Improving Viterbi is Hard: Better Runtimes Imply Faster Clique Algorithms

Arturs Backurs [1] **Christos Tzamos** [1]

Abstract

The classic algorithm of Viterbi computes the most likely path in a Hidden Markov Model (HMM) that results in a given sequence of observations. It runs in time $O(Tn^2)$ given a sequence of T observations from a HMM with n states. Despite significant interest in the problem and prolonged effort by different communities, no known algorithm achieves more than a polylogarithmic speedup. In this paper, we explain this difficulty by providing matching conditional lower bounds. Our lower bounds are based on assumptions that the best known algorithms for the All-Pairs Shortest Paths problem