RAIRO-Oper. Res. 49 (2015) 511–526
DOI: 10.1051/ro/2014053

**RAIRO Operations Research**
www.rairo-ro.org

# VITERBI ALGORITHMS FOR HIDDEN SEMI-MARKOV MODELS WITH APPLICATION TO DNA ANALYSIS

Christina-Elisavet Pertsinidou[1,2] and Nikolaos Limnios[1]

**Abstract.** In this paper we present a new Viterbi algorithm for Hidden semi-Markov models and also a second algorithm which is a generalization of the first. These algorithms can be used to decode an unobserved hidden semi-Markov process and it is the first time that the complexity is achieved to be the same as in the Viterbi for Hidden Markov models, *i.e.* a linear function of the number of observations and quadratic function of the number of hidden states. An example in DNA Analysis is also given.

**Keywords.** Viterbi algorithm, Hidden semi-Markov model, hidden Markov model, DNA Analysis.

**Mathematics Subject Classification.** 68Q25, 68Q15, 60K15, 65K05.

## 1. Introduction

Hidden Markov models (HMMs) are important tools in estimation and analysis of biological sequences and many other systems. Though, the main disadvantage is that the sojourn times of the hidden process are geometrically distributed, which is inappropriate for some applications. A hidden semi-Markov model (*e.g.*, [1]), (HSMM), which is a generalization of a HMM, is a solution to the problem because it allows arbitrary sojourn time distributions for the hidden process. In the sequel, we will present a Viterbi algorithm for hidden semi-Markov models and a

Received March 3rd, 2013. Accepted October 8, 2014.

[1] Université de Technologie de Compiègne, Laboratoire de Mathématiques Appliquées, Centre de Recherches de Royallieu, CS 60319, 60203 Compiègne Cedex, France. p.eli.christina@gmail.com

[2] Aristotle University of Thessaloniki, School of Mathematics, 54124 Thessaloniki, Greece

generalization of this new algorithm. To our knowledge this kind of algorithms for HSMMs are proposed for the first time.

In the literature, (*e.g.* [4–6, 9, 11]), there exist algorithms named Viterbi for HSMMs algorithms, but most of them, do not really relay on the definitions of HSMMs ([1]) and some of them even if they do, they have a double (or even more) maximization, which makes them very complicated and difficult to be used. We recall that the Viterbi algorithm for HMMs (see appendix) has a single maximization, for the hidden states. For example in [9], there is a double maximization, one for the sojourn times and one for the hidden states, which leads to a higher complexity. More precisely, if d represents the number of hidden states, T the number of observations, $D_{\max}$ the maximum time duration in any given state and k the average number of states that can transition to a given state, then the complexity of the algorithm in [9] is $O(dTD_{\max}k)$. In [5] there is again a second maximization as for the state duration and the complexity is $O(d^2TD)$ where $D$ represents the sum of the maximum duration in each hidden state. In [4] there is a different definition concerning the emmision probabilities, than in [1] and also again a double maximization. The complexity in this case is $O(d^2TL)$, where L is the sum of durations of d states. In addition, in [6] there exists an algorithm with two maximizations, which leads to a complexity that can become $O(dT(d+T))$, in the worst case. Finally, in [11] though the algorithm is referred as Viterbi for HSMMs, the definitions of the transition probabilities and the emission probabilities differ a lot from those in [1]. Furthermore, in [11], there are presented three specific cases of HSMMs with complexities of the form $O((M^2+MD^2)T)$, $O((M^2D)T)$ and $O((MD+M^2)T)$. We overcome these problems by working with backward-recurrence times of the associated backward recurrence Markov chain $(Z,U)$, (defined in the sequel), instead of working with sojourn times. This allow us to create an innovative algorithm for HSMMs, for the first time, with only one maximization and the complexity is the same as in the Viterbi HMM algorithm, *i.e.* $O(Td^2)$. There is a great importance of having such an algorithm because of the wide range of fields that it can be applied, such as DNA analysis and speach recognition.

The paper is organised as follows:

Section 2 presents some basic notions of the semi-Markov chains, the Markov renewal chains and the hidden semi-Markov framework.

Section 3 presents the new Viterbi algorithm, analyzes its structure and provides an application in a Poisson distributed HSMM.

Section 4 presents a generalisation of the new Viterbi algorithm and an application in DNA Analysis.

Section 5 contains the conclusions.

Section 6 is the appendix, which presents the classical Viterbi algorithm for HMMs, a pseudocode for the new algorithm, a pseudocode for the generalisation of the new algorithm and further details about the complexity.

## 2. Hidden semi-Markov framework

Before presenting the Hidden semi-Markov framework, we will present at first some basic notions concerning the semi-Markov chains and Markov renewal chains (see [1]) required for the understanding of HSMMs.

Consider a random system with finite state space $E = \{1, \ldots, d\}$. We suppose that the evolution in time of the system is described by the following chains:

The chain $J = (J_n)_{n\in\mathbb{N}}$ with state space $E$, where $J_n$ is the system state at the $n$th jump time.

The chain $S = (S_n)_{n\in\mathbb{N}}$ with state space $\mathbb{N}$, where $S_n$ is the $n$th jump time. We suppose that $S_0 = 0$ and $0 < S_1 < S_2 < \ldots < S_n < S_{n+1} < \ldots$

The chain $X = (X_n)_{n\in\mathbb{N}}$ with state space $\mathbb{N}$, $X_n := S_n - S_{n-1}$ for all $n \in \mathbb{N}^*$ and $X_0 := 0$. Thus for all $n \in \mathbb{N}^*$, $X_n$ is the sojourn time in state $J_{n-1}$, before the $nth$ jump. One fundamental notion for our work is that of the semi-Markov kernel in discrete time.

**Definition 2.1.** (discrete-time semi-Markov kernel) A matrix-valued function $q = q_{ij}(k)$ is said to be a discrete semi-Markov kernel if it satisfies the following three properties:

(1) $0 \leq q_{ij}(k), i, j \in \mathbb{N}$,
(2) $q_{ij}(0) = 0, i, j \in E$,
(3) $\sum_{k=0}^{\infty} \sum_{j\in E} q_{ij}(k) = 1, i \in E$.

