

Efficiency and Accuracy Trade-Offs in Process Detection

Annarita Giani

Thayer School of Engineering, Dartmouth College Hanover, NH, US 03755
annarita.giani@dartmouth.edu

ABSTRACT

Hidden Discrete Event Systems Models (HDESM) are discrete event dynamical system models whose underlying internal state spaces are not directly observable. Observations on such systems are artifacts of the hidden, internal states and are not deterministically or uniquely associated with the hidden states. The distribution of an observation of a HDESM is typically given by a probability distribution conditioned on the hidden state of the system. Classical linear systems, Hidden Markov Models (HMM) and certain types of Petri Net models are examples of HDESM's.

A major challenge in working with this type of model is the estimation of HDESM's hidden states based on a sequence of observations. In some cases, well-known algorithms can be used to solve this problem. In many cases of practical interest, however, the complexity of those algorithms is too high to be practical. New ideas and algorithms are therefore needed for effective solutions to the state estimation problem.

In this paper we will investigate sub-classes of HDESM's whose structure would allow efficient state estimation algorithms to exist. Such structures could be related to the sparsity and/or equivalence class structure of transition dynamics within the underlying discrete event system. Efficient algorithms that compute approximate solutions will be investigated with the goal of understanding the trade-offs between computational efficiency and estimation accuracy. Ideas on how to implement such trade-offs also are proposed.

Keywords: Discrete Event Systems, Hidden State Estimation, Viterbi Algorithm, Process Detection

1. INTRODUCTION

This paper addresses questions related to the general problem of identifying and tracking discrete event systems that are only partially observable. We will call models of such systems *Hidden Discrete Event System Models* (HDESM). An example is a Markov chain, with states $X_1, X_2, \dots, X_n, \dots$, whose states X_i are observable only through a stochastic emission of finitely many observations: $Y_i = \phi(X_i)$. Such processes are known as discrete time Hidden Markov Models (HMM) [1]. There are two important problems:

1. given a set of observation and models (or processes) determine the most likely process-to-observation association;
2. given a sequence of observations and an HDESM process, determine the sequence of non-observable states that "best" explains the observations.

There are several possible optimal criteria for a hidden state sequence. A well known technique for finding the best state sequence, based on dynamic programming methods, is the Viterbi algorithm [2]. It maximizes $P(Q|O, \lambda)$, the probability of the hidden state sequence given the sequence of observations and the complete parameter set of the model. This well-known algorithm executes in time N^2T (quadratic in the number of states of the Markov chain and linear in the length of the sequence of observations). Unfortunately, in many applications N can be quite large. For example, N is exponential in the number of places of a Stochastic Petri Net, whose semantics is a HMM. Under such conditions the Viterbi algorithm becomes extremely inefficient.

There is a wide variety of applications that can be modeled as HDESM and for which specific efficient (sub-quadratic in the number of states) hidden states estimation algorithms are needed. Worthy of mention are the following:

- Tracking several targets having stochastic trajectories given a sequence of consecutive radar position snapshots is a classic application. If each snapshot does not reveal the identity of the specific target, we first need to associate, in a unique way, sets of observations to existing tracks (forming hypotheses) and then examine each single hypothesized track to reconstruct the trajectory of the corresponding vehicle.
- The most popular algorithms for automatic speech recognition are based on statistical methods. They generate a sequence of word hypotheses from an acoustic signal.
- Model-based monitoring and diagnostic procedures for complex systems rely on these techniques as well. An example of particular interest is the analysis of reliability and availability of electrical power networks in which sequences of events can lead on occasion to blackouts. In this specific case we must consider an evolution matrix in which some of the transitions have a very low probability to occur.
- Parallel activities or concurrency can be easily expressed in terms of Petri Nets if information is shared among several processors.
- Performance evaluation, risk analysis and queuing networks often are modeled as a variation of Stochastic Petri Nets (SPN). The underlying Markov chain associated with the Marking Graph is used to compute statistics.
- The liveness and safeness properties of a Petri net often are used as correctness criteria in communication protocols.
- In the field of work flow management, business process logic is explicitly represented. Petri nets can be used as a design language for the specification of complex work flows, and can provide powerful analysis techniques that can be used to verify the correctness of work flow procedures.

Without a Viterbi-like algorithm the computational problem of determining which sequence of process states is most likely to account for an observed event sequence would be exponential in most cases. In fact the Viterbi class of algorithms uses dynamic programming and the additive properties of log-likelihood functions to make the computation tractable.

