functions exactly. If limited data and/or expert knowledge is available, this is clearly an unrealistic requirement. By enforcing precision nevertheless, the resulting model and the inferences it produces are, although precise, not guaranteed to be supported by the evidence, thereby creating a false sense of correctness.

In order to avoid this problem, one can allow for local models that are represented by a set of mass functions instead of a single one, thereby obtaining a so-called *credal network* [1]. In this paper, we will consider the special case of an *imprecise hidden Markov model* (iHMM), which is the credal network version of an HMM. We explain how the problem of finding a state sequence with maximal posterior probability can be generalised to this framework, and present an algorithm that is capable of solving this new version of the problem in an efficient manner. In this way, we obtain a robust alternative to the *Viterbi algorithm* [5].

A similar study has been conducted in Ref. [2] as well. However, the iHMM that was considered in that paper was of a completely different kind: instead of regarding an iHMM as a collection of HMMs—as we will do, and as is technically referred to as assuming ‘strong independence’—the authors of Ref. [2] considered a so-called iHMM under epistemic irrelevance; see Ref. [1] for more information. In the conclusions of Ref. [2], the authors wondered whether or not it was possible to obtain similar results for our version of an iHMM. The present paper illustrates that this is indeed the case.

We start in Section 2 by introducing HMMs, also discussing the problem that is solved by the Viterbi algorithm. In Section 3, we generalise this problem towards *imprecise* hidden Markov models, which we introduce, and explain how it leads us to consider a set of maximal sequences as estimates for the hidden state sequence. We derive a manageable expression for this set in Section 4, and use it in Section 5 to derive an algorithm that is able to calculate the set of all maximal sequences in a recursive manner. In Section 6, we explain how for some common imprecise models, the parameters that are required to run our algorithm can be calculated easily. We end the paper in Section 7 by presenting a number of experiments, showing that the time complexity of our algorithm tends to be linear in the number of maximal sequences, and illustrating how this number depends on the local models of the iHMM.

2 Hidden Markov Models

A hidden Markov model (HMM) is a probabilistic graphical model that has a graphical structure of the form depicted in Figure 1.

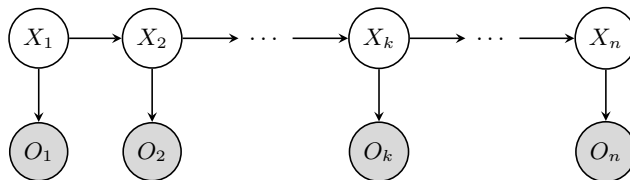


Fig. 1. Graphical structure of a hidden Markov model.

It consists of $2n$ random variables that can be categorised into n hidden state variables X_1, X_2, \dots, X_n and n observable output variables O_1, O_2, \dots, O_n . For any given k in $\{1, \dots, n\}$, the variables X_k and O_k take values in their respective possibility space \mathcal{X}_k and \mathcal{O}_k . We assume that every possibility space is finite.

2.1 Local Uncertainty Models

For the first state variable X_1 , we have an *initial model* that, since \mathcal{X}_1 is assumed to be finite, can be characterised by a probability mass function p_1 on \mathcal{X}_1 . For any x_1 in \mathcal{X}_1 , $p_1(x_1)$ is the probability that X_1 takes the value x_1 . For the subsequent state variables X_k , with k in $\{2, \dots, n\}$, we have a *transition model* p_k . For any

x_k in \mathcal{X}_k and x_{k-1} in \mathcal{X}_{k-1} , $p_k(x_k|x_{k-1})$ is the probability that X_k assumes the value x_k , conditional on X_{k-1} being equal to x_{k-1} . For notational convenience, we introduce a trivial state variable X_0 that assumes only a single value \square ; hence, $\mathcal{X}_0 := \{\square\}$ and, whenever we write x_0 , this is taken to be equal to \square . This trick allows us to regard p_1 as a conditional model as well, by defining $p_1(x_1|\square) := p_1(x_1)$. Finally, for every output variable O_k , with k in $\{1, \dots, n\}$, we have an *emission model* q_k . For every o_k in \mathcal{O}_k and x_k in \mathcal{X}_k , it provides us with the conditional probability $q_k(o_k|x_k)$ that O_k assumes the value o_k , given that X_k is equal to x_k .

2.2 Constructing a Joint Model

By imposing the usual Markov condition for Bayesian networks, the global model of an HMM—a global mass function p —is completely determined by its local models; it suffices to multiply them. For all $x_{1:n}$ in $\mathcal{X}_{1:n} := \times_{k=1}^n \mathcal{X}_k$ and $o_{1:n}$ in $\mathcal{O}_{1:n} := \times_{k=1}^n \mathcal{O}_k$, we find that

$$p(x_{1:n}, o_{1:n}) = \prod_{k=1}^n p_k(x_k|x_{k-1})q_k(o_k|x_k),$$

where we use the shorthand notations $x_{1:n} := (x_1, \dots, x_n)$ and $o_{1:n} := (o_1, \dots, o_n)$ to refer to the state and output sequence, respectively.

2.3 The Viterbi Algorithm

One of the most important problems in an HMM is to try and estimate the unknown hidden state sequence, based on an observed output sequence $o_{1:n}$ in $\mathcal{O}_{1:n}$. This is commonly done by choosing a state sequence $x_{1:n}$ that maximises the posterior probability $p(x_{1:n}|o_{1:n})$, as obtained through Bayes' rule. We will call such a state sequence *optimal*. Assuming that $p(o_{1:n})$ is positive, we find that the set of all optimal sequences is equal to

$$\arg \max_{x_{1:n} \in \mathcal{X}_{1:n}} p(x_{1:n}, o_{1:n}) = \arg \max_{x_{1:n} \in \mathcal{X}_{1:n}} \prod_{k=1}^n p_k(x_k|x_{k-1})q_k(o_k|x_k). \quad (1)$$

A well-known method for finding an arbitrary element of this set—and hence an estimate for $x_{1:n}$ —is to apply the Viterbi algorithm [5]; see Ref. [3] for a good introduction. By proceeding in a recursive fashion, this algorithm manages to be very efficient: its time complexity is only $O(nm^2)$, where m is the size of the biggest possibility space for the states: $m := \max\{|\mathcal{X}_k| : k \in \{1, \dots, n\}\}$.