**Definition 2.2.** (Markov renewal chain) The chain $(J, S) = (J_n, S_n)_{n\in\mathbb{N}}$ is said to be a Markov renewal chain (MRC) if for all $n \in \mathbb{N}$, for all $i, j \in E$, and for all $k \in \mathbb{N}$ it satisfies almost surely

$$\begin{aligned}\mathbb{P}(J_{n+1} = j, S_{n+1} - S_n = k \mid J_0, \ldots, J_n; S_0, \ldots S_n)\\ = \mathbb{P}(J_{n+1} = j, S_{n+1} - S_n = k \mid J_n).\end{aligned}$$

Moreover, if the above equation is independent of n, then (J,S) is said to be homogeneous and the discrete-time semi-Markov kernel q is defined by

$$q_{ij}(k) := \mathbb{P}(J_{n+1} = j, X_{n+1} = k \mid J_n = i).$$

We also denote, by $p = (p_{ij})_{i,j\in E}$ the transition matrix of $(J_n)$, defined by

$$p_{ij} := \mathbb{P}(J_{n+1} = j \mid J_n = i), i, j \in E, n \in \mathbb{N}.$$

**Definition 2.3.** (conditional distribution of sojourn times) $f_{ij}(\cdot)$, the conditional distribution of $X_{n+1}, n \in \mathbb{N}$:

$$f_{ij}(k) := \mathbb{P}(X_{n+1} = k \mid J_n = i, J_{n+1} = j), k \in \mathbb{N}.$$

We would like to mention that the semi-Markov kernel verifies the relation $q_{ij} = f_{ij}p_{ij}$, for all $i, j \in E$ and $k \in \mathbb{N}$ such that $p_{ij} \neq 0$.

**Definition 2.4.** (sojourn time distributions in a given state) For all $i \in E$, let us denote by:

(1) $h_i(\cdot)$ the sojourn time distribution in state $i$:

$$h_i(k) := \mathbb{P}(X_{n+1} = k \mid J_n = i) = \sum_{j \in E} q_{ij}(k), k \in \mathbb{N}.$$

(2) $H_i(\cdot)$ the sojourn time cumulative distribution in state $i$:

$$H_i(k) := \mathbb{P}(X_{n+1} \leq k \mid J_n = i) = \sum_{l=1}^{k} h_i(l), k \in \mathbb{N}.$$

We will also denote by $\bar{H}_i := 1 - H_i(k) = \mathbb{P}(X > k)$ the survival function of $H_i(k)$.

**Definition 2.5.** (semi-Markov chain)
Let $(J, S)$ be a Markov renewal chain. The chain $Z = (Z_k)_{k \in \mathbb{N}}$ is said to be a semi-Markov chain associated to the MRC $(J, S)$ if

$$Z_k := J_{N(k)}, k \in \mathbb{N},$$

where

$$N(k) := \max\{n \in \mathbb{N} \mid S_n \leq k\}$$

is the discrete-time counting process of the number of jumps in $[1, k]$.

We want to point out that it is only a matter of technical convenience that we have chosen to define $N(k)$ as the counting process of the number of jumps in $[1, k]$ instead of $[0, k]$.

Let the row vector $\alpha = (\alpha_1, \ldots \alpha_s)$ denote the initial distribution of the semi-Markov chain $Z = (Z_k)_{k \in \mathbb{N}}$, *i.e.*, $\alpha_i := \mathbb{P}(Z_0 = i) = \mathbb{P}(J_0 = i), i \in E$.

In HSMMs now, the unobserved process is a semi-Markov chain and the observed process is independent conditionally on the values of the semi-Markov chain. Let $E = \{e_1, \ldots, e_d\}$ and $A = \{a_1, \ldots, a_s\}$ be two finite sets. Consider also the double chain $(Z_k, Y_k)_{k \geq 0}$, where $(Z_k)$ is an E-valued semi-Markov chain, with semi-Markov kernel $q(k) = (q_{ij}(k); i, j \in E)$, $k \in \mathbb{N}$, and initial probability $\tilde{\alpha}$. The process $(Y_k)$ is the observed process, with distribution depending on the values of $(Z_k)$, *i.e.*,

$$\mathbb{P}(Y_k = y \mid Z_k = z) =: R(z, y), \quad (z, y) \in E \times A. \tag{2.1}$$

When the emission probability matrix is of the form (2.1) the hidden semi-Markov model is denoted by SM1-M0. When relation (2.1) is replaced by the following one

$$\mathbb{P}(Y_k = y \mid Z_k = z, Y_{k-1} = x) =: R(z, x, y)$$

then the model is denoted by SM1-M1. The process $(Z_k, Y_k)_{k \geq 0}$ is called a HSMM.

Starting from the initial hidden semi-Markov chain $(Z_k, Y_k)_{k\in\mathbb{N}}$, we have an associated hidden Markov chain $((Z,U),Y) = ((Z_k,U_k),Y_k)_{k\in\mathbb{N}}$ with $(Z_k,U_k)_{k\in\mathbb{N}}$ a Markov chain and $(Y_k)_{k\in\mathbb{N}}$ a sequence of conditionally independent random variables. The backward recurrence times $(U_k)_{k\in\mathbb{N}}$ of the semi-Markov chain $(Z_k)_{k\in\mathbb{N}}$ are given by the equation:

$$U_k = k - S_{N(k)},$$

where $N(k)$ is the number of transitions until time $k$. The associated backward recurrence Markov chain $(Z,U)$ transition probabilities are (see [3]) :

$$\tilde{p}_{(i,t_1)(j,t_2)} = \begin{cases} q_{ij}(t_1+1)/\bar{H}_i(t_1) & \text{if } i \neq j,\ t_2 = 0 \\ \bar{H}_i(t_1+1)/\bar{H}_i(t_1) & \text{if } i = j,\ t_2 - t_1 = 1\,, \\ 0 & \text{elsewhere} \end{cases}$$

where $\bar{H}_i(\cdot)$ is the survival function of sojourn time in state $i$,

$$\bar{H}_i(n) = 1 - \sum_{j\in E}\sum_{k=1}^{n} q_{ij}(k), n \in \mathbb{N}^*.$$