While looking for fast algorithms for broad classes of Discrete Event Systems (DES) could be too ambitious, it is certainly of great interest to investigate sub-classes of such systems for which efficient algorithms exist. Once such sub-classes are discovered, we can design and implement efficient algorithms, analyze their computational complexity and verify experimentally whether the theoretical conclusions are confirmed by the empirical evidence.

There might be cases for which efficient algorithms can be derived by relaxing the structure of the ideal model. The Viterbi algorithm would, in such cases, run comparatively more efficiently on the approximated model at the expense of a “loss of precision”. This paper address the problem of investigating and deciding the best trade-off between the computational complexity of the algorithm and the performance of the approximated model with respect to a base case that is assumed to be “correct”. In particular we will introduce the idea of accuracy and complexity functions, which are used to establish trade-offs between complexity of the estimation algorithm and accuracy of the solution.

In the next section we will discuss previous work done in this framework. Section 3 presents the idea of Optimization Function. Section 4 shows how this concept can be applied to the real case of tracking physical objects. Conclusions follow in Section 5 and future work is presented at the end.

2. RELATED WORK

The problem of inferring a process’ state history from observations of the process is of growing importance in the field of computer science and engineering. Stochastic models are a powerful tool used in many new technologies. The theory of HMM, for instance, is essential to the area of pattern recognition [3].

Petri Nets [4] were first introduced to describe and analyze Discrete Event Systems, but soon were extended to other applications. The formalism, in fact, has been enlarged to Continuous Petri Nets [5] and to Hybrid Petri Nets [6].

Stochastic Petri Nets [7, 8] are generally used to estimate statistics of interest in performance evaluation. Usually Petri net models do not simplify the evaluation since they rely on an underlying Markov chain. There are cases in which the particular structure of the transition matrix is exploited to enhance the statistical analysis of the system. Methodologies for modeling a system composed of parallel activities with synchronization points have been developed. In [9, 10] a formalism called Stochastic Automata Network was introduced and studied under Markovian assumptions. The above approach based on a state-transition representation of a parallel system replaces other approaches based on SPNs for the computation of performance indexes [11]. SPNs, in fact, usually do not lead to compact models, as they are computationally expensive and they provide few results for dealing with state-space explosion.

In [12] the authors introduced a new model that allows the specification, simulation and estimation of hidden states in mixed systems. The model, called Calculus of Stochastic System (CSS), concentrates on the simulation of systems containing both stochastic and non-stochastic components. They introduce a general combinator that permits the specification of an arbitrary mixed system in terms of elementary components of only two types. They also use modularity for the estimation of hidden quantities of interest. They extend the notions of generalized likelihood and conditional likelihood to mixed systems and show that the Viterbi algorithm can be generalized accordingly. The above structure was then employed in the effective construction of Partially Stochastic Petri Nets used to solve the problem of fault detection and diagnosis in distributed systems [13, 14].

Net unfolding techniques associated with partial-order semantics were used to provide a generalized Viterbi algorithm for maximum-likelihood estimation of hidden state history from observed alarms. The work was motivated by the problem of inferring, from measurements, the hidden internal state of a distributed asynchronous system. Since global knowledge of the state space, together with transition probabilities, is impractical for large networks, standard Stochastic Petri Nets are not the best model formulation. It is desirable that regions of the net that are not directly in competition generate independent behavior. When transitions are not competing, it is not randomized, but unknown, which one fires first; only partial ordering is randomized. This leads to the definition of Partially Stochastic Petri Nets. Ordinary Stochastic Petri Nets instead assume that all enabled transitions are in competition at a given time even when they do not use the same resources. The great advantage of partial order semantics is that the full transition probability matrix on the marking graph is replaced by very a few factors that are easy to compute. This approach bypasses the problem of state explosion.

A distributed state reconstruction algorithm for a discrete event system, described as the parallel composition of subsystems, is presented in [15]. Given a number of sequences of observables, the problem of recovering the most likely global trajectory was addressed by designing an agent for each sub-system. These agents know their local transition and run a Viterbi algorithm locally. Synchronization of the information with the rest of the system leads to a distributed Viterbi algorithm.

The problem of estimating the marking of a Petri Net based on observations occurs in many settings. In particular, the issue of state estimation under partial state observation has been discussed in the literature. If the system is represented as a finite automata an observer can be designed for a partially observed system [16]. Approaches for building observers that allow the reconstruction of the state of finite automata after a sequence of bounded length have been proposed as well [17]. Also, it has been proven that the marking consistent with an observed sequence can be described by a constraint set of linear inequalities with a structure that does not change as the length of the observed sequence increases [18].