3 Imprecise Hidden Markov Models

In classical HMMs—the ones considered in the previous section—the local uncertainty models are mass functions, which have to be quantified exactly, say, with arbitrary precision. However, in many instances (e.g., if little data and/or

expert knowledge is available), this requirement is clearly unreasonable. For example, what if there is some probability for which an expert is only able to provide an interval, rather than an exact value? In order to model such situations in a more flexible manner, one can use a so-called *imprecise hidden Markov model* (iHMM) which, basically, is just a set of HMMs. They all share the same graphical structure, but their local probability mass functions may differ.

3.1 Local Imprecise Uncertainty Models

The crucial difference between iHMMs and HMMs is that their local models are not required to consist of a single mass function. Instead, a set of mass functions may be used. The only restrictions we impose is that this set should be *compact*, and that it should assign positive probability to every event; for ease of reference, we call this last requirement the *positivity assumption*.

In order to make this more formal, we denote by $\Sigma_{\mathcal{X}}$ the set of all mass functions on some generic possibility space \mathcal{X} and let $\text{int}\Sigma_{\mathcal{X}}$ be its interior, as defined by $\text{int}\Sigma_{\mathcal{X}} := \{p \in \Sigma_{\mathcal{X}} : (\forall x \in \mathcal{X}) p(x) > 0\}$. We allow our local models to be any compact subset \mathcal{M} of $\text{int}\Sigma_{\mathcal{X}}$. Compactness is imposed to ensure that minima and maxima are well-defined, thereby allowing us to consider, for all x in \mathcal{X} , its so-called *lower and upper probability*, as defined by

$$\underline{p}(x) := \min\{p(x) : p \in \mathcal{M}\} \quad \text{and} \quad \bar{p}(x) := \max\{p(x) : p \in \mathcal{M}\}.$$

The positivity assumption is imposed for didactical reasons only, as it allows us to avoid a number of cumbersome technicalities. In its most general form—which, due to page limit constraints, we are unable to discuss here—the algorithm we are about to introduce works for compact subsets of $\Sigma_{\mathcal{X}}$ as well.

We introduce the following notation for the different local uncertainty models of an iHMM. For X_1 , the initial model is denoted by \mathcal{M}_1 and can be any compact subset of $\text{int}\Sigma_{\mathcal{X}_1}$. Similarly, the transition model at position k , conditional on $X_{k-1} = x_{k-1}$, is a compact subset $\mathcal{M}_{X_k|x_{k-1}}$ of $\text{int}\Sigma_{\mathcal{X}_k}$. This set may be different for every x_{k-1} in \mathcal{X}_{k-1} . It will be convenient to refer to them collectively by means of the shorthand notation $\mathcal{M}_{X_k|X_{k-1}} := \times_{x_{k-1} \in \mathcal{X}_{k-1}} \mathcal{M}_{X_k|x_{k-1}}$. An element $p_k(\cdot|X_{k-1})$ of $\mathcal{M}_{X_k|X_{k-1}}$ is then a tuple, consisting of probability mass functions $p_k(\cdot|x_{k-1})$ in $\mathcal{M}_{X_k|x_{k-1}}$, one for every x_{k-1} in \mathcal{X}_{k-1} . For $k = 1$, we have that $\mathcal{M}_{X_1|X_0} = \mathcal{M}_{X_1|\square} := \mathcal{M}_1$. Finally, the emission model at position k , conditional on $X_k = x_k$, is a compact subset $\mathcal{M}_{O_k|x_k}$ of $\text{int}\Sigma_{\mathcal{O}_k}$. We write $\mathcal{M}_{O_k|X_k} := \times_{x_k \in \mathcal{X}_k} \mathcal{M}_{O_k|x_k}$ to refer to all the different conditional models for O_k at once.

3.2 Constructing an Imprecise Joint Model

By specifying these imprecise, set-valued local models, we also specify, in a very natural way, a corresponding family of joint probability mass functions

$$\mathcal{M} := \left\{ \prod_{k=1}^n p_k(X_k|X_{k-1})q_k(O_k|X_k) : \right. \\ \left. (\forall k \in \{1, \dots, n\}) p_k(\cdot|X_{k-1}) \in \mathcal{M}_{X_k|X_{k-1}}, q_k(\cdot|X_k) \in \mathcal{M}_{O_k|X_k} \right\}. \quad (2)$$

Every probability mass function p in \mathcal{M} corresponds to a different HMM, whose local probability mass functions are selected from the imprecise, set-valued local models that were discussed in the previous section. Together, these HMMs—and their joint mass functions—constitute an iHMM. Note that, since multiplication is a continuous operation, the compactness of the local imprecise models guarantees that \mathcal{M} is compact as well. Furthermore, since the local models satisfy the positivity assumption, \mathcal{M} satisfies it too.

3.3 Generalising the Notion of Optimality

Since we are now working with a set \mathcal{M} of joint mass functions rather than a single mass function p , the concept of ‘maximising posterior probability’ is no longer well-defined. Hence, we need to come up with some other way of estimating the hidden sequence $x_{1:n}$ based on an observed output sequence $o_{1:n}$; we need a new notion of optimality. Different imprecise-probabilistic decision criteria can be used for this purpose; see Ref. [4] for an overview. In the precise case—if \mathcal{M} is a singleton—all these approaches coincide with the one that is adopted in Section 2.3.

The approach that we will use here is to adopt the decision criterion of maximality [6]. The idea is to introduce a strict preference relation \succ between state sequences. For any two state sequences $x_{1:n}$ and $\hat{x}_{1:n}$ in $\mathcal{X}_{1:n}$, we say that $x_{1:n}$ is better than $\hat{x}_{1:n}$, and write $x_{1:n} \succ \hat{x}_{1:n}$, if

$$(\forall p \in \mathcal{M}) p(x_{1:n}|o_{1:n}) > p(\hat{x}_{1:n}|o_{1:n}), \quad (3)$$

This preference relation induces a strict partial order on the set of all state sequences $\mathcal{X}_{1:n}$, and we call a sequence $\hat{x}_{1:n}$ *maximal* if it is undominated in this partial order or, equivalently, if no other sequence is better. This leads us to consider as optimal sequences the elements of

$$\text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n}) := \{\hat{x}_{1:n} \in \mathcal{X}_{1:n} : (\forall x_{1:n} \in \mathcal{X}_{1:n}) x_{1:n} \not\succ \hat{x}_{1:n}\}. \quad (4)$$

4 A More Convenient Characterisation of Maximality

In its current form, our characterisation of maximality is—although intuitive—rather impractical. Therefore, as a first step in developing an efficient algorithm for calculating the maximal sequences, we set out to derive a more convenient expression for $\text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n})$.

