The Most Probable Labeling Problem in HMMs and Its Application to Bioinformatics

Broňa Brejová, Daniel G. Brown*, and Tomáš Vinař**

School of Computer Science, University of Waterloo, Waterloo ON N2L 3G1 Canada
{bbrejova,browndg,tvinar}@uwaterloo.ca

Abstract. Hidden Markov models (HMMs) are often used for biological sequence annotation. Each sequence element is represented by states with the same label. A sequence should be annotated with the labeling of highest probability. Computing this most probable labeling was shown NP-hard by Lyngsø and Pedersen [9]. We improve this result by proving the problem NP-hard for a fixed HMM. High probability labelings are often found by heuristics, such as taking the labeling corresponding to the most probable state path. We introduce an efficient algorithm that computes the most probable labeling for a wide class of HMMs, including models previously used for transmembrane protein topology prediction and coding region detection.

1 Introduction

We present several contributions towards understanding the most probable labeling problem in hidden Markov models (HMMs) and its impact on bioinformatics applications. We prove the problem NP-hard even for a fixed HMM; previous NP-hardness proofs constructed HMM topologies whose size depended on an input instance, which is not appropriate in the context of bioinformatics applications. We also characterize a class of HMMs where the problem can be solved in polynomial time, and finally we demonstrate the usefulness of our findings, using examples from bioinformatics literature and simple experiments.

HMMs are often used in bioinformatics for sequence annotation tasks, such as gene finding (*e.g.*, [1]), protein secondary structure prediction (*e.g.*, [8]), and transmembrane protein topology (*e.g.*, [7]). An HMM is a generative probabilistic model composed of states and transitions. In each step of the generative process, the current state randomly generates one character of the DNA or protein sequence according to an emission probability table associated with the state. An outgoing transition is randomly chosen and followed to a new state, according to the transition probability table at that state.

In each application, biological knowledge is reflected in the model's topology. Parameters are set through automatic training so that the model generates sequences like those in nature. States represent distinct biological features, such

* Supported by NSERC and the Human Frontier Science Program.
** BB and TV supported by NSERC grant RGPIN46506-01 and by CITO.

I. Jonassen and J. Kim (Eds.): WABI 2004, LNBI 3240, pp. 426–437, 2004.

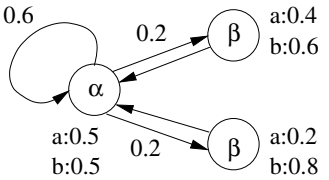

Fig. 1. An HMM with the multiple path problem. The most probable path for the string "ababa" is "$\alpha\alpha\alpha\alpha\alpha$," with probability 0.004, while the most probable labeling is "$\alpha\beta\alpha\beta\alpha$," with probability 0.01. The highest probability path with the same labeling has probability only 0.003.

as introns, exons, and splicing signals in gene finding, or transmembrane helices, cytoplasmic and non-cytoplasmic loops in transmembrane topology prediction.

For sequence annotation, we label each state with the feature it represents. For a given sequence, we examine paths through the model (or *state paths*) generating it. The labelings of such paths represent potential annotations. The HMM defines a probability distribution over all possible labelings: a given labeling's probability is the sum of the probabilities of the paths that generate it. Finding high probability labelings of a sequence in an HMM is *HMM decoding*.

The most widely used decoding method is the *Viterbi algorithm* [11]. It uses dynamic programming to find the most probable state path and reports the labeling associated with it. However, one labeling may have multiple paths, with its probability being the sum of probabilities of these paths. We would prefer to find the *most probable labeling* instead of the *most probable state path.*

Consider the example in Fig.1. The most probable labeling and most probable state path yield different annotations, which is surely a source for concern. We say that such HMM exhibits the *multiple path problem*. Moreover, as the sequence $(ab)^n$ grows in our example, the number of paths forming the most probable labeling increases exponentially. The probability of each single path is very low compared to the probability of the most probable path.

This problem has been recognized before [1, 6] and various heuristics have been suggested. A common (but rarely implemented) idea is to compute the k most probable paths, providing k candidate labelings [3]. These paths can be found efficiently with an algorithm for finding the k shortest paths in a directed acyclic graph [4]. However, this approach may fail, since the probability of each path in the most probable labeling may be small.