Despite intense research efforts in the area of hidden parameter estimation, there still is lack of appropriate characterization of trade-off between accuracy of the hidden state estimated sequence and the time needed to obtain the solution.

3. OPTIMIZATION CRITERIA

Intuitively a structural simplification (e.g. reduction of the number of possible hidden transitions and/or states) of a given HDESM will allow a reduction in the computational complexity of the state estimation process at the expense of the accuracy with respect to the original model.

There are situations in which it is imperative to have an immediate response. Consider, for example, a high-danger detection process in which we prefer the risk of an inaccurate response to the risk of getting a correct response late. Detecting problems in a nuclear plant is one of these cases. On the other hand there might be cases in which we are willing to wait and we need a perfect hidden state sequence. In the framework of target surveillance or web monitoring, we may be interested in quantifying the risk of not having accurate information about the hidden state of the system. Thus, a reasonable goal would be to keep the risk bounded at the cost of compromising between the computing time necessary to achieve a certain situation awareness and its accuracy. In general this trade-off will originate from the specific application to which the model refers. If a Viterbi algorithm is not viable for a model due to high cost, the model could be approximated by a model belonging to a family of sub-models. Different HDESM can be obtained through a process of refinement of the state space called vertical mapping in which the state space changes or through a different filtering called horizontal mapping, in which the state space is kept the same.

The notion of optimality that we propose is application dependent. Accuracy of the solution and complexity of the estimation algorithm cannot be optimized at the same time, and we propose to optimize an objective function that is application dependent. Every time that a hidden state sequence must be computed, the model that best fits the need of the user will be considered. That model might be different from the original, but it is the one that best satisfies the user in terms of the trade-off between accuracy and time needed to run the estimation algorithm. The algorithm that we are proposing might be of great interest in implementation of a Process Query System [19] where sets of models are submitted as queries to the system, which returns the instance of the model that most likely generated a certain sequence of observations.

Although this approach is applicable to any type of HDESM's, for simplicity, we will concentrate on HMM's. Extensions to other type of HDESM's will be proposed in later papers. Given a HMM we can build a continuous family of approximation models M_δ ($0 \leq \delta \leq 1$) where M_0 is the extreme case that represents exactly the model M and M_1 is the worst possible approximation. The idea is that in going from $\delta = 0$ to $\delta = 1$ the hidden state sequence estimation becomes less and less correct. On the other hand the complexity of the hidden state estimation algorithm decreases and a solution can be obtained in a shorter time. Deciding which δ is appropriate for each circumstance is not obvious. We develop some functions a-priori, based on the model and the family of chosen sub-models, and then leave the user to decide the accuracy he requires. An accuracy function $A(\delta)$ describes how what accuracy of the hidden state estimation varies in the family of models M_δ . A complexity function $C(\delta)$ describes how complex the estimation algorithm is for each member of the family. Let us keep the number of observation fixed and see how we can build the two functions.

3.1. Accuracy Function

$A(\delta)$ takes as a parameter a value that characterizes each of the sub-models and for each of them returns some sort of accuracy of the estimation algorithm. When talking about accuracy it is natural to introduce the concept of distance from the correct case. To do so, we must decide what is the metric space we are going to consider, and as we will see, there are different choices. The first step consists in running the exact hidden state estimation algorithm. Given a sequence of L observations, (o_1, \dots, o_L) run the algorithm and compute $(s_1^*, \dots, s_L^*) = \mathbf{s}^*$, the exact hidden state sequence that best associate with the sequence of observations. Once we have the perfect solution we can go through the family M_δ and see, for each sub-model, how the hidden state sequence differs from the exact one. This difference provides the base for constructing the accuracy function of the model. Summarizing the two steps:

1. Let the real hidden state estimation algorithm run to obtain the real sequence of hidden states \mathbf{s}^* . This case corresponds to $\delta = 0$.
2. From a theoretical point of view δ , the parameter that identifies each sub-model, can vary continuously in the interval $[0, 1]$. What we have then is a continuous family of sub-models. To make experiments possible, the interval can be discretized into D samples $(\delta_0, \dots, \delta_D)$ in which $\delta_0 = 0$ and $\delta_D = 1$. For each of the models M_{δ_i} with $0 \leq i \leq D$ we compute the hidden state algorithm obtaining the sequence of state (s_1^i, \dots, s_L^i) . We will call each of these sequences \mathbf{s}^i . The next step is deciding how each of the \mathbf{s}^i differs from the exact sequence \mathbf{s}^* . The idea is to estimate $d(\mathbf{s}^i, \mathbf{s}^*)$. We can naturally define this function as