4.1 Defining the Local Parameters

We start by introducing a number of important local parameters. As we will see, they are crucial to the developments in the remainder of this paper. For all k in $\{1, \dots, n\}$, o_k in \mathcal{O}_k , x_k and \hat{x}_k in \mathcal{X}_k , and x_{k-1} and \hat{x}_{k-1} in \mathcal{X}_{k-1} , we define

$$\omega_k(x_k, \hat{x}_k, o_k) := \min_{q_k(\cdot|\mathcal{X}_k) \in \mathcal{M}_{\mathcal{O}_k|\mathcal{X}_k}} \frac{q_k(o_k|x_k)}{q_k(o_k|\hat{x}_k)}$$

and

$$\chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1}) := \min_{p_k(\cdot|X_{k-1}) \in \mathcal{M}_{X_k|X_{k-1}}} \frac{p_k(x_k|x_{k-1})}{p_k(\hat{x}_k|\hat{x}_{k-1})}. \quad (5)$$

The following result establishes that, for most values of x_k , \hat{x}_k , x_{k-1} and \hat{x}_{k-1} , these parameters can be calculated easily.

Proposition 1. *The parameters $\omega_k(x_k, \hat{x}_k, o_k)$ and $\chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1})$ can be calculated easily in most instances:*

$$\omega_k(x_k, \hat{x}_k, o_k) = \begin{cases} 1 & \text{if } x_k = \hat{x}_k, \\ \frac{q_k(o_k|x_k)}{q_k(o_k|\hat{x}_k)} & \text{if } x_k \neq \hat{x}_k, \end{cases}$$

$$\chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1}) = \begin{cases} 1 & \text{if } x_k = \hat{x}_k \text{ and } x_{k-1} = \hat{x}_{k-1}, \\ \frac{p_k(x_k|x_{k-1})}{p_k(x_k|\hat{x}_{k-1})} & \text{if } x_{k-1} \neq \hat{x}_{k-1}. \end{cases}$$

The only exception—which is the case that is not covered by Proposition 1—is $\chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1})$, with $x_k \neq \hat{x}_k$ and $x_{k-1} = \hat{x}_{k-1}$. In general, this parameter will have to be calculated by performing the actual minimisation in Eq. (5), for example, by fractional linear programming techniques. However, for many commonly used local models, closed-form expressions are available even in this case; we will come back to this in Section 6.

4.2 Rewriting the Solution Set

As we are about to show, the local parameters that were just introduced allow us to greatly simplify Eq. (4). As a first step, we rewrite Eq. (3) in the following manner:

$$\begin{aligned} x_{1:n} \succ \hat{x}_{1:n} &\Leftrightarrow (\forall p \in \mathcal{M}) p(x_{1:n}, o_{1:n}) > p(\hat{x}_{1:n}, o_{1:n}) \\ &\Leftrightarrow (\forall p \in \mathcal{M}) \frac{p(x_{1:n}, o_{1:n})}{p(\hat{x}_{1:n}, o_{1:n})} > 1 \Leftrightarrow \min_{p \in \mathcal{M}} \frac{p(x_{1:n}, o_{1:n})}{p(\hat{x}_{1:n}, o_{1:n})} > 1, \end{aligned} \quad (6)$$

where the equivalences are a consequence of Bayes' rule, our positivity assumption, and the compactness of \mathcal{M} .

The nice thing about Eq. (6) is that the minimum at the right hand side can be easily calculated. Indeed, by exploiting the factorised form of the mass functions p in \mathcal{M} , splitting up the global minimum, and pushing the resulting individual minima inside, we find that

$$\begin{aligned} \min_{p \in \mathcal{M}} \frac{p(x_{1:n}, o_{1:n})}{p(\hat{x}_{1:n}, o_{1:n})} &= \min_{p \in \mathcal{M}} \prod_{k=1}^n \frac{p_k(x_k|x_{k-1}) q_k(o_k|x_k)}{p_k(\hat{x}_k|\hat{x}_{k-1}) q_k(o_k|\hat{x}_k)} \\ &= \prod_{k=1}^n \min_{p_k(\cdot|X_{k-1}) \in \mathcal{M}_{X_k|X_{k-1}}} \frac{p_k(x_k|x_{k-1})}{p_k(\hat{x}_k|\hat{x}_{k-1})} \min_{q_k(\cdot|X_k) \in \mathcal{M}_{O_k|X_k}} \frac{q_k(o_k|x_k)}{q_k(o_k|\hat{x}_k)} \\ &= \prod_{k=1}^n \chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1}) \omega_k(x_k, \hat{x}_k, o_k). \end{aligned} \quad (7)$$

It is now but a small step to reformulate Eq. (4). By combining Eqs. (6) and (7), we easily find that

$$\hat{x}_{1:n} \in \text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n}) \Leftrightarrow \max_{x_{1:n} \in \mathcal{X}_{1:n}} \prod_{k=1}^n \chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1}) \omega_k(x_k, \hat{x}_k, o_k) \leq 1, \quad (8)$$

where the maximum is trivially attained because $\mathcal{X}_{1:n}$ is a finite set.

For a given, fixed state sequence $\hat{x}_{1:n}$ in $\mathcal{X}_{1:n}$, calculating the maximum in Eq. (8) is a problem that is—formally—very closely related the one that is tackled by the Viterbi algorithm; compare Eqs. (1) and (8). Hence, it should not come as a surprise that, by Eq. (8), checking whether $\hat{x}_{1:n}$ is maximal can be done in an equally efficient way: $O(nm^2)$. It suffices to calculate the maximum in Eq. (8) in a recursive fashion.

5 A Recursive Algorithm

Although Eq. (8) already simplifies the problem of finding $\text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n})$, applying it directly is clearly not efficient enough. The main bottleneck is that it requires us to check the maximality of each individual state sequence separately. Since there are exponentially many such state sequences, this quickly becomes intractable. In order to avoid this exponential blow-up, we will now develop an algorithm that is able to rule out the maximality of many state sequences at once, without having to explicitly check the maximality of each of them individually.