The aim is to observe a realisation of $(Y_k, 0 \le k \le n)$, *i.e.*, $y_0^n = (y_0, \dots y_n) \in A^{n+1}$ and try to find out the corresponding hidden regime, *i.e.*, $z_0^n = (z_0, \dots z_n) \in E^{n+1}$, a realisation of $(Z_k, 0 \le k \le n)$. The parameters of the discrete hidden Markov model are $(P_{ij}, R, \tilde{\alpha})$, where $\tilde{\alpha}$ is the initial probability of the Markov chain. In HMMs in order to find the best hidden sequence we need to compute the $\arg\max_z P(z \mid y)$ which is equivalent to computing $\arg\max_z P(z,y)$. The Viterbi algorithm, (*e.g.*, [8]) for HMMs maximizes for all k the joint probability

$$P(z_1, \dots z_k, y_1, \dots y_k) = \mathbb{P}(Z_1^k = z_1^k, Y_1^k = y_1^k)$$

and computes in every step the $\arg\max_{z_1^k}(P(z_1, \dots z_k, y_1, \dots y_k))$. By backtracking we obtain the optimal hidden state sequence. The algorithm can be written in a logarithmic mode (base 2) which is recommended in order to avoid an underflow error. In the first step we have the initial conditions, in the second step the recurrence formula, in the third step we find the last hidden state and the probability of the optimal path and in the fourth step by backtracking, we find the hidden state sequence.

## 3. A Viterbi algorithm for a SM1-M0 model

### 3.1. Joint probability formulas

The proposed Viterbi algorithms for SM1-M0, (Sect. 3.2), and SM1-M1 (Sect. 3.3), are both forward-backward. The Viterbi algorithm for HMMs that already exists in the literature can only be applied when the sojourn times are geometrically distributed. This is the basic drawback of HMMs compared to HSMMs, where the distribution of sojourn times can be arbitrary. We overcome this problem by proposing a new algorithm for a SM1-M0 model. The key feature of this new algorithm is the use of the associated backward recurrence Markov chain, which by definition allows the sojourn times to be arbitrarily distributed and at the same time maintains the property of the HMM to be able to move step by step, which is very important for the structure and the complexity of the new algorithm. One of our goals is to remain as close as possible to the structure of the classical Viterbi algorithm for HMMs in order to achieve the minimum complexity that we could, but at the same time generalize this to the HSMM case. Thus, the proposed new algorithm is a specification of the classical one to the HSMM context.

In the first step we have the initial conditions, in the second step the recurrence formula, in the third step we find the last hidden state and the optimal path's probability and in the fourth step by backtracking we derive the hidden state sequence. The following Proposition 3.1, help us to construct the algorithm.

**Proposition 3.1.** *The joint probability $u_k = \mathbb{P}\left(Z_0^k = i_0^k\right)$ is given by the following recurrence formula*

$$u_k = a_k\left(i_0^k\right) u_{k-1}, \quad k \geq 1, u_0 = \tilde{\alpha}(i_0), i_{-1} = i_{0-}$$

*where,*

$$a_k\left(i_0^k\right) = \sum_{l_1=0}^{k-1} \sum_{l_2 \in \{0, l_1+1\}} \tilde{p}_{(i_{k-1},l_1)(i_k,l_2)} \times \mathbf{1}_{\{i_{k-1}=\ldots=i_{k-l_1-1}\neq i_{k-l_1-2}\}}, \quad k \geq 1$$

*and*

$$\mathbf{1}_A(x) = \mathbf{1}_{\{x\in A\}} := \begin{cases} 1 & \textit{if } x \in A \\ 0 & \textit{elsewhere} \end{cases}$$

*Proof.* We have

$$\begin{aligned}
u_k &= \mathbb{P}\left(Z_0^k = i_0^k\right) \\
&= \sum_{l_1=0}^{k-1} \sum_{l_2 \in \{0, l_1+1\}} \mathbb{P}(Z_0^{k-2} = i_0^{k-2}, Z_{k-1} = i_{k-1}, U_{k-1} = l_1, Z_k = i_k, U_k = l_2) \\
&= \sum_{l_1=0}^{k-1} \sum_{l_2 \in \{0, l_1+1\}} \mathbb{P}\left(Z_0^{k-1} = i_0^{k-1}\right) \mathbb{P}\left(U_{k-1} = l_1 \mid Z_0^{k-1} = i_0^{k-1}\right) \\
&\quad \times \mathbb{P}\left(Z_k = i_k, U_k = l_2 \mid Z_0^{k-2} = i_0^{k-2}, Z_{k-1} = i_{k-1}, U_{k-1} = l_1\right) \\
&= \sum_{l_1=0}^{k-1} \sum_{l_2 \in \{0, l_1+1\}} u_{k-1} \tilde{p}_{(i_{k-1},l_1)(i_k,l_2)} \mathbf{1}_{\{i_{k-1}=\ldots=i_{k-l_1-1}\neq i_{k-l_1-2}\}} \\
&= u_{k-1} \sum_{l_1=0}^{k-1} \sum_{l_2 \in \{0, l_1+1\}} \tilde{p}_{(i_{k-1},l_1)(i_k,l_2)} \mathbf{1}_{\{i_{k-1}=\ldots=i_{k-l_1-1}\neq i_{k-l_1-2}\}},
\end{aligned}$$

which proves the claimed result. □

**Proposition 3.2.** *The joint probability $u_k = \mathbb{P}(Z_0^k = i_0^k)$ is also given by the following equation:*

$$\begin{aligned}
u_k &= \mathbb{P}\left(Z_0^k = i_0^k\right) \\
&= \bar{H}_{i_0}(k) \times \mathbf{1}_{\{i_{l_0}=i_0, 0\le l_0 \le k\}} \\
&\quad + \sum_{n=1}^{k} \sum_{0<s_1<\ldots<s_n\le k} \sum_{J_1,\ldots,J_n\in E} q_{i_0 j_1}(s_1) q_{j_1 j_2}(s_2-s_1)\ldots q_{j_{n-1}j_n}(s_n - s_{n-1}) \\
&\quad \times \bar{H}_{j_n}(k-s_n) \times \mathbf{1}_{\{i_{l_0}=j_0, 0\le l_0<s_1, i_{l_1}=j_1, s_1\le l_1<s_2,\ldots,i_{l_n}=j_n, s_n\le l_n\le k\}}.
\end{aligned}$$