$$d(\mathbf{s}^*, \mathbf{s}^i) = d((s_1^*, \dots, s_L^*), (s_1^i, \dots, s_L^i)) = \frac{1}{L} \{d(s_1^*, s_1^i) + \dots + d(s_L^*, s_L^i)\} = \frac{1}{L} \sum_{j=1}^L d(s_j^*, s_j^i)$$

but now the problem of deciding which metric to use arises. If the state space is by nature embedded in a metric space, we may associate each state with a point in the space and then consider the usual distance between points of the metric space. Tracking problems in which each state is represented by a cell in a Cartesian space is an example. This is not the general case, however, so a broader notion of distance between sequences of states must be introduced. Also there are cases in which geographical vicinity is not representative of the distance between states since embedding in Euclidean space does not capture the structure of the model. If two roads run on different side of a river, for example, and a car is driving on one of the roads, it is not easy to cross to the road on the other side. The car has to drive until it finds a bridge that could be far away. This is an example in which Cartesian vicinity does not reflect model vicinity. Also, even if the set of states can easily be embedded into a Cartesian space, it might be unreasonable to require the symmetry since there might be cases in which it is easy to go in one direction but impossible to go in the opposite direction (one-way streets are an example). There has been some research in the field of calculating distances between Hidden Markov Models. In [20] for example the authors investigate two methods, the first one based on Euclidean distance, the second one on Kullback-Liebler distance, which measures the discriminating power of the probability metric applied to the space of feature sequences induced by the models.

In the following we propose some different notions of distance between two states of the model.

- $d(s_j^*, s_j^i) = 0$ if $s_j^* = s_j^i$
 $d(s_j^*, s_j^i) = 1$ otherwise
- $d(s_j^*, s_j^i)$ is in some way related to the distance between the two states in the underlying graph. An example could be the minimum number of edges between the two nodes or the probability of transitioning from one node to the other.
- $d(s_j^*, s_j^i)$ is related to the entropy of the possible hidden state sequences that may have produced a certain sequence of observations, $H(S|O)$. This quantity measures the uncertainty of the state sequences.

We can relate the accuracy of each subsystem to the distances between the exact hidden state sequence and the one computed from each sub-system and from that build the entire accuracy function. Intuitively accuracy should be inversely related to distance so for each δ_i a value $A(\delta_i)$ is introduced as

$$A(\delta_i) = \frac{1}{1 + d(\mathbf{s}^*, \mathbf{s}^i)}$$

In this way $\max_i A(\delta_i) = 1$ for $i = 1, \dots, D$. Once all the values $A(\delta_i)$ are calculated, the continuous function $A(\delta)$ where $0 \leq \delta \leq 1$ can be computed through an interpolation of the values.

The weakness of this approach is that the accuracy depends on the sequence of observations. Ideally this function would depend only on the model.

3.2. Complexity Function

This function represents how complex the estimation algorithm is. It mostly depends on the number of connections between states, so as we move from a fully connected situation ($\delta = 0$) to the opposite case ($\delta = 1$), the complexity certainly must decrease. A reasonable quantity to represent the complexity is the number of operations needed to compute the Viterbi algorithm. This number depends on

N , the number of hidden states of the Markov chain,

T , the number of possible observations,

$h_i = \# \{s \mid P(o_i|s) \neq 0\}$, the number of states from which we can observe o_i , $i = 1 \dots T$,

$$h = \max_i h_i, i = 1 \dots T,$$

$$k_i, \text{ the number of predecessors of state } s_i, i = 1 \dots N,$$

$$k = \max_i k_i, i = 1 \dots N,$$

where some of the above quantities are different for each sub-model. So the complexity function is $C(\delta) = C(h(\delta), k(\delta))$. As for the accuracy function we can compute the number of operations needed to perform the algorithm for each of the models $C(\delta_i)$ with $0 \leq i \leq D$ and interpolate it to calculate $C(\delta)$.

3.3. Trade-off between accuracy and complexity

What has been done so far is totally dependent on the model and the structure of the sub-models. Nothing of what was said is influenced by the user request on how good the solution should be. The user, knowing the details of the situation he is trying to investigate, should have the power to decide the type of solution he wants. This is done by requesting that the accuracy of the solution be above a threshold a_0 . Of course, small a_0 means having a quick but less accurate solution. On the other hand, if a_0 is large, it means that the user is willing to wait more time to get a more accurate solution. So the only parameter that is situational awareness is a_0 . The index of the sub-model the best fits the user need is

$$\arg \min_i C(\delta_i)$$

with the condition that $A(\delta_i) > a_0$. It will be the one used to calculate the hidden state estimation.