5.1 Ruling out Multiple Sequences at Once

The central idea of our algorithm is to regard the set of all state sequences $\mathcal{X}_{1:n}$ as a search tree in which we can navigate while deciding whether a branch is useful or not to explore further. If we are able to infer that there is no maximal state sequence that starts with a given initial segment, then we can completely ignore all branches that start with this segment.

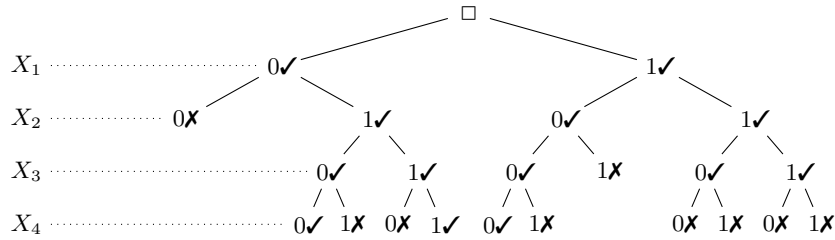


Fig. 2. Example of a search tree for binary sequences of length four.

Example 1. Consider the tree in Figure 2, which corresponds to $\mathcal{X}_{1:4}$, with binary local state spaces $\mathcal{X}_k := \{0, 1\}$. In the leftmost part of the tree, the initial segment ‘00’ is investigated. If we are able to infer that there is no maximal sequence that starts with ‘00’—for the sake of this example, we assume this is the case—then we can stop exploring the tree further in this direction and cut off the tree at the corresponding node. For the initial segment ‘101’, a similar situation occurs.

◆

In order to get this idea to work, it is crucial to have a simple check that allows us to conclude that none of the maximal state sequences in $\text{opt}(\mathcal{X}_{1:n}|o_{1:n})$ starts with some given initial segment. In other words, for any k in $\{1, \dots, n\}$ and $\hat{x}_{1:k}^*$ in $\mathcal{X}_{1:k}$, we need to be able to check if

$$(\forall \hat{x}_{1:n} \in \text{opt}(\mathcal{X}_{1:n}|o_{1:n})) \hat{x}_{1:k} \neq \hat{x}_{1:k}^*. \quad (\text{C1})$$

By Eq. (8), this is equivalent to checking whether

$$\begin{aligned} & (\forall \hat{x}_{k+1:n} \in \mathcal{X}_{k+1:n}) \max_{\substack{x_{1:n} \\ \in \mathcal{X}_{1:n}}} \prod_{i=1}^k \chi_i(x_i, x_{i-1}, \hat{x}_i^*, \hat{x}_{i-1}^*) \omega_i(x_i, \hat{x}_i^*, o_i) \\ & \quad \chi_{k+1}(x_{k+1}, x_k, \hat{x}_{k+1}, \hat{x}_k^*) \omega_{k+1}(x_{k+1}, \hat{x}_{k+1}, o_{k+1}) \\ & \quad \prod_{j=k+2}^n \chi_j(x_j, x_{j-1}, \hat{x}_j, \hat{x}_{j-1}) \omega_j(x_j, \hat{x}_j, o_j) > 1. \\ \Leftrightarrow & \min_{\substack{\hat{x}_{k+1:n} \\ \in \mathcal{X}_{k+1:n}}} \max_{\substack{x_{1:n} \\ \in \mathcal{X}_{1:n}}} \prod_{i=1}^k \chi_i(x_i, x_{i-1}, \hat{x}_i^*, \hat{x}_{i-1}^*) \omega_i(x_i, \hat{x}_i^*, o_i) \\ & \quad \chi_{k+1}(x_{k+1}, x_k, \hat{x}_{k+1}, \hat{x}_k^*) \omega_{k+1}(x_{k+1}, \hat{x}_{k+1}, o_{k+1}) \\ & \quad \prod_{j=k+2}^n \chi_j(x_j, x_{j-1}, \hat{x}_j, \hat{x}_{j-1}) \omega_j(x_j, \hat{x}_j, o_j) > 1. \quad (\text{C1}') \end{aligned}$$

The problem with criterion (C1') is that it is very difficult to calculate the maximum and minimum because they run over exponentially large spaces. In order to circumvent this issue, one can split the global maximum into local maxima that run over the individual states x_i in \mathcal{X}_i , and push these maxima inside. This leads to the equivalent condition

$$\begin{aligned} & \min_{\substack{\hat{x}_{k+1:n} \\ \in \mathcal{X}_{k+1:n}}} \max_{x_1 \in \mathcal{X}_1} \chi_1(x_1, \square, \hat{x}_1^*, \square) \omega_1(x_1, \hat{x}_1^*, o_1) \cdots \\ & \quad \max_{x_k \in \mathcal{X}_k} \chi_k(x_k, x_{k-1}, \hat{x}_k^*, \hat{x}_{k-1}^*) \omega_k(x_k, \hat{x}_k^*, o_k) \\ & \quad \max_{x_{k+1} \in \mathcal{X}_{k+1}} \chi_{k+1}(x_{k+1}, x_k, \hat{x}_{k+1}, \hat{x}_k^*) \omega_{k+1}(x_{k+1}, \hat{x}_{k+1}, o_{k+1}) \\ & \quad \max_{x_{k+2} \in \mathcal{X}_{k+2}} \chi_{k+2}(x_{k+2}, x_{k+1}, \hat{x}_{k+2}, \hat{x}_{k+1}) \omega_{k+2}(x_{k+2}, \hat{x}_{k+2}, o_{k+2}) \\ & \quad \cdots \max_{x_n \in \mathcal{X}_n} \chi_n(x_n, x_{n-1}, \hat{x}_n, \hat{x}_{n-1}) \omega_n(x_n, \hat{x}_n, o_n) > 1. \quad (\text{C1}'') \end{aligned}$$