## 3.2. A Viterbi algorithm for a SM1-M0 model

The proposed new algorithm is the following one

**Algorithm 3.3.** ***Step 1.*** *Initial conditions. For $k = 0$,*

$$d_0(i_0) = \log_2(\tilde{\alpha}(i_0)) + \log_2(R(i_0, y_0)), \quad b_0(i_0) = 0.$$

***Step 2.*** *For $k \ge 1$*

$$\begin{aligned}
d_k(i_k) &= \max_{i_{k-1}\in E}[d_{k-1}(i_{k-1}) + \log_2(a_k(i_0^k))] + \log_2(R(i_k, y_k)) \\
b_k(i_k) &= \arg\max_{i_{k-1}\in E}[d_{k-1}(i_{k-1}) + \log_2(a_k(i_0^k))]
\end{aligned}$$

*where*

$$a_k(i_0^k) = \sum_{l_1=0}^{k-1} \sum_{l_2\in\{0,l_1+1\}} \tilde{p}_{\{(i_{k-1},l_1)(i_k,l_2)\}} \times \mathbf{1}_{\{i_{k-1}=\ldots=i_{k-l_1-1}\neq i_{k-l_1-2}\}}, \quad k \ge 1.$$

***Step 3.*** *Termination If $k = T-1$ ($T$ is the length of the observation sequence),*

$$\begin{aligned}
P &= \max_{i_{T-1}\in E}[d_{T-1}(i_{T-1})], \quad P^* = 2^{max_{i_{T-1}\in E}[d_{T-1}(i_{T-1})]}, \\
q_{T-1} &= \arg\max_{i_{T-1}\in E}[d_{T-1}(i_{T-1})].
\end{aligned}$$

***Step 4.*** *Sequence of states with backward-forward steps*

$$q_{k-1} = b_k(q_k), \quad k = T-1,\ldots,1.$$

More precisely, in step 3, $P$ maximizes the optimal path's probability (in logarithm base 2), $P^*$ represents the optimal path's probability and $q_{T-1}$ is the last decoded hidden state. All the previous hidden states are revealed by the backward-forward technique in step 4.

The complexity of the Viterbi algorithm in HMM is $O(T \times d^2)$, (see [10]). The complexity of the proposed algorithms is also $O(T \times d^2)$, where $T$ is the observation sequence length and $d$ the number of hidden states. This is because, we have only one maximization and this maximization (in every iteration), allow us to know exactly which non-zero $\tilde{p}_{\{(i_{k-1},l_1)(i_k,l_2)\}}$ probability we need to use from the quantities

$$a_k\left(i_0^k\right) = \sum_{l_1=0}^{k-1} \sum_{l_2\in\{0,l_1+1\}} \tilde{p}_{\{(i_{k-1},l_1)(i_k,l_2)\}} \times \mathbf{1}_{\{i_{k-1}=\ldots=i_{k-l_1-1}\neq i_{k-l_1-2}\}}, \quad k \geq 1,$$

without needing to sum over all possible recurrence times.

The complexity will be better understood following the analysis of the algorithm structure, provided in the sequel. In Appendix A.2. a pseudocode of the algorithm and further information about the complexity are presented.

**Remark 3.4.** The algorithm was tested for the Markovian case, where the distribution of the sojourn times is the geometric, the results were compared to those of the classical Viterbi for HMMs and were found to be the same. A Markov chain with transition matrix $p_{ij}$ is a particular case of a Markov renewal chain. The formula for computing the Markov chain's semi-Markov kernel can be found in [1].

**Remark 3.5.** The algorithm was constructed for the case that the transition probability matrix of the imbedded Markov chain is of the standard form. This is the general case where all the diagonal elements are zero. Any transition probability matrix can be transformed to this form, by setting the diagonal elements equal to zero and replacing the other elements using the formula $p'_{ij} = p_{ij}/(1-p_{ii})$. The standard form is unique for every transition probability matrix. This should not be confused with the transition probabilities of the associated backward recurrence Markov chain, where transitions to the same state are allowed.

**Remark 3.6.** The discrete HSMM framework requires that for time $k = 0$ the conditional distribution of the sojourn times is zero, $f_{ij}(0) = 0$, and therefore the kernel probabilities at $k = 0$ are also zero, $q_{ij}(0) = 0$. This means that the probability law of the sojourn times should be defined in $\mathbb{N}^*$. In case though the probability law is defined in $\mathbb{N}$ and thus $f_{ij}(0) > 0$, and $q_{ij}(0) > 0$, before using the algorithm one should proceed to the following transformation

$$f'_{ij}(0) = 0, f'_{ij}(k) = f_{ij}(k)/(1 - f_{ij}(0)), \quad k \geq 1. \tag{3.1}$$

In order to see the correctness of the algorithm let us analyze, as an example, the first three steps. By the definition of the problem we need to compute in each step the joint probability $P({Z_0}^k, {Y_0}^k)$.

For $k = 0$ (in multiplication form)

$$d_0(i_0) = \tilde{\alpha}(i_0)R(i_0, y_0).$$

This is the trivial case, exactly the same as in the Viterbi algorithm for HMM.

For $k = 1$

$$\begin{aligned}\mathbb{P}\left(Z_0^1 = i_0^1, Y_0^1 = y_0^1\right) &= R(i_0, y_0)R(i_1, y_1)u_1\\ &= R(i_0, y_0)R(i_1, y_1)a_1({i_0}^1)u_0\\ &= R(i_0, y_0)R(i_1, y_1)a_1({i_0}^1)\tilde{\alpha}(i_0)\\ d_1(i_1) &= \max[d_0(i_0)a_1({i_0}^1)]R(i_1, y_1).\end{aligned}$$

This is exactly the first step of our algorithm, if we take the logarithm. The $a_k({i_0}^k)$, as mentioned above, is always only one non-zero $\tilde{p}$ probability.