A different heuristic, called the 1-best algorithm, was suggested by Krogh [6]. The algorithm, similar to Viterbi, maintains a pool of several candidates for the labeling. The algorithm guarantees only that the probability of the resulting labeling is at least as high as the probability of the most probable state path.

Finally, one can apply *a posteriori* decoding – using the forward-backward algorithm [3] to compute the most probable label at each sequence position. However, no state path may correspond to this annotation, so we cannot guarantee that it is consistent with biological constraints of the model. To complete the heuristic, a second step is required to modify such a labeling to obtain a plausible annotation (see, *e.g.*, [10]).

In general, it is unlikely that there exists an efficient algorithm for finding the most probable labeling: the problem is NP-hard [9] (see also earlier work on a related model [2]). However, in these NP-hardness proofs, the size of the

HMM obtained by the reduction is polynomial in the size of the input instance. This is not appropriate, since in our applications the model is fixed (constant size) and the input sequence can be very long. This leads to the question of whether there is an algorithm for this problem whose runtime is polynomial in the sequence length but exponential in the model size. In Section 2 we show that this is unlikely: the problem is NP-hard even for a *fixed HMM of constant size*.

Ideally, one would like to distinguish between HMMs for which the most probable labeling problem is NP-hard and HMMs for which it is possible to solve the problem in polynomial time. As a first step in this direction, we present an algorithm that can find the most probable labeling for a wide class of HMMs in polynomial time, and we give a sufficient condition characterizing this class. The class includes topologies commonly used for various bioinformatics applications. Finally, we provide simple experiments that show that using the most probable labeling instead of the most probable paths may increase accuracy.

2 NP-Completeness Proof

Here, we give a new NP-completeness proof for the most probable labeling problem. We show a construction of a *specific* HMM, for which if we could compute the most probable labeling, we could solve instances of SAT. We first present a path counting problem on directed acyclic "layered" graphs, which we show is NP-complete. Then, we show a reduction from this problem to the HMM problem, and demonstrate that for SAT instances, we will have a specific HMM of constant, though large, size for which decoding is hard.

Layered Digraphs. A *colored proper layered digraph* is a directed graph with vertices arranged in layers $L_1, L_2, \ldots, L_w$. Each edge connects a vertex in some layer L_i to a vertex in layer L_{i+1}. Each vertex is colored white or black.

A *layer coloring* is an assignment of a color (white or black) to each layer. A directed path from layer L_1 to layer L_w is *consistent with a layer coloring* if the colors of the vertices on the path match the layer coloring.

Definition 1 (Best-Layer-Coloring**).** *Given a colored proper layered digraph G and a threshold T, is there a layer coloring which has at least T paths consistent with it?*

Theorem 1. Best-Layer-Coloring *is NP-complete, even if each layer has at most a constant number of vertices.*

Proof. Best-Layer-Coloring is in NP: for a given layer coloring, the number of consistent paths can be computed by simple dynamic programming.

To prove NP-hardness, we reduce SAT to Best-Layer-Coloring. Consider an instance of SAT with n variables $u_1, u_2, \ldots, u_n$ and m clauses $c_1, c_2, \ldots, c_m$. We give an overview of the construction in Fig.2.

Goal of the Construction. The graph consists of $m+1$ blocks $0, 1, 2, \ldots, m$, each with $4n$ layers. The layer coloring of each block represents a truth assignment of variables $u_1, \ldots, u_n$. The truth assignment of each variable is encoded by four

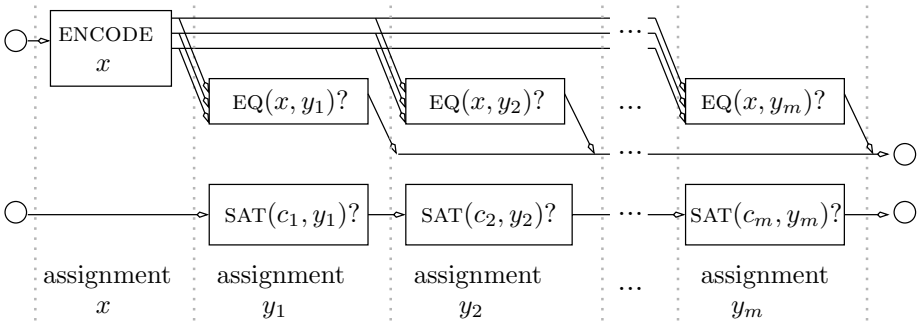

Fig. 2. Overview of the graph construction. The boxes represent components of the graph construction. Lines connecting the components represent layered subgraphs that propagate the number of paths from left to right regardless of the layer coloring (this is achieved by using one black and one white vertex in each layer).