4. A CASE STUDY: TRACKING PHYSICAL OBJECTS

In this section we present a direct application of the approach previously developed. We focus on obtaining a hierarchy of models whose computational complexity of the hidden parameter estimation problem can be controlled. Intuitively those models have different accuracy. In this paper we do not provide a quantitative analysis of their accuracy but only of their complexity. A complete trade-off analysis will be accomplished in future works.

4.1. Problem definition

A vehicle that moves in a plane describes a continuous trajectory that could be represented by a parameterized curve. If we discretize the plane into cells and we take snapshots of the position of the target at regular intervals of time, we obtain essentially a sequence of quantized samples of the trajectory. Assuming that the determination of the position of the target is noisy, the problem of determining the best sequence of states that explains the observations becomes equivalent to a problem of filtering digitized random signals transmitted over noisy channels. Thus, the method can be seen as a technique to perform efficient filtering. In fact, it has already been observed that under certain conditions Hidden Markov Models for single target tracking are equivalent to Kalman Filters.²¹

The sparsity of the transition matrix can be used to reduce the complexity of the estimation procedure, since lower number of connections between nodes will allow faster algorithms. A sparsity assumption is reasonable in physical object tracking since the velocity of an object is bounded. In this case, efficiency can be traded-off against the quality of filtering that intuitively is related to the accuracy of the probabilistic model that we have chosen for the vehicle kinematics (the value of our threshold). We implemented a Viterbi algorithm optimized for the special case of sparse Markov Models. This optimization was aimed at reducing the overall complexity of the algorithm.

A family of approximated Markov Models for the problem of tracking a ground vehicle has been considered. The models were built such that they perform a Brownian motion (random walk with bounded velocity) in a bounded portion of the plane. They were obtained by first discretizing the observation space into cells of fixed dimensions (cell model) and then by calculating the probabilities $P(B|A)$ of the vehicle being in cell B given that Δt units of time earlier it was in cell A . Assuming that at time t the target in question lies within cell A , then several degrees of connectivity in the Markov chain can be attained by approximating the bivariate probability density associated with the random walk starting from A at time t with a probability density that is nonzero

only on bounded neighborhoods of A . This corresponds to neglecting the probability that the target reaches a very distant cell within a given time frame.

This seemed a fair assumption considering that the velocity of the target is known to be bounded. In general, if the random walk is modeled using a bivariate Gaussian density (as typically results in a symmetric random walk on the plane) it is possible to replace the Gaussian bell with a surface that is zero on the points P on which the values of the Gaussian were below a given threshold δ . Different values of δ will produce different surfaces (non normalized) that are nonzero only on a finite number of cells (bounded neighborhood of C). The smaller the value of δ , the higher will be the number of cells on which the surface is non null. Hence, we can control the degree of connectivity of the chain by varying δ . The two limit values $\delta \rightarrow \text{Gaussian}(0)$ and $\delta \rightarrow 0$ will produce respectively a Markov chain with no edges and a fully-connected Markov chain.

4.2. Implementation

In our preliminary study of the problem of reconstructing the state trajectory of a ground vehicle from noisy measurements, we employed first Kalman Filters [22] and then Hidden Markov Models [23]. In the case of HMM's, the analysis of the complexity of the Viterbi algorithm on a given sequence of observations, as well as empirical evidence (see Section 4.3), has shown that two factors seem to play a crucial role in making the computation tractable. The first one is the sparsity of the Markov chain associated with the HMM together with very narrow conditional probability distributions of the observations given a hidden state, $P(O|S)$ (an ideal situation would be represented by a constant number of observations for which $P(O|S) \neq 0$ for all S). The second one is low cardinality of the state space, i.e., a small number of possible hidden states of the Markov chain. Those factors are typically exploited in the design of efficiently decodable stochastic error-correcting codes [24] and heavily affect not only the efficiency of the processing algorithms, but also the accuracy of the computed solutions.

We use these two factors to explore more efficient estimation algorithms for the vehicles position. We produced six different implementation of the Viterbi algorithm. Two models and three different data structure have been considered.

In this analysis six implementations of the Viterbi algorithm were actually produced. Two different models and three different data structure have been considered.