By applying a similar procedure to the global minimum, we finally obtain the following inequality:

$$\begin{aligned}
& \max_{x_1 \in \mathcal{X}_1} \chi_1(x_1, \square, \hat{x}_1^*, \square) \omega_1(x_1, \hat{x}_1^*, o_1) \cdots \\
& \max_{x_k \in \mathcal{X}_k} \chi_k(x_k, x_{k-1}, \hat{x}_k^*, \hat{x}_{k-1}^*) \omega_k(x_k, \hat{x}_k^*, o_k) \\
& \min_{\hat{x}_{k+1} \in \mathcal{X}_{k+1}} \max_{x_{k+1} \in \mathcal{X}_{k+1}} \chi_{k+1}(x_{k+1}, x_k, \hat{x}_{k+1}, \hat{x}_k^*) \omega_{k+1}(x_{k+1}, \hat{x}_{k+1}, o_{k+1}) \\
& \min_{\hat{x}_{k+2} \in \mathcal{X}_{k+2}} \max_{x_{k+2} \in \mathcal{X}_{k+2}} \chi_{k+2}(x_{k+2}, x_{k+1}, \hat{x}_{k+2}, \hat{x}_{k+1}) \omega_{k+2}(x_{k+2}, \hat{x}_{k+2}, o_{k+2}) \\
& \cdots \min_{\hat{x}_n \in \mathcal{X}_n} \max_{x_n \in \mathcal{X}_n} \chi_n(x_n, x_{n-1}, \hat{x}_n, \hat{x}_{n-1}) \omega_n(x_n, \hat{x}_n, o_n) > 1. \tag{C2}
\end{aligned}$$

However, this criterion is not equivalent to the previous ones. By pushing the local minima inside and beyond the maxima, we obtain a number that, although it is guaranteed never to be bigger, might be lower than the number we started out from. Hence, criterion (C2) is only a *sufficient* condition for (C1) to hold.

Nevertheless, we prefer criterion (C2) over (C1) because it is easier to check. Indeed, if for all k in $\{1, \dots, n\}$, x_k in \mathcal{X}_k and $\hat{x}_{1:k}$ in $\mathcal{X}_{1:k}$, we consider the parameters $\gamma_k(x_k, \hat{x}_k)$ and $\delta_k(x_k, \hat{x}_{1:k})$, as defined recursively by

$$\gamma_k(x_k, \hat{x}_k) := \min_{\hat{x}_{k+1} \in \mathcal{X}_{k+1}} \max_{x_{k+1} \in \mathcal{X}_{k+1}} \chi_{k+1}(x_{k+1}, x_k, \hat{x}_{k+1}, \hat{x}_k) \omega_{k+1}(x_{k+1}, \hat{x}_{k+1}, o_{k+1}) \gamma_{k+1}(x_{k+1}, \hat{x}_{k+1})$$

and

$$\delta_k(x_k, \hat{x}_{1:k}) := \max_{x_{k-1} \in \mathcal{X}_{k-1}} \chi_k(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1}) \omega_k(x_k, \hat{x}_k, o_k) \delta_{k-1}(x_{k-1}, \hat{x}_{1:k-1}),$$

starting from $\gamma_n(x_n, \hat{x}_n) := 1$ and $\delta_1(x_1, \hat{x}_1) := \chi_1(x_1, \square, \hat{x}_1, \square) \omega_1(x_1, \hat{x}_1, o_1)$, then it is relatively easy to see that criterion (C2) reduces to the following simple inequality:

$$\max_{x_k \in \mathcal{X}_k} \delta_k(x_k, \hat{x}_{1:k}^*) \gamma_k(x_k, \hat{x}_k^*) > 1. \tag{C2'}$$

Whenever (C2') holds, we are guaranteed that (C1) holds as well and therefore, that there are no maximal state sequences that start with $\hat{x}_{1:k}^*$. As we will see, it is now but a small step to turn this into a working algorithm.

For $k = n$, criterion (C2') is even more powerful. In that case, the minima in expressions (C1'), (C1'') and (C2) disappear, thereby making these conditions equivalent. Hence, we find that for $k = n$, (C1) and (C2') are equivalent. Furthermore, criterion (C2') now reduces to

$$\max_{x_n \in \mathcal{X}_n} \delta_n(x_n, \hat{x}_{1:n}^*) > 1 \tag{C2*}$$

and (C1) and therefore also (C2*) serves as a necessary as well as sufficient condition for $\hat{x}_{1:n}^*$ *not* to be maximal: $\hat{x}_{1:n}^*$ is a maximal state sequence if and only if criterion (C2*) *fails*.

5.2 Turning it into an Algorithm

It is relatively easy to turn the ideas and formulas of the previous section into a working algorithm that is able to construct the set $\text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n})$ in an efficient manner. Algorithm 1 provides a pseudo-code version. As input data, it requires an output sequence $o_{1:n}$, the local parameters χ_k and ω_k and the global parameters γ_k . All these parameters can—and should—be calculated beforehand. Note that this is not the case for the parameters δ_k ; it is not feasible to calculate these beforehand, as there are simply too many.

Algorithm 1: MaxiHMM

Data: the local parameters χ_k and ω_k , an output sequence $o_{1:n}$, and the corresponding global parameters γ_k

Result: the set $\text{opt}(\mathcal{X}_{1:n}|o_{1:n})$ of all maximal state sequences

```

1  $\text{opt}(\mathcal{X}_{1:n}|o_{1:n}) \leftarrow \emptyset$ 
2 for  $\hat{x}_1 \in \mathcal{X}_1$  do
3   for  $x_1 \in \mathcal{X}_1$  do
4      $\delta_1(x_1, \hat{x}_1) \leftarrow \chi_1(x_1, \hat{x}_1)\omega_1(x_1, \hat{x}_1, o_1)$ 
5     if  $\max_{x_1 \in \mathcal{X}_1} \delta_1(x_1, \hat{x}_1)\gamma_1(x_1, \hat{x}_1) \leq 1$  then Recur(1,  $\hat{x}_1$ ,  $\delta_1(\cdot, \hat{x}_1)$ )
6 return  $\text{opt}(\mathcal{X}_{1:n}|o_{1:n})$ 