consecutive layer colors: ○ ○ ○ ○ (false) or ○ ○ ● ○ (true). Layer colorings that are not of this form have no corresponding paths, so we do not consider them. Thus we use the terms "layer coloring of a block" and "truth assignment of a block" interchangeably. Let x be the truth assignment of block 0 and let $y_1, \ldots, y_m$ be the truth assignments of blocks $1, \ldots, m$.

In a "yes" instance of SAT, we want the truth assignments $x, y_1, \ldots, y_m$ to be the *same* satisfying truth assignment.

Details of Construction. We will decompose the structure of the graph into several components, each of them having several inputs and outputs. An input of a component is the number of consistent paths ending in a designated vertex on the left-most layer of the component. Similarly, an output of a component is the number of consistent paths ending in a designated vertex on the right-most layer of the component. Let $A \xrightarrow{x} B$ denote a component transforming a vector of inputs A to a vector of outputs B when corresponding layers have coloring x.

The component ENCODE(x) in Fig.2 encodes the truth assignment x as a vector of three integers $v(x)$ on its output. In each of the blocks $1, 2, \ldots, m$, we enforce the truth assignment to be the same as x with component EQ(x, y_i). The input of this component is the vector $v(x)$, and the output is the number $2K(n)$, where $K(n) = 4^n - 2^{n+1} + 1$, if the truth assignments x and y_i are the same, or a number smaller than $2K(n)$ otherwise. In particular, the two components have the following specification: ENCODE$(x) : 1 \xrightarrow{x} (1, K(n) - b(x)^2, 2b(x))$ and EQ$(x, y, i) : (1, \alpha, \beta) \xrightarrow{y} K(n) - b(y)^2 + \beta \cdot b(y) + \alpha$, where $b(x)$ is the number whose binary representation encodes the truth assignment x of variables $u_1, \ldots, u_n$ with u_1 as the highest-order bit and u_n as the lowest-order bit.

Finally, component SAT(c_i, y_i) outputs its input, if truth assignment y_i satisfies clause c_i, or 0 otherwise. The input to SAT(c_1, y_1) is 1. Details are omitted.

There is an additional layer before the first block and after the last block to ensure the proper start and end of each consistent path.

The threshold T is $2m \cdot K(n) + 1$. For the number of consistent paths to reach this threshold, all of the block colorings must represent the same truth

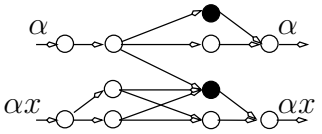

Fig. 3. One section of component $\text{MULT}(x) : \alpha \xrightarrow{x} \alpha b(x)$.

assignment and the assignment must satisfy all clauses (otherwise the number of consistent paths will be smaller).

Components ENCODE *and* EQ. These can be built from subcomponents $\text{MULT}(x) : \alpha \xrightarrow{x} \alpha b(x)$ and $\text{SQUARE}(x) : 1 \xrightarrow{x} K(n) - b(x)^2$. They consist of identical 4-layer sections, each processing one bit of $b(x)$. Section k of component $\text{SQUARE}(x)$ has four inputs and outputs $(1, B(k-1), C(y, k-1), D(y, k-1)) \xrightarrow{t} (1, B(k), C(z, k), D(z, k))$, where y is the binary representation of truth assignment of the first $k-1$ variables, t is the truth assignment of the k-th variable, $z = 2y + t$, and B, C, D are defined as follows:

$$B(k) = 2^{k+2} - 4 = 2B(k-1) + 4 \tag{1}$$

$$C(z,k) = B(k) - 4z = \begin{cases} 2C(y,k-1) + 4, & \text{if } t = 0 \\ 2C(y,k-1), & \text{if } t = 1 \end{cases} \tag{2}$$

$$D(z,k) = 4^k - 2^{k+1} + 1 - z^2 = \begin{cases} 4D(y,k-1) + B(k-1) + 1, & \text{if } t = 0 \\ 4D(y,k-1) + C(y,k-1), & \text{if } t = 1 \end{cases} \tag{3}$$