4.2.1. Models

- *Uniform probability to the neighbors.* The first model assigns uniform probabilities for transitions to neighbor cells, producing a Hidden Markov Model graph whose transition matrix is very sparse. A family of sub-models can be easily obtained varying the degree d of adjacency considered. The model M_1 , for instance, represents degree 1 connectivity. It means that, if the target is in cell A , the probability of being on a cell at the next step is zero everywhere except for the 8 neighbor cells surrounding A , and A itself. Extending non-zero probability to the crown around them leads to model M_2 . Increasing d , the models become more and more accurate but also the complexity of the estimation algorithm grows. The best trade-off is situation dependent.
- *Gaussian Distribution.* The second model is based on a bivariate Gaussian distribution defining the transition probabilities. In particular, consider a vehicle having velocity v and performing a bi-dimensional and symmetric Brownian Motion. Given a region B of the space we can compute the probability of being in that area given that at the previous step the vehicle was in area A as

$$P_{B|A} = \int_B \frac{1}{(2\pi)^{n/2} \sqrt{|C|}} \exp\left(-\frac{1}{2}x^T C^{-1}x\right)$$

where C is the Covariance Matrix $C = \begin{pmatrix} v\Delta t & 0 \\ 0 & v\Delta t \end{pmatrix}$

In the real case $P_{B|A} \neq 0 \forall B$ which means that there is always a non-zero probability to have a transition from a cell to any other cell. This concept transfers into a completely connected transition matrix. We decided to put a threshold δ_0 on the probability transition such that if $P_{B|A} < \delta_0$ it collapses it to zero. Once this process is finished we re-normalize the entire non-zero probability transition. In this way many edges of the transition graphs are removed and the estimation algorithm is faster. Approximating the original model with a sparse one gives an approximated, but faster, solution. The family of sub-model is naturally obtained varying the parameter δ_0 in the interval $[0, \text{Gaussian}(0)]$.

4.2.2. Algorithms

Our idea of optimizing the Viterbi algorithm is based on two factors. The first is, as we said, the sparsity of the Markov chain (or better of its underlying graph). In this way only a constant small number of “predecessors” of a given state must be considered. The second factor is that the conditional distributions of the observations given a state are concentrated on a few states. This allows us to predetermine the number of states to look at for each observation.

- *Naïve implementation.* The first, naïve implementation follows the general dynamic programming formulation of the algorithm and so builds a $N \times T$ matrix for the determination of the “optimal” sequence of states that “explain” the given sequence of observations. This straightforward implementation, though correct, did not seem to take advantage of the sparse nature of the models we wanted to use. In fact at each step of the Viterbi algorithm a complete search must be performed to find all the predecessors.
- *Fast implementation.* A second version was realized to exploit sparsity. The result was an algorithm that builds an array of T arrays. The idea is that given an observation, say O , we can assume that $P(O|S)$, the probability that the target will be in cell O when it is observed given that the model is in the hidden state S will be nonzero only for states S , associated with a bounded neighborhood of O . Thus, when computing an iteration of the Viterbi algorithm, we do not need to consider all the possible states but only those for which $P(O|S) \neq 0$. Moreover, those collections of sets can be determined in advance, once and for all, at the time of building the model.
- *Super-fast implementation.* The third solution came as a further improvement over the second one. In fact, in the second approach, the matrix is an array of arrays where the number of columns corresponds to the number of observations and, for each observation, we had an array with all the states for which the probability of having that observation was nonzero. The improvement consisted of replacing the array of arrays with an array of Hash Tables of states indexed by their label*. This way, during the examination of observation O at time t and given a certain state S , such that $P(O|S) \neq 0$, it was possible to reduce the search time for all the predecessors of S in the set of states associated with the previously examined observation (at time $t - 1$). This gain is clearly at the expense of an increase in the overhead to set up a sophisticated data structure in place of a simple array. Therefore we expect to observe significant benefits in Markov Models that have large numbers of states and, for each observation, a significantly smaller number of states to consider. Besides Hash Tables, other convenient data structures could be used. For example, Red Black trees and AVL trees would offer an access time proportional to the logarithm of the cardinality of the stored set of states.

4.3. Performance evaluation

Given the implementation of the algorithms as in Section 4.2.2 a performance analysis has been conducted. As a reminder,

N is the number of hidden states of the Markov chain,

T is the number of possible observations,

$h_i = \# \{s \mid P(o_i|s) \neq 0\}$ is the number of states from which we can observe o_i , $i = 1 \dots T$,

*A hash table has average searching time $O(1)$, constant in the number of operations.

$$h = \max_i h_i,$$

k_i is the number of predecessors of state s_i , $i = 1 \dots N$,

$$k = \max_i k_i.$$

The computation involved in the calculation of the *naïve implementation* of the Viterbi Algorithm requires on the order of $N^2 \cdot T$. In fact for each of the T observations we must find the states that give that specific observation and for each state we need to identify all its predecessors. In the case of the *fast implementation*, for each observation, the algorithm has to consider at most h states and for each of those states there are at most k predecessors. Locating one of the k predecessor in the data structure considered here requires $O(h)$ time. The total number of operations is therefore on the order of $h^2 \cdot k \cdot T$. The *super-fast implementation* exploits the faster average searching time of a hash table to reduce the complexity of the algorithm. In particular h states still need to be considered for each observation and again each of those states has at most k predecessor. Locating one such predecessor now requires constant time. So the average complexity of the algorithm is $h \cdot k \cdot T$. So we have shown how the complexity of the algorithm can decrease.