For $k = 2$ by definition we need to compute the probability

$$\mathbb{P}\left({Z_0}^2 = {i_0}^2, {Y_0}^2 = {y_0}^2\right).$$

It is

$$\begin{aligned}\mathbb{P}\left(Z_0^2 = i_0^2, Y_0^2 = y_0^2\right) &= R(i_0, y_0)R(i_1, y_1)R(i_2, y_2)u_2\\ &= R(i_0, y_0)R(i_1, y_1)R(i_2, y_2)a_2\left(i_0^2\right)u_1\\ &= R(i_0, y_0)R(i_1, y_1)R(i_2, y_2)a_2\left(i_0^2\right)a_1\left(i_0^1\right)\tilde{\alpha}(i_0)\end{aligned}$$

$$d_2(i_2) = \max\left[d_1(i_1)a_2\left(i_0^2\right)\right]R(i_2, y_2),$$

which is the second step of our algorithm, if we take the logarithm, and so on. Some other questions that may arise: is the algorithm really that fast? And how do we derive immediately which non-zero $\tilde{p}$ probability corresponds each time to the $a_k(i_0^k)$ quantity?

We will try to answer these questions by assuming, as an example, that we have two hidden states 1 and 2 (for simplicity reasons). We will present the first three steps of the algorithm so as to show how it works.

For $k = 0$

$$d_0(1) = \log_2(\tilde{\alpha}(1)) + \log_2(R(1, Y_0)), \quad d_0(2) = \log_2(\tilde{\alpha}(2)) + \log_2(R(2, Y_0)).$$

This is the trivial step exactly as in the Viterbi HMM algorithm.

For $k = 1$

$$\begin{aligned}d_1(1) &= \max[d_0(1) + \log_2(\tilde{p}_{(1,0)(1,1)}), d_0(2) + \log_2(\tilde{p}_{(2,0)(1,0)})] + \log_2(R(1, Y_1))\\ d_1(2) &= \max[d_0(1) + \log_2(\tilde{p}_{(1,0)(2,0)}), d_0(2) + \log_2(\tilde{p}_{(2,0)(2,1)})] + \log_2(R(2, Y_1)).\end{aligned}$$

For a better understanding of the quantities $d_k(i_k)$ and $b_k(i_k)$ we mention as an example that the quantity $d_1(1)$ evaluates which trajectory is more likely to have occured, by maximizing the probability. For example, if we have maximum for the first part of $d_1(1)$ this means that it is more likely that at $k = 0$, we started from state 1, in order to visit state 1 at $k = 1$ ($d_1(1)$).

The quantity $b_1(1)$ reveals the state for which we had the maximum, here state 1. Though, we need to recall that the optimal trajectory will be entirely revealed only in the end of the process by backtracking (step 4), as in the classical Viterbi for HMMs.

For a better understanding of the way that the algorithm works we mention as an example that in $d_1(1)$, in the first term of the maximum, we derive the probability $\tilde{p}$, from the structure of the algorithm. Since we have $d_0(1)$ in the first term this gives the first index of the probability, thus $\tilde{p}_{(1,0)}$. By the structure of the algorithm we assume that for $k = 1$ we have $d_1(1)$, which gives the second index, since $i_2$ is now the hidden state 1. The backward-reccurence time $l_2$ is always derived by the definition of the probabilities $\tilde{p}$. Therefore, the probability of the first term of $d_1(1)$ is $\tilde{p}_{(1,0)(1,1)}$. In the same way we can find all the $\tilde{p}$ probabilities of that step.

Let assume that the maximum of $d_1(1)$ is given by its first term (we accept the trajectory $k = 0$, hidden state $= 1$, $k = 1$, hidden state $= 1$) and the maximum of $d_1(2)$ is given by the second term (we accept the trajectory $k = 0$, hidden state $= 2$, $k = 2$, hidden state $= 2$. From step $k = 2$ and on, all the $\tilde{p}$ probabilities will be derived in the following way.

$$d_2(1) = \max[d_1(1) + \log_2(\tilde{p}_{(1,1)(1,2)}), d_1(2) + \log_2(\tilde{p}_{(2,1)(1,0)})] + \log_2(R(1, Y_2))$$
$$d_2(2) = \max[d_1(1) + \log_2(\tilde{p}_{(1,1)(2,0)}), d_1(2) + \log_2(\tilde{p}_{(2,1)(2,2)})] + \log_2(R(2, Y_2)).$$

In $d_2(1)$, the first term is $d_1(1)+\log_2(\tilde{p}_{(1,1)(1,2)})$. The $d_1(1)$ probability provides the information (by the previous maximization) that at $k = 1$ we assumed to be at hidden state 1 and the backward-reccurence time was 1, (the probability of the maximum was $\tilde{p}_{(1,0)(1,1)}$), therefore the second index of the previous maximum probability (1,1) becomes the first index of the next probability, related to $d_1(1)$, $\tilde{p}_{(1,1)(1,2)}$. Since we are now computing $d_2(1)$ this means that the next hidden state is assumed to be $i_2 = 1$ and $l_2 = 2$ by the definition of $\tilde{p}$. Thereby, the first term's $d_2(1)$ probability is now $\tilde{p}_{(1,1)(1,2)}$. The same way we can find all the $\tilde{p}$ probabilities from now and on. It is obvious that the calculations are very simple and using a programming language the algorithm is very fast, due to the fact that there is only one maximization for each $d_k(i_k)$. A few examples are presented in the sequel.