The D output of the last section has value $D(b(x), n) = K(n) - b(x)^2$, as desired. Fig.3 shows the construction of the simpler subcomponent for multiplication, $\text{MULT}(x)$. Other drawings are omitted due to space.

Summary. The total number of vertices in each layer is at most 29. Thus, an instance of SAT can be reduced to BEST-LAYER-COLORING with a constant number of nodes per layer. □

Connection to HMMs. Now, with the hardness of our problem shown, we connect it to HMMs by reducing it to the most probable labeling problem.

Theorem 2. *For a given constant k and a colored proper layered digraph G with at most k vertices in each layer, there exists an HMM M and a binary string S such that the most probable labeling of string S in M represents the best layer coloring of G. Moreover, the topology, emission, and transition probabilities of M depend only on the constant k, not on the size of G, and the size of S is polynomial in the number of layers of G.*

Proof. (Sketch.) We can construct an HMM and an input string so that paths in the HMM will correspond to paths in G, and all paths have the same probability. The states represent possible configurations of edges in G outgoing from each vertex of one layer (pairs (i, V'), where i is a vertex in a layer, and V' is a subset of vertices in the next layer). Since there is only a constant number of vertices in each layer, this HMM is fixed and does not depend on the structure of G. The structure of G is encoded in the input string: each symbol represents the configuration of edges of one layer in G. The alphabet size is constant and can be further reduced to binary by replacing every symbol by a binary string. □

Corollary 1. *There exists a* specific *HMM for which it is NP-hard to find the most probable labeling of an input binary string.*

Proof. Theorem 1 shows there exists a constant k such that finding the best layer coloring is NP-hard, even if each layer has size at most k. Theorem 2 shows this problem can be reduced to the most probable labeling of a binary string in a fixed HMM. □

The HMM obtained in the proof of Corollary 1 is very large. It is possible to use ideas from the proof of Theorem 1 to reduce SAT to the most probable labeling problem directly, obtaining a much smaller HMM.

3 Computing the Most Probable Labeling

We have shown that in general it is NP-hard to compute the most probable labeling for a given HMM. However, we can characterize special classes of HMMs for which the most probable labeling can be computed efficiently. For example, consider an HMM where any two state paths Π_1 and Π_2 have different labelings. The most probable state path can be computed with the Viterbi algorithm, and thus so can be the most probable labeling. The runtime is $O(nm\Delta)$ time, where n is the length of the sequence, m is the number of states in HMM, and Δ is the maximum in-degree of a state.

Here, we introduce *extended labelings* and give an algorithm that computes the *most probable extended labeling* in polynomial time. We characterize a class of HMMs for which an extended labeling uniquely determines a single state labeling. For this class, our algorithm finds the most probable labeling. Even if the input HMM does not belong to the class, our algorithm returns a labeling with probability at least as high as the probability of the most probable state path. We give several practical examples found in this class of HMMs.

3.1 Most Probable Extended Labeling

Recall that a hidden Markov model is a generative probabilistic model, consisting of *states* and *transitions*. The HMM starts in a designated initial state s. In each step, a symbol is generated according to the emission probabilities of the current state. Unless the HMM has reached a designated final state f, it follows one of the transitions to another state. Let $e_u(x)$ be the emission probability of generating character x in state u, $a_{u,v}$ the probability of a transition from u to v, $\ell(u)$ the label of state u, and $\text{in}(u)$ the set of states having a transition to state u.