Given the different implementation of the algorithms a performance analysis has been conducted (see Figure 1). For each of the three Viterbi algorithms, the time for completing the hidden state estimation given a sequence of observations has been registered.

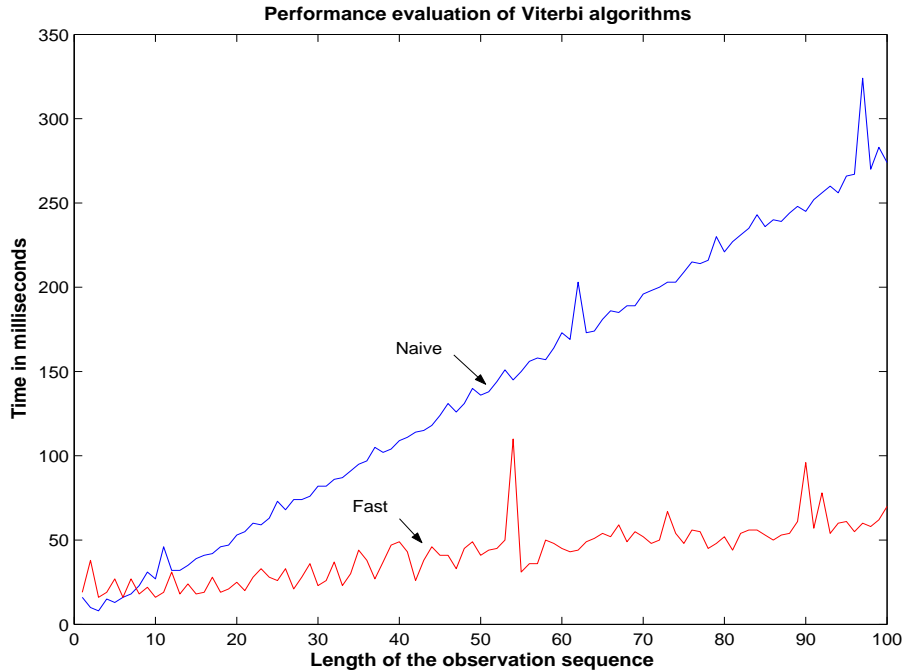


Figure 1. Empirical performance evaluation of two implementations of the Viterbi algorithm.

The graph indicates the time needed to run both the naïve implementation and the fast one for the model that considers a Gaussian probability distribution. It is evident that the naïve execution is significantly slower than the fast one. Although they are both linear in the number of observations, the slope of the two curves is different due to different multiplicative constants. In the naïve case they are quadratic in the number of states N , while in the fast implementation they are proportional to $h^2 \cdot k$, and if the model is sparse $h^2 \cdot k \ll N^2$. This experiment confirms the theoretical performance evaluation analysis. The super-fast implementation was not included since the factor h was only 9, not enough to improve the running time of the algorithm over the fast implementation.

5. CONCLUSION

The problem of estimating state histories from observations is central in many different methods and paradigms employed in the whole area of information technologies. While trying to solve the problem in the general way is too challenging, concentrating on sub-classes of problems can lead to efficient solutions. A wide range of applications can exploit these improvements.

Our approach consists of quantifying the quality of the goodness of different models in order to establish trade-offs between complexity and accuracy. In order to do this we introduced an optimization problem that involves accuracy and complexity functions that have to be kept under thresholds depending on the situation.

6. FUTURE WORK

We intend to investigate the different metrics for computing the distance between sequences of states. An important problem to study is the relationship between Accuracy Function and $H(S|O)$ and how complexity relates to these quantities. Our intuition is that if $H(S|O)$ is large the function $A(\delta)$ is very sensitive. Experiments to confirm this expectation will be performed. The final goal is to quantify the performance of the model through these mathematical functions and not through the model itself. Analysis of the correlation between decreasing the complexity of the algorithm and the structure of the underlying graph will be conducted. Different types of filtering will be explored, cases of vertical and horizontal mapping will be considered.