**Example 3.7.** An application of the Viterbi SM1-M0 algorithm in a Poisson distributed HSMM.

We assume that we have two hidden states 1 and 2. The transition matrix is of the standard form and the distributions of the sojourn times are $q_{12}(k) = (e^{-5}5^k)/k!, k \geq 0$ and $q_{21}(k) = (e^{-3}3^k)/k!, k \geq 0$ respectively, where by $q_{ij}$ we denote the semi-Markov kernel. We recall that for this example the transformation in equation (3.1) needs to be done before using the algorithm. The observations H and T are emitted from the hidden states 1,2 with emission probabilities $R(1, H) = 0.2$, $R(1, T) = 0.8$, $R(2, H) = 0.7$ and $R(2, T) = 0.3$. We use the Viterbi algorithm for hidden semi-Markov models, as described above, to define the most likely sequence of hidden states for the observed sequence $TTTTTTTHHHTHHTTT$. Thus, we obtain the probabilities $d_k(i_k)$ and the quantities $b_k(i_k)$ in Table 1, the optimal hidden state sequence and the optimal path probability. Hidden state

TABLE 1. Quantities $d_k(i_k)$ and $b_k(i_k)$ for the example 3.7.

| $k$ | $d_k(1)$ | $d_k(2)$ | $b_k(1)$ | $b_k(2)$ |
|---|---|---|---|---|
| 0 | −1.3219 | −2.7369 | 0 | 0 |
| 1 | −1.6936 | −4.7206 | 1 | 2 |
| 2 | −2.1481 | −6.9310 | 1 | 2 |
| 3 | −2.7221 | −6.5256 | 1 | 1 |
| 4 | −3.4376 | −6.5256 | 1 | 1 |
| 5 | −4.3024 | −6.8475 | 1 | 1 |
| 6 | −5.3158 | −7.4325 | 1 | 1 |
| 7 | −8.4721 | −7.0175 | 1 | 1 |
| 8 | −11.7639 | −7.7787 | 1 | 2 |
| 9 | −11.9385 | −8.7668 | 2 | 2 |
| 10 | −10.4531 | −11.2131 | 2 | 2 |
| 11 | −12.8248 | −12.661 | 1 | 2 |
| 12 | −15.2792 | −14.314 | 1 | 2 |
| 13 | −15.3712 | −17.3754 | 2 | 2 |
| 14 | −15.7429 | −20.6054 | 1 | 2 |
| 15 | −16.1973 | −20.9899 | 1 | 1 |

TABLE 2. Performance of the algorithm.

| observation segment | Algorithm 3.3 | Algorithm in [6] |
|---|---|---|
| 1 | 650/814 | 637/814 |
| 2 | 634/817 | 615/817 |
| 3 | 650/841 | 648/841 |
| 4 | 672/867 | 657/867 |
| 5 | 616/823 | 593/823 |
| 6 | 581/803 | 572/803 |
| 7 | 585/773 | 574/773 |
| 8 | 626/815 | 624/815 |
| 9 | 641/831 | 631/831 |
| 10 | 576/752 | 567/752 |

sequence:

$$q_0^{15} = \{1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1\}$$

with probability $P_{15}^* = 1.3308 \times 10^{-5}$.

Using the same parameters as in example 3.7 we simulated 10 differents segments of the observations T and H emitted from hidden states 1 and 2, in order to test the performance of the algorithm. The results were compared to those obtained from the implementation of the algorithm in [6]. The implementation of the algorithm in [6] was done using the R programming language (see [2]). The score shown in Table 2 represents the hidden states decoded correctly/observation sequence length. We would like to remind though, the much lower complexity of Algorithm 3.3 compared to the one in [6].

## 4. A Viterbi algorithm for a SM1-M1 model

For $(Z, Y)$ a hidden semi-Markov chain of type SM1-M1 let us define the emission probability matrix of the conditional Markov chain Y as

$$R_{i_k;y_{k-1},y_k} = \mathbb{P}(Y_k = y_k \mid Z_k = i_k, Y_{k-1} = y_{k-1}).$$

The only difference from the Viterbi algorithm for a SM1-M0 model, presented above, is that in the second step we replace the emission probability matix $R(i_k, y_k)$ with the emission probability matrix of the form $R(i_k, y_{k-1}, y_k)$ The proposed algorithm is the following one

**Algorithm 4.1.** ***Step 1.*** *Initial conditions. For $k = 0$,*

$$d_0(i_0) = \log_2(\tilde{\alpha}(i_0)) + \log_2(R(i_0, y_0)), \quad b_0(i_0) = 0.$$

***Step 2.*** *For $k \geq 1$,*

$$d_k(i_k) = \max_{i_{k-1}\in E}[d_{k-1}(i_{k-1}) + \log_2(a_k(i_0^k))] + \log_2(R(i_k, y_{k-1}, y_k)),$$
$$b_k(i_k) = \arg\max_{i_{k-1}\in E}[d_{k-1}(i_{k-1}) + \log_2(a_k(i_0^k))].$$

*where*

$$a_k(i_0^k) = \sum_{l_1=0}^{k-1} \sum_{l_2\in\{0,l_1+1\}} \tilde{p}_{\{(i_{k-1},l_1)(i_k,l_2)\}} \times \mathbf{1}_{\{i_{k-1}=\ldots=i_{k-l_1-1}\neq i_{k-l_1-2}\}}, \quad k \geq 1.$$

***Step 3.*** *Termination If $k = T-1$ ($T$ is the length of the observation sequence),*

$$P = \max_{i_{T-1}\in E}[d_{T-1}(i_{T-1})], \quad P^* = 2^{\max_{i_{T-1}\in E}[d_{T-1}(i_{T-1})]},$$

$$q_{T-1} = \arg\max_{i_{T-1}\in E}[d_{T-1}(i_{T-1})].$$

***Step 4.*** *Sequence of states with backward-forward steps*

$$q_{k-1} = b_k(q_k), \quad k = T-1, \ldots, 1.$$

In Appendix A.3 we provide a pseudocode for Algorithm 4.1. Remarks 3.4–3.6 hold true for this algorithm too.

**Example 4.2.** An application of the Viterbi SM1-M1 algorithm in DNA Analysis (for DNA Analysis examples, see also [7]).