Definition 2 (Extended labeling). *A* critical edge *is an edge between states of different label. The* extended labeling *of a state path $\pi_1\pi_2\ldots\pi_n$ is the pair (L, C), where $L = \lambda_1, \lambda_2, \ldots, \lambda_n$ is the sequence of labels of each state in the path and $C = c_1, c_2, \ldots, c_k$ is the sequence of critical edges followed on the path.*

Theorem 3 (The most probable extended labeling). *For a given sequence $S = x_1 \dots x_n$ and an HMM with m states, it is possible to compute the most probable extended labeling in $O(n^2 m L_{\max} \Delta)$, where $L_{\max}$ is the maximum number of states with the same label, and Δ is the largest in-degree of a state.*

Proof. We modify the Viterbi algorithm for computing the most probable state path. Let $V[u,i] = \max \Pr(x_1 \dots x_i, \pi_1 \dots \pi_i)$, where the maximum is taken over all state paths $\pi_1 \dots \pi_i$ starting in state s and ending in state u. The Viterbi algorithm computes the values $V[u,i]$ by dynamic programming with the following recurrence, examining all possible options for the second to last state:

$$V[u,i] = \max_{v \in \mathrm{in}(u)} V[v, i-1] \cdot a_{v,u} \cdot e_u(x_i). \tag{4}$$

We modify the Viterbi algorithm as follows. Let $L[u,i] = \max \Pr(x_1 \dots x_i, (L,C), \pi_i = u)$, where the maximum is taken over all extended labelings (L,C) and the generating process ends in state u. Instead of considering possible options for the second last state, we will examine all possible durations of the last segment with the same label and instead of the most probable path in such segment, we will compute the sum of all possible state paths in this segment. If the segment starts at position $j \leq i$ of the sequence, such sum $P[v,u,j,i]$ is the probability of generating the sequence $x_j \dots x_i$ starting in state v and ending in state u, using only states with label $\lambda(u)$. We get the following recurrence:

$$L[u,i] = \max_{j \leq i} \max_{v:\lambda(v)=\lambda(u)} \max_{\substack{w \in \mathrm{in}(v): \\ \lambda(w) \neq \lambda(v)}} L[w, j-1] \cdot a_{w,v} \cdot P[v,u,j,i] \tag{5}$$

We compute values of L in order of increasing i. For each i, we compute all relevant values of $P(v,u,j,i)$ in order of decreasing j by an algorithm similar to backward algorithm (see, *e.g.*, [3]), using the following recurrence:

$$P[v,u,j,i] = \sum_{w:v \in \mathrm{in}(w), \lambda(v)=\lambda(w)} e_v(x_j) \cdot a_{v,w} \cdot P[w,u,j+1,i] \tag{6}$$

When the computation of L is finished, the most probable extended labeling can be reconstructed by tracing back labels and critical edges used to obtain the value of $L[f,n]$, as in the Viterbi algorithm. □

Note that the probability of the extended labeling returned by the algorithm is always at least as high as the probability of the most probable path Π found by Viterbi algorithm. This is because the probability of the extended labeling corresponding to Π must be at least as high as the probability of Π itself.

3.2 The Sufficient Condition

The algorithm defined above is guaranteed to compute the most probable labeling for a much wider class of HMMs than the Viterbi algorithm. Here is a sufficient condition for this class:

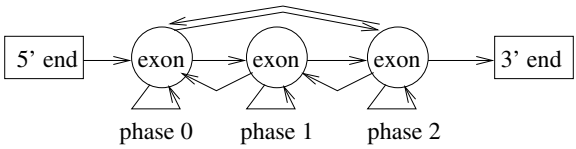

Fig. 4. Simplified model of ESTScan. ESTScan uses an HMM for predicting coding part of an EST. Compared to a typical coding region predictor, ESTScan needs to handle insertions and deletions within the coding sequence caused by the low quality of EST sequencing. The exact place of a sequencing error cannot be easily identified, so to simply distinguish coding part of ESTs from non-coding, we assign the same label to all states corresponding to coding sequence. The resulting HMM has the multiple path problem, with each path corresponding to some combination of insertions and deletions. The actual model used in ESTScan has a more complicated topology, ensuring for example that only one insertion or deletion can occur within the same codon.