```

Procedure **Recur**($k, \hat{x}_{1:k}, \delta_k(\cdot, \hat{x}_{1:k})$)

```

1 if  $k = n$  then
2   add  $\hat{x}_{1:n}$  to  $\text{opt}(\mathcal{X}_{1:n}|o_{1:n})$  ▷ We found a solution!
3 else
4   for  $\hat{x}_{k+1} \in \mathcal{X}_{k+n}$  do
5      $\hat{x}_{1:k+1} \leftarrow (\hat{x}_{1:k}, \hat{x}_{k+1})$  ▷ Append  $\hat{x}_{k+1}$  to the end of  $\hat{x}_{1:k}$ 
6     for  $x_{k+1} \in \mathcal{X}_{k+1}$  do
7        $\delta_{k+1}(x_{k+1}, \hat{x}_{1:k+1}) \leftarrow \max_{x_k \in \mathcal{X}_k} \chi_{k+1}(x_{k+1}, x_k, \hat{x}_{k+1}, \hat{x}_k)$ 
8          $\omega_{k+1}(x_{k+1}, \hat{x}_{k+1}, o_{k+1})$ 
9          $\delta_k(x_k, \hat{x}_{1:k})$ 
10      if  $\max_{x_{k+1} \in \mathcal{X}_{k+1}} \delta_{k+1}(x_{k+1}, \hat{x}_{1:k+1})\gamma_{k+1}(x_{k+1}, \hat{x}_{k+1}) \leq 1$  then
11        Recur( $k + 1, \hat{x}_{1:k+1}, \delta_1(\cdot, \hat{x}_{1:k+1})$ )

```

The Procedure **Recur** implements the recursive nature of our algorithm. In it, we traverse the search tree that corresponds to $\mathcal{X}_{1:n}$ in depth-first order. That is, if the algorithm is unable to infer that there are no maximal sequences starting with $\hat{x}_{1:k}$ —if criterion (C2') fails—it presumes there are and immedi-

ately descends to depth $k + 1$ to check criterion (C2') again. In order to be able to perform this check for $k + 1$, we need the parameters $\delta_{k+1}(x_{k+1}, \hat{x}_{1:k+1})$, which—as said before—have not been calculated beforehand. However, luckily, these parameters can easily be calculated while running the algorithm, based on the parameters $\delta_k(x_k, \hat{x}_{1:k})$ that were used in the previous step; see Lines 7–9 of the Procedure **Recur**.

When we arrive at depth n , we check criterion (C2')—which is now equivalent to (C2*)—and, if it fails, we add the current sequence $\hat{x}_{1:n}$ to the solution set. After all, if criterion (C2*) fails, we are guaranteed to have found a maximal solution. Since, while running the algorithm, we have only “ignored” sequences that were definitely not maximal—because criterion (C2') was true—this means that the MaxiHMM algorithm does indeed succeed in constructing the set $\text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n})$ correctly.

Example 2. In Figure 2, the MaxiHMM algorithm starts by checking criterion (C2') for $\hat{x}_1 = 0$. The criterion fails, and therefore, the algorithm descends to depth 2, now checking criterion (C2') for $\hat{x}_{1:2} = 00$. This time, the criterion is true, allowing us to “ignore” all the sequences that start with ‘00’. Next, the algorithm checks criterion (C2') for $\hat{x}_{1:2} = 01$, which turns out to be false. By proceeding in this way, we eventually find that in this case, $\text{opt}_{\max}(\mathcal{X}_{1:4}|o_{1:4})$ consists of three maximal sequences: ‘0100’, ‘0111’ and ‘1000’. ♦

5.3 Complexity Analysis

The time complexity of the MaxiHMM algorithm depends on a number of factors. First of all, we have to take into account the size S of the set $\text{opt}_{\max}(\mathcal{X}_{1:n}|o_{1:n})$ we are looking for. After all, if all state sequences in $\mathcal{X}_{1:n}$ are maximal, then no single branch can be pruned from the search tree. In that case, the complete tree has to be traversed, which clearly has a time complexity that is exponential in n . Note that this is far from surprising; in this case, even simply printing all the maximal sequences has such a complexity.

In general, our algorithm is linear in the number of times criterion (C2') fails or, equivalently, the number of times we execute Line 5 of the MaxiHMM algorithm or Line 11 of the Procedure **Recur**. For ease of reference, let us denote this number by C . For example, in Figure 2, C is the number of ✓-signs. By taking a closer look at the pseudo-code of our algorithm, we find that it has a time complexity of the order $O(Cm^2)$.

Let us now assume that criterion (C2') is equivalent to (C1). Then every time criterion (C2')—and hence (C1)—fails, the current node in the search tree is guaranteed to be part of a maximal state sequence. Since there are S maximal state sequences, each of which consists of n nodes, we find that C is bounded above by Sn . Hence, under the assumption that (C2') is equivalent to (C1), the time complexity of our algorithm is $O(Snm^2)$. Interestingly, this is linear in the number of maximal sequences S . It is also comparable to the complexity of the Viterbi algorithm, since in that particular case, $S = 1$.

Of course, as explained in Section 5.1, the criteria (C1) and (C2') are not equivalent and therefore, from a theoretical point of view, the aforementioned complexity cannot be guaranteed. For example, it might occur that $C > Sn$; in Figure 2, we see that $13 = C > Sn = 12$. Nevertheless, it turns out that in practice—we illustrate this in Section 7.1—the time complexity of our algorithm tends to increase linearly in S . This suggests that—despite the fact that criteria (C2') and (C1) are not guaranteed to be identical—the aforementioned time complexity of $O(Snm^2)$ might be a good approximation of reality.

6 Common Local Models and Their Parameters

In order to apply the above algorithm, all that is needed are the local parameters ω_k and χ_k . For general compact local models, these can be obtained by applying the formulas in Section 4.1. However, for some specific classes of local models, closed-form expressions for these parameters are available as well. In the following, we discuss two such instances: local models that are obtained by means of ϵ -contamination and local models that are derived from data using Walley's Imprecise Dirichlet Model (IDM) [7].

6.1 Frequently Used Imprecise-Probabilistic Models

The perhaps simplest way to obtain an imprecise local model is to ϵ -contaminate a mass function p in $\Sigma_{\mathcal{X}}$. For any ϵ in $[0, 1]$, the corresponding ϵ -contaminated model is defined as

$$\mathcal{M}_p^\epsilon := \{(1 - \epsilon)p + \epsilon q : q \in \Sigma_{\mathcal{X}}\}.$$

It is a closed, bounded and therefore also compact subset of $\Sigma_{\mathcal{X}}$. For $\epsilon = 0$, we find that $\mathcal{M}_p^0 = \{p\}$, thereby recovering the precise-probabilistic case. As ϵ increases, additional mass functions are added. For $\epsilon = 1$, \mathcal{M}_p^1 is equal to $\Sigma_{\mathcal{X}}$, thereby representing complete model uncertainty. In order to satisfy our positivity assumption, we require that p is an element of $\text{int}\Sigma_{\mathcal{X}}$ and that $\epsilon < 1$.