Real DNA sequences can be described by a hidden semi-Markov model with hidden states representing different types of nucleotide composition. Consider a SM1-M1 that includes two hidden states 1 for higher and 2 for lower $C+G$ content, respectively. The transition probability matrix is of the standard form and

TABLE 3. Quantities $d_k(i_k)$ and $b_k(i_k)$ for the example 4.2.

| $k$ | $d_k(1)$ | $d_k(2)$ | $b_k(1)$ | $b_k(2)$ |
|---|---|---|---|---|
| 0 | −2.3219 | −3.3219 | 0 | 0 |
| 1 | −3.5977 | −7.9302 | 1 | 1 |
| 2 | −4.9072 | −8.2540 | 1 | 1 |
| 3 | −9.3252 | −6.1392 | 1 | 1 |
| 4 | −8.0016 | −9.6131 | 2 | 2 |
| 5 | −10.0848 | −12.0249 | 1 | 1 |
| 6 | −12.8094 | −12.8343 | 1 | 1 |
| 7 | −14.0187 | −17.3632 | 2 | 1 |
| 8 | −15.2945 | −19.6269 | 1 | 1 |
| 9 | −21.3410 | −16.8216 | 1 | 1 |
| 10 | −20.0060 | −19.9736 | 2 | 2 |
| 11 | −23.0891 | −24.0676 | 1 | 2 |
| 12 | −27.3987 | −27.4236 | 1 | 1 |
| 13 | −31.6080 | −28.6306 | 2 | 1 |
| 14 | −31.8150 | −31.5196 | 2 | 2 |
| 15 | −35.8982 | −37.9355 | 1 | 2 |

the probability density functions of the holding times $t_{12}$ and $t_{21}$ are $q_{12}(k) = 0.59^{(k-1)^{1.2}} - 0.59^{k^{1.2}}$ and $q_{21}(k) = 0.45^{(k-1)^{0.74}} - 0.45^{k^{0.74}}$ (discrete Weibull distribution). Nucleotides T (thymine), C (cytosine), A (adenine), G (guanine) are emitted from states 1 and 2 for k=0 with probabilities 0.1,0.4,0.3,0.2 and 0.5,0.2,0.2,0.1 respectively and for $k \geq 1$ from the following emission probability matrices (emitted from state 1 and state 2, respectively)

$$\boldsymbol{R_1} = \begin{array}{c} \\ T \\ C \\ A \\ G \end{array} \begin{array}{c} \begin{array}{cccc} T & C & A & G \end{array} \\ \begin{pmatrix} 0.1 & 0.2 & 0.5 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0.7 \\ 0.2 & 0.1 & 0.4 & 0.3 \\ 0.1 & 0.8 & 0.03 & 0.007 \end{pmatrix} \end{array}, \quad \boldsymbol{R_2} = \begin{array}{c} \\ T \\ C \\ A \\ G \end{array} \begin{array}{c} \begin{array}{cccc} T & C & A & G \end{array} \\ \begin{pmatrix} 0.4 & 0.1 & 0.2 & 0.3 \\ 0.8 & 0.1 & 0.05 & 0.05 \\ 0.25 & 0.3 & 0.15 & 0.3 \\ 0.02 & 0.08 & 0.7 & 0.2 \end{pmatrix} \end{array}.$$

We use the Viterbi algorithm for SM1-M1 to define the most likely sequence of hidden states for the observed sequence $CGCTAAGCGATCCTGT$. Thus, we obtain the probabilities $d_k(i_k)$ and the quantities $b_k(i_k)$ in Table 3, the optimal hidden state sequence and the optimal path's probability. Hidden state sequence:

$$q_0^{15} = \{1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1\}$$

with probability $P_{15}^* = 1.5616 \times 10^{-11}$.

## 5. CONCLUSIONS

These algorithms are original and have an innovative and useful form. They can be used in a lot of applications, so as to decode an unobserved hidden semi-Markov process, via an observed process. Furthermore, their complexity is the same as in the Viterbi HMMs case, *i.e.* $O(Td^2)$, which has not been achieved until now. Their only difference from the Viterbi in the HMM case is that instead of the transition probabilities of the embedded Markov chain, now we use the $\tilde{p}_{(i,t_1)(j,t_2)}$, the transition probabilities of the associated backward recurrence Markov chain $(Z, U)$.

## Appendix A

### A.1 Viterbi algorithm for HMMs [8]

**Algorithm A.1.** ***Step 1.*** *Initial conditions. For $k = 1$,*

$$d_1(i_1) = \log_2(\tilde{\alpha}(i_1)) + \log_2(R(i_1, y_1)), \quad b_1(i_1) = 0.$$

***Step 2.*** *For $k > 1$,*

$$d_k(i_k) = \max_{i_{k-1} \in E} [d_{k-1}(i_{k-1}) + \log_2(p_{i_{k-1},i_k})] + \log_2(R(i_k, y_k)),$$
$$b_k(i_k) = \arg\max_{i_{k-1} \in E} [d_{k-1}(i_{k-1}) + \log_2(p_{i_{k-1},i_k})]$$

***Step 3.*** *Termination If $k = T$ ($T$ is the number of observations),*

$$P = \max_{i_T \in E} [d_T(i_T)], \quad P^* = 2^{\max_{i_T \in E}[d_T(i_T)]}, \quad q_T = \arg\max_{i_T \in E} [d_T(i_T)],$$