Definition 3. *An HMM* satisfies the critical edge condition *for an input sequence S if any two paths with the same labeling have the same sequence of critical edges.*

Corollary 2. *If an HMM satisfies the critical edge condition for a sequence S, then the above algorithm computes the most probable labeling of sequence S.*

Proof. We call a labeling Λ (extended labeling, state path) *possible* with respect to sequence S, if $\Pr(\Lambda \mid S) > 0$.

The algorithm above computes the most probable extended labeling. Therefore for the statement to be false, the most probable labeling and the most probable extended labeling must be different.

This happens only if two different extended labelings Λ_1 and Λ_2 correspond to the most probable labeling. Let Π_1 be a state path corresponding to Λ_1, and Π_2 be a state path corresponding to Λ_2. Since Λ_1 and Λ_2 are different, the paths Π_1 and Π_2 must differ in at least one critical edge; yet both Π_1 and Π_2 produce the same labeling. Therefore the HMM cannot satisfy the critical edge condition. □

Examples. The simplified model of ESTScan [5] shown in Fig.4 has the multiple path problem. Therefore the Viterbi algorithm is not appropriate for decoding it. The model satisfies the critical edge condition, so our algorithm finds the most probable labeling. The condition is satisfied because the states labeled "exon" are grouped in a subgraph with only one incoming and outgoing edge.

A more complicated example is the simple model of exon/intron structure of eukaryotic genes in Fig.5. Multiple copies of the same intron model preserve three-periodicity of coding regions. The multiple path problem is caused by ambiguity of transition between the two "intron" states. The model does not violate our condition, since the length of the exonic sequence uniquely determines which critical edge will be used.

Testing the Critical Edge Condition. We can test algorithmically whether a given HMM topology (not considering emission probabilities) satisfies the critical edge

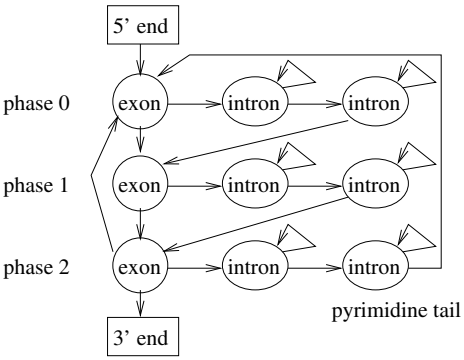

Fig. 5. Simple model of exon/intron structure. Intronic sequence in DNA contains a pyrimidine-rich tail close to the acceptor site. Its composition is very different from the rest of the intron, providing strong support for a neighboring acceptor site. The tail has variable length and does not have a clear boundary. This creates the multiple path problem because there are always several high-probability alternatives for the transfer from "intron" state to the "tail" state.

condition for every input sequence. We first use depth-first search to build a set S_s of all pairs of states that are reachable from the start state by the same labeling. We start from pair $(s, s) \in S_s$ and in each iteration we add a new pair (u, v) if $\lambda(u) = \lambda(v)$, and there exists $(u', v') \in S_s$ such that $u' \in \text{in}(u)$ and $v' \in \text{in}(v)$. Similarly, we also build a set S_f of all pairs of states, from which the final state can be reached by the same labeling. For the critical condition to be violated, there must exist a pair $(u, v) \in S_s$ and $(u', v') \in S_f$ such that $\lambda(u) \neq \lambda(u')$, and (u, u') and (v, v') are two different transitions. The algorithm takes $O(m^4)$ time.

It is possible to modify this verification algorithm to verify the critical edge condition in $O(m^4|\Sigma|^2)$ time, if emission probabilities are given. Note that this test may yield a different result, since some states may not produce some of the alphabet symbols, making it impossible for two different paths with the same extended labeling to generate the same string; hence, this extended algorithm may find even more HMMs that satisfy the condition.

And finally, we can also verify the condition for a given HMM and input string in $O(nm^4)$ time. In that case, we will build a set of state pairs that can be reached by the same labeling for each position in the sequence.

3.3 Introducing Silent States

Silent states do not emit symbols. They are sometimes used in HMMs as a convenient modeling extension (see [3, Section 3.4]). Both the Viterbi algorithm and our algorithm can be easily extended to HMMs with silent states [3, Section 3.4]. The example in Fig.6 shows that some HMMs can be transformed to equivalent HMMs that satisfy the critical edge condition by addition of silent states. Thus, in our case, the silent states are a crucial modeling tool.