We also intend to develop a framework for establishing how sensitive a model is with respect to a set of chosen metrics, a characterization of approximability with respect to those metrics, and crucial values of the parameter vector.

ACKNOWLEDGMENTS

This work was partially supported by: ARDA Grant F30602-03-C-0248 and National Institute of Justice, Department of Justice award number 2000-DT-CX-K001. Points of view in this document are those of the authors and do not necessarily represent the official position of the sponsoring agencies or the U.S. Government. Many thanks to Professors George Cybenko and Valentino Crespi for their valuable suggestions on this research.

REFERENCES

1. Y. Ephraim and N. Merhav, "Hidden Markov Processes," *IEEE Transactions on Information Theory* **48**(6), pp. 1518–1569, 2002.
2. G. Forney, "The Viterbi Algorithm," *Proc. of the IEEE*, pp. 268–278, March 1973.
3. L.R. Rabiner and B. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP magazine* **3**, pp. 4–16, 1986.
4. T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. of the IEEE* **77**, pp. 541–580, 1989.
5. H. Alla and R. David, "Continuous Petri Nets," *Proc. 8th Int. Workshop on Appl. Theory of Petri Nets*, pp. 275–294, 1987.
6. J. LeBail, H. Alla, and R. David, "Hybrid Petri Nets," *Proceedings 1st European Control Conference*, pp. 1472–1479, 1991.
7. M. Molloy, "Performance Analysis using Stochastic Petri Nets," *IEEE Trans. Computers* **31**, pp. 913–917, 1982.
8. G. B. M.A. Marsan and G. Conte, "A class of Generalized Stochastic Petri Nets," *ACM trans. Comput. Syst.* **2**, pp. 93–122, May 1984.
9. B. Plateau and J. Fourneau, "A methodology for solving Markov Models of Parallel Systems," *Journal of Parallel and Distributed Computing* **12**, pp. 370–387, 1991.
10. B. Plateau and K. Atif, "Stochastic Automata Network for Modeling Parallel Systems," *IEEE Transactions on software engineering* **17**(10), pp. 1093–1108, October 1991.
11. M. Vernon and M. Holliday, "Performance Analysis of Multiprocessor Cache Consistency Protocols using Generalized Timed Petri Nets," *Proc. Performance 1986 and ACM Sigmetrics 1986*, pp. 9–17, May 1986.

12. A. Benveniste, B. C. Levy, E. Fabre, and P. L. Guernic, "A Calculus of Stochastic Systems for the Specification, Simulation, and Hidden State Estimation of Mixed Stochastic/Non-Stochastic Systems," *Theoretical Computer Science* **152**(2), pp. 171–217, 1995.
13. R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, and A. Benveniste, "A Petri Net approach to Fault Detection and Diagnosis in distributed Systems. Part I : Application to Telecommunication Networks, Motivations and Modeling," *Proc. of the 1997 IEEE Control and Decision Conference (CDC)* , december 1997.
14. A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard, "A Petri Net approach to Fault Detection and Diagnosis in distributed Systems. Part II : extending iterbi Algorithm and HMM Techniques to Petri Nets," *Proc. IEEE CDC* , december 1997.
15. E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith, "Distributed State Reconstruction for Discrete Event Systems," *Proc. of the 2000 IEEE Control and Decision Conference (CDC)* , December 2000.
16. P. Ramadge, "Observability of Discrete-Event Systems," *Proc. 25th Conf. on decision and Control* , pp. 1108–1112, December 1986.
17. C. Ozveren and A. Willsky, "Observability of Discrete Event Dynamic Systems," *IEEE Transactions on Automatic Control* **35**(7), pp. 797–806, July 1990.
18. A. Giua, J. Julvez, and C. Seatzu, "Marking Estimation of Petri Nets based on Partial Observation," *Proceedings of the American Control Conference* , pp. 326–331, June 2003.
19. G. Cybenko *et al.*, "Process Query Systems for Surveillance and Awareness," *Proc. of the Systemics, Cybernetics and Informatics* , 2003.
20. M. Falkhausen, H. Reininger, and D. Wolf, "Calculation of Distance Measures between Hidden Markov Models," *Proc. Eurospeech* , pp. 1487–1490, 1995.
21. T. Minka, "From Hidden Markov Models to Linear Dynamical Systems," *Technical report, MIT.* , 1999.
22. D. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transaction on Automatic Control* **24**(6), pp. 843–854, 1979.
23. L.R.Rabiner, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. of the IEEE* **77**, pp. 257–286, 1989.
24. D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE TIT: IEEE Transactions on Information Theory* **45**(2), pp. 399–431, 1999.