*where $p_{i_{k-1},i_k}$ is the transition probability matrix.* ***Step 4.*** *Sequence of states with backward-forward steps*

$$q_k = b_{k+1}(q_{k+1}), \quad k = T-1, \ldots, 1.$$

### A.2 A pseudocode for the new algorithm of type SM1-M0

**Input**

The observation sequence $Y_0^{T-1}$ of length T. The emission probabilities $R[i, Y_k]$ for all hidden states $i$, $i \in \{1, \ldots d\}$. The kernel probabilities $q_{ij}[n]$. The survival functions of the sojourn times, $\bar{H}_i[n]$, for all i. The $\tilde{p}_{(i,l)(j,m)}$ probabilities of the form $\tilde{p}_{(i,l)(j,m)} = q_{ij}(l+1)/\bar{H}_i(l)$, if $i \neq j$, or $\tilde{p}_{(i,l)(j,m)} = \bar{H}_i(l+1)/\bar{H}_i(l)$ if $i = j$. The initial probabilities $\tilde{\alpha}[i]$ for all i.

```
// Algorithm, Step 1 ($k = 0$)
For $j = 1$ to $d$ Do
  $d_0[j] = \mathrm{Log2}[\tilde{\alpha}[j]] + \mathrm{Log2}[R[j, Y_0]]$
End For
//Step 2 (first for $k = 1$)
For $j = 1$ to $d$ Do
  For $i = 1$ to $d$ Do
  $l[1, i, j] = 0$ //1 since $k = 1$
    If($i = j$) Then
```

```
            $m[1,i,j] = 1$
        Else $(i \neq j)$
            $m[1,i,j] = 0$
        End If
        $a[i,j] = d_0[i] + \text{Log2}[\tilde{p}_{(i,l[1,i,j])(j,m[1,i,j])}] + \text{Log2}[R[j,Y_1]]$
        $d_1[j] = \max(\text{of quantities } a[i,j]\ )$
        $b_1[j] = \arg\max[d_1[j]]$
    End For
End For
//Step 2 (continue for $k \geq 2$)
For $k = 2$ to $T-1$ Do
    For $j = 1$ to d Do
        For $i = 1$ to d Do
            $l[k,i,j] = m[k-1, b_{k-1}[i], i]$
            If $i = j$ Then
                $m[k,i,j] = m[k-1, b_{k-1}[i], i] + 1$
            Else $i \neq j$
                $m[k,i,j] = 0$
            End If
            $a[i,j,k] = d_{k-1}[i] + \text{Log2}[\tilde{p}_{(i,l[k,i,j])(j,m[k,i,j])} + \text{Log2}[R[j,Y_k]]$
            $d_k[j] = \max(\text{of the quantities } a[i,j,k])$
            $b_k[j] = \text{argmax}[d_k[j]]$
        End For
    End For
End For
//Step 3, quantities $d_{T-1}[j]$ already evaluated, for all j, in the previous step
Do
    Quantity$P = \max[d_{T-1}[j]]$
    $P^* = 2^P$ // Optimal path probability
    $Q_{T-1} = \text{argmax}[d_{T-1}[j]]$ // last hidden state
//Backward procedure to reveal the hidden states $Q_k$
For $k = T-1$ to 1 Do
    $Q_{k-1} = b_k[Q_k]$
End For
```

**Complexity** As it can be seen in the above pseudocode, there exist four different loops, two of which are nested. Therefore it is,

$$O\left(d + d^2 + (T-2)d^2 + (T-1)\right) = O\left((T-1)d^2\right) < O\left(Td^2\right).$$

Hence, the complexity is $O(Td^2)$.

The term $(T-2)d^2$ results from the third nested loop, since the length of the observation sequence is $T$ and we start from $k = 2$. The analysis of the algorithm presented before the Example 3.4, provides an easier way to understand the structure of the algorithm and derive the complexity. For example in this analysis, we realize that for two hidden states, for $k = 1$ we need to performe $2^2$ computations. The same holds for $k = 2,\ldots$ Taking into consideration the initial step and assuming that the observation sequence length is $T$ then we can understand why the complexity is $O(Td^2)$.

## A.3 A pseudocode for the new algorithm of type SM1-M1

**Input** The same as before. The only difference is that we need to define the emission probabilities $R[i, Y_k]$, which will be used only in the initial step, and furthermore the emission probabilities $R[i, Y_{k-1}, Y_k]$.

//Algorithm, Step 1 ($k = 0$), the same as in A.2

//Step 2 (first for $k = 1$), the same as in pseudocode A.2, replace only $R[j, Y_1]$ by $R[j, Y_0, Y_1]$, in quantities $a[i, j]$.

//Step 2 (continue for $k \geq 2$), the same as in pseudocode A.2, replace only $R[j, Y_k]$ by $R[j, Y_{k-1}, Y_k]$, in quantities $a[i, j, k]$.

//Step 3, from now and on the pseudocode is exactly the same as in A.2.

*Acknowledgements.* The authors appreciate the comments of the anonymous reviewer which contributed to improve the quality of the publication and would like to thank the reviewer for the thorough and helpful review. They would also like to thank Jan Bulla who was always very willing to provide his software about the paper in [2], which helped us compare the performance of the Algorithm 3.3 to that one in [6].

## References

[1] V.S. Barbu and N. Limnios, *Semi-Markov chains and hidden semi-Markov models toward applications*. Springer, New York (2008).

[2] J. Bulla, I. Bulla and O. Nenadić, An R package for analyzing hidden semi-Markov models. *Comput. Stat. Data Anal.* **54** (2010) 611–619.

[3] O. Chryssaphinou, M. Karaliopoulou and N. Limnios, On discrete time semi-Markov chains and applications in words occurrences. *Commun. Stat. Theory Methods* **37** (2008) 1306–1322.

[4] N.A. Dasu, *Implementation of hidden semi-Markov models*. Th.D. Thesis, University of Nevada, Las Vegas (2011).

[5] M. Dong and D. He, Hidden semi-Markov model-based methodology for multi-sensor equipment health diagnosis and prognosis. *Eur. J. Oper. Res.* **178** (2011) 858–878.

[6] Y. Guédon, Estimating hidden semi-Markov chains from discrete sequences. *J. Comput. Graph. Stat.* **12** (2003) 604–639.

[7] T. Koski, *Hidden Markov models for bioinformatics*. Kluwer, Dordrecht (2001).

[8] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **77** (1989) 257–286.

[9] A. Taghva, Hidden semi-Markov models in the computerized decoding of microelectrode recording data for deep brain stimulator placement. *World Neurosurg.* **75** (2011) 758–763.

[10] M. Tang and P. Di Cristo, Backward Viterbi beam search for utilizing dynamic task complexity information, in *Proc. Conference International Speech Communication Association* (2008) 2090–2093.

[11] S.Z. Yu, Hidden semi-Markov models. *Artif. Intell.* **174** (2010) 215–243.