The corresponding lower and upper probabilities are easily calculated. For example, if $|\mathcal{X}| \geq 2$, then for any singleton $x \in \mathcal{X}$, we find that

$$\bar{p}_\epsilon(x) := \max \{\tilde{p}(x) : \tilde{p} \in \mathcal{M}_p^\epsilon\} = (1 - \epsilon)p(x) + \epsilon, \quad (9)$$

$$\underline{p}_\epsilon(x) := \min \{\tilde{p}(x) : \tilde{p} \in \mathcal{M}_p^\epsilon\} = (1 - \epsilon)p(x). \quad (10)$$

Example 3. Consider a ternary sample space $\mathcal{X} = \{a, b, c\}$. Then any probability mass function p on \mathcal{X} can be identified with a point in an equilateral triangle with height one, which represents the simplex $\Sigma_{\mathcal{X}}$; see Figure 3. The ϵ -contaminated model \mathcal{M}_p^ϵ is represented by an equilateral triangle with height ϵ , which 'grows' around p as ϵ increases. \blacklozenge

The following lemma establishes a technical property of ϵ -contaminated models that will enable us to obtain closed-form expressions for the local parameters ω_k and χ_k .

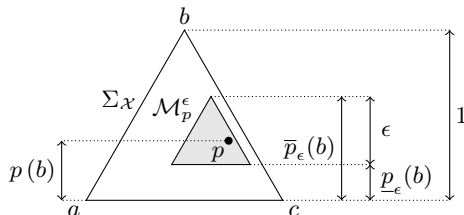


Fig. 3. Constructing an ϵ -contaminated model.

Proposition 2. Consider any $p \in \text{int}(\Sigma_{\mathcal{X}})$, with $|\mathcal{X}| \geq 2$, and any $\epsilon \in [0, 1)$. Then $\mathcal{M}_p^\epsilon \subseteq \text{int} \Sigma_{\mathcal{X}}$ and, for all $x, \hat{x} \in \mathcal{X}$ such that $x \neq \hat{x}$:

$$\min_{p' \in \mathcal{M}_p^\epsilon} \frac{p'(x)}{p'(\hat{x})} = \frac{\underline{p}_\epsilon(x)}{\overline{p}_\epsilon(\hat{x})}.$$

Besides ϵ -contamination, another popular method for constructing imprecise local models is to derive them from data by means of Walley's IDM; see Ref. [7] for a thorough discussion. For our presents purposes, it suffices to mention that the resulting predictive model on \mathcal{X} coincides with an ϵ -contaminated model \mathcal{M}_p^ϵ , with p and ϵ constructed as follows. For a data set of n experiments, n_x of which are equal to x , we let $p(x) := n_x/n$. Furthermore, $\epsilon := s/n+s$, where $s > 0$ is a parameter of the IDM that can be interpreted as a degree of cautiousness. In order for our positivity assumption to be satisfied, we require that, for all $x \in \mathcal{X}$, $n_x > 0$. Using this connection between the IDM and ϵ -contamination, we find that, with $|\mathcal{X}| \geq 2$, for any $x \in \mathcal{X}$:

$$\underline{p}_{\text{IDM}}(x) := \frac{n_x}{n+s} \text{ and } \overline{p}_{\text{IDM}}(x) := \frac{n_x+s}{n+s}.$$

6.2 Local Parameters for an ϵ -contaminated Model

An important advantage of working with ϵ -contaminated local models (and therefore also the IDM) is that the corresponding local parameters ω_k^ϵ and χ_k^ϵ are extremely easy to calculate.

For $\omega_k^\epsilon(x_k, \hat{x}_k, o_k)$, it suffices to plug the local lower and upper probabilities, as obtained by Eqs. (9) and (10), into the expression that is provided by Proposition 1. We can proceed in much the same way to calculate $\chi_k^\epsilon(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1})$, except if $x_k \neq \hat{x}_k$ and $x_{k-1} = \hat{x}_{k-1}$. However, luckily, in this case, Proposition 2 is applicable, which allows us to optimise the numerator and denominator in Eq. (5) separately anyway, as in the case $x_{k-1} \neq \hat{x}_{k-1}$. Hence, we find that

$$\chi_k^\epsilon(x_k, x_{k-1}, \hat{x}_k, \hat{x}_{k-1}) = \begin{cases} 1 & \text{if } x_k = \hat{x}_k \text{ and } x_{k-1} = \hat{x}_{k-1}, \\ \frac{\underline{p}_k^\epsilon(x_k|x_{k-1})}{\overline{p}_k^\epsilon(x_k|\hat{x}_{k-1})} & \text{if } x_k \neq \hat{x}_k \text{ or } x_{k-1} \neq \hat{x}_{k-1}. \end{cases}$$

7 Experiments

We conclude this paper with a number of experiments. In order to allow us to visualise them easily, we focus on binary (i)HMMs. Hence, for all k in $\{1, \dots, n\}$: $\mathcal{X}_k = \mathcal{O}_k := \{0, 1\}$. We start from a precise stationary HMM. By stationarity, and since binary mass functions can be specified by means of a single number, the local models of this HMM are completely characterised by five numbers: $q(0|0) = 0.9$, $q(0|1) = 0.1$, $p_1(0) = 0.5$, $l := p(0|0)$ and $m := p(0|1)$. We turn this precise HMM into an imprecise one by ϵ -contaminating all of its local models with the same ϵ . In this way, we obtain a stationary iHMM, meaning that the *imprecise* local models do not depend on k . Note however that, by Eq. (2), the corresponding family of *precise* HMMs contains stationary as well as non-stationary ones.