Example. Silent states are useful when one wants to provide several models for a particular sequence element. An example of such a model is TMHMM [7], shown in Fig.7. The two different models of non-cytoplasmic loops create the multiple path problem, potentially decreasing prediction accuracy if the Viterbi algorithm is used. We introduce silent states to ensure that the critical edge condition is satisfied.

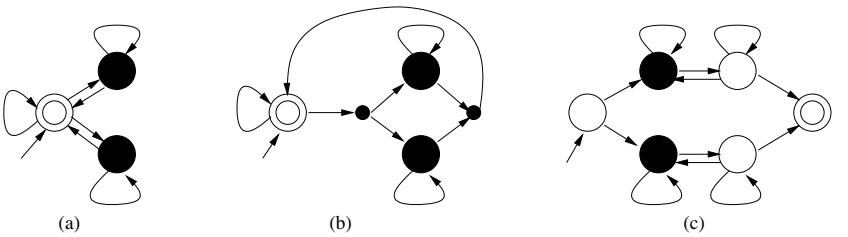

Fig. 6. Example of usefulness of silent states. The color of each state represents its label. Silent states are shown as smaller circles. The HMM (a) violates the critical edge condition and cannot be decoded by our algorithm. There is no equivalent topology without silent states satisfying the condition. Using silent states, we can construct an equivalent HMM (b) that satisfies the critical edge condition. However, the technique is not universal: HMM (c) cannot be so transformed to comply with the condition.

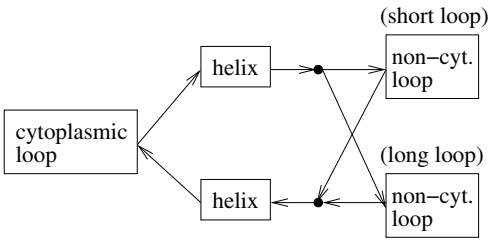

Fig. 7. TMHMM: prediction of topology of transmembrane proteins. The task is to predict positions of transmembrane helices, cytoplasmic, and non-cytoplasmic loops. Boxes in the figure represent groups of states with the same label.

4 Experiments

We have designed a simple experiment to test if decoding with most probable labeling increases accuracy. We used the HMM in Fig.8 to generate 5000 sequences of mean length about 500 for various combinations of the parameters p_1 and p_2.

We then used three decoding algorithms: standard Viterbi, our algorithm for most probable labeling, and Viterbi on a simplified model. In the simplified model, we replaced the two "gray" states with self-loops with one state and set its parameters to maximize likelihood (probability of the self-loop: 0.97, probability of emission of 1: $(2p_2 + p_1)/3$). This new HMM does not have the multiple path problem and therefore the Viterbi algorithm yields the most probable labeling.

We evaluated the error rate (percentage of the positions that were mislabeled compared to the labels on the state path that generated each sequence) for each algorithm. Fig.9 shows our results.

We have observed two trends in the data. First, using the most probable labeling does increases the accuracy. Second, the Viterbi algorithm on a simplified model, which does not have the multiple path problem, often performs better than the Viterbi algorithm on the full model. This behavior is paradoxical: using a model that is further from reality, we have obtained better results.

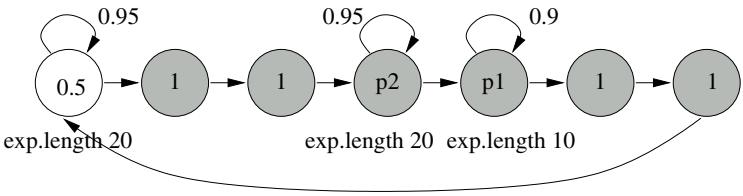

Fig. 8. HMM used in experiments. The HMM over alphabet $\{0, 1\}$ is inspired by intron model with composition changing towards the end of "gray" region. Colors (white or gray) represent the labels. The numbers inside states represent emission probability of symbol 1 (p_1 and p_2 are parameters).