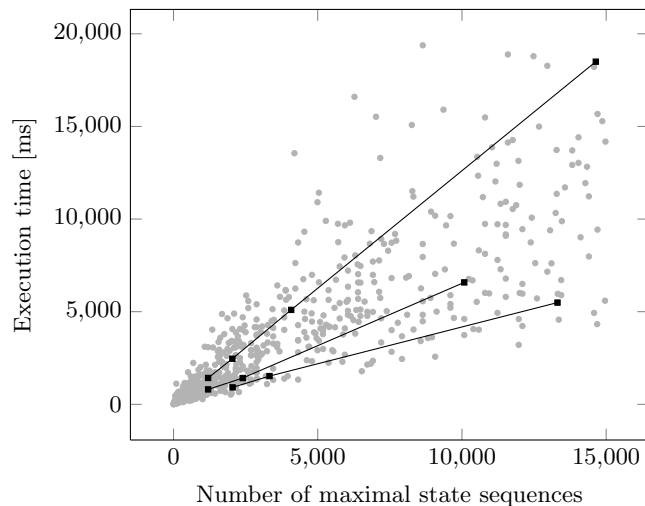


Fig. 4. Scatterplot of 760 + 10 complexity experiments.

7.1 Computational Complexity Experiments

We begin by corroborating the statement that was made in Section 5.3: that in practice, the time complexity of our algorithm tends to increase linearly in the number of maximal state sequences. Figure 4 illustrates the correlation between the execution time of the algorithm and the number of maximal sequences it produces. For these experiments, we chose $l = 0.9$ and $m = 0.1$. The grey dots correspond to 760 randomly generated output sequences of length $n = 100$, with values of ϵ ranging from 0.01 to 0.1. We clearly recognise some kind of cone, which already suggests that the execution time increases linearly in the number

of maximal sequences. In black, we plot the results for three additional random but fixed sequences, for different values of ϵ ; experiments that correspond to the same output sequence have been connected. This time, the observed linearity is rather striking.

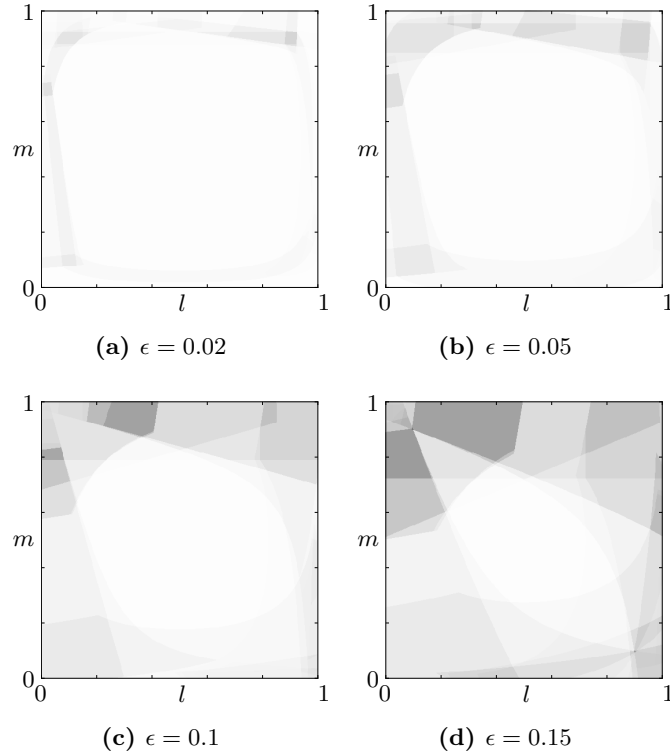


Fig. 5. Number of maximal state sequences, for different local models.

7.2 A Closer Look at the Number of Maximal State Sequences

Since the complexity of our algorithm depends so crucially on the number of maximal sequences it returns, we will now take a closer look at this number and investigate the extent to which it depends on the local models of our iHMM. We do this by visualising the number of maximal sequences as a function of the transition probabilities l and m in four heat plots, each of which corresponds to a different value for ϵ . The output sequence is ‘1100110011’, with $n = 10$. The results are depicted in Figure 5. White corresponds to a single maximal sequences, whereas pitch black corresponds to 200 sequences being maximal.

As expected, the number of maximal state sequences increases with ϵ . That is, the regions that correspond to a higher number of maximal sequences become

wider as ϵ increases. The maximum number of maximal sequences that can be observed in these plots is about 160—for $\epsilon = 0.15$, the tiny black dot near the upper left corner of the heat plot ($l = 0.1$ and $m = 0.9$). Note that this is only 16% of the maximum number possible, which is $2^{10} = 1024$. The large, (dark) gray regions correspond to 77 maximal sequences. Finally, and this is rather remarkable, we observe that there are fairly large regions in which—even for $\epsilon = 0.15$ —there is only one maximal state sequence.

8 Conclusions and Future Work

The main contribution of this paper is an algorithm that can construct the set of all maximal state sequences of an iHMM in an efficient manner, thereby providing a robust version of the Viterbi algorithm. Our experiments show that the time complexity of this algorithm tends to increase linearly in the number of maximal state sequences. Finally, we have illustrated how this number depends upon the parameters of the iHMM.

We see a number of interesting avenues for future research, the most important of which is perhaps to apply our algorithm to a real-life problem, and to compare the results with those of the Viterbi algorithm. Another, more theoretically oriented line of research is to develop the algorithm without the positivity assumption. Finally, we would like to see whether the ideas in this paper can be used to develop similarly efficient algorithms for credal networks whose graphical structure is more complicated than that of an HMM.

Acknowledgements Jasper De Bock is a PhD Fellow of the Research Foundation Flanders (FWO) and wishes to acknowledge its financial support. The authors would also like to thank Alessandro Antonucci and Cassio P. de Campos for a number of stimulating discussions on the topic of this paper.

References

1. Cozman, F.G.: Credal networks. *Artificial Intelligence* 120, 199–233 (2000)
2. De Bock, J., De Cooman, G.: State sequence prediction in imprecise hidden Markov models. In: 7th ISIPTA (2011)
3. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. In: *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286 (1989)
4. Troffaes, M.C.M.: Decision making under uncertainty using imprecise probabilities. *International Journal of Approximate Reasoning* 45, 17–29 (2007)
5. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In: *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269 (1967)
6. Walley, P.: *Statistical Reasoning with Imprecise Probabilities*. Chapman & Hall (1991)
7. Walley, P.: Inferences from Multinomial Data: Learning about a Bag of Marbles. *Journal of the Royal Statistical Society Series B* 58, no. 1, 3–57 (1996)