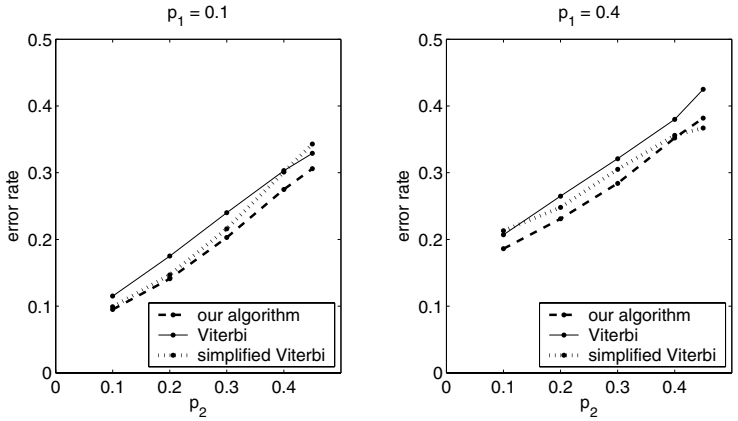

Fig. 9. Results of experiments. Error rate using three decoding algorithms: Viterbi, our algorithm for the most probable labeling, and Viterbi in simplified model.

5 Conclusions and Future Work

We have investigated the most probable labeling problem in HMMs. We showed that the problem is NP-hard, even for a fixed HMM constructed in the proof, in contrast to previous NP-hardness proofs, where the HMM constructed depended on the input instance (in most biological applications, the HMM is fixed).

Even though the problem is NP-hard in general, it is possible to compute the most probable labeling for some HMMs. We provided an $O(n^2)$ time decoding algorithm and characterized a wide class of HMMs that can be decoded by our algorithm. This run time may cause problems in applications with long input sequences, such as gene finding. Still, it is acceptable in other cases, such as analysis of protein sequences or ESTs. In practice, the running time can be further decreased by application of biological constraints (such as location of open reading frames) and various stopping conditions.

The model topologies that can be decoded by our algorithm include those for transmembrane protein topology prediction (TMHMM), distinguishing coding regions in ESTs (ESTScan), or intron model in gene finding. We also noted that

the use of the Viterbi algorithm instead of the most probable labeling may lead to paradoxical behavior, where a more accurate model will yield worse results.

Several problems remain open. First, we do not know at present any polynomial algorithm for finding the most probable labeling for model shown in Fig.6c. Is decoding of this simple model NP-hard? Similar topologies are useful in various applications for providing alternative models for multi-label structures (such as different types of genes). More generally, can we provide a complete characterization of the models that are NP-hard to decode? Second, are there HMM topologies (other than ones without the multiple path problem) that can be decoded in subquadratic time? Such models may be useful in applications where the input sequence is long. Finally, we would like to test the most probable labeling algorithm in the applications mentioned above.

Acknowledgments

The authors would like to thank Ming Li and Prabhakar Ragde for useful suggestions.

References

1. C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94, 1997.
2. F. Casacuberta and C. de la Higuera. Computational Complexity of Problems on Probabilistic Grammars and Transducers. In *Grammatical Inference: Algorithms and Applications (ICGI)*, volume 1891 of *LNCS*, pages 15–24. Springer, 2000.
3. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.
4. D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.
5. C. Iseli, C. V. Jongeneel, and P. Bucher. ESTScan: a program for detecting, evaluating, and reconstructing potential coding regions in EST sequences. In *Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 138–148, 1999.
6. A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. In *Fifth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 179–186. AAAI Press, 1997.
7. A. Krogh, B. Larsson, G. von Heijne, and E. L. Sonnhammer. Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *Journal of Molecular Biology*, 305(3):567–570, 2001.
8. P. Lio, N. Goldman, J. L. Thorne, and D. T. Jones. PASSML: combining evolutionary inference and protein secondary structure prediction. *Bioinformatics*, 14(8):726–733, 1998.
9. R. B. Lyngsø and C. N. S. Pedersen. The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computer and System Sciences*, 65(3):545–569, 2002.
10. P. L. Martelli, P. Fariselli, A. Krogh, and R. Casadio. A sequence-profile-based HMM for predicting and discriminating beta barrel membrane proteins. *Bioinformatics*, 18(Suppl 1):S46–53, 2002.
11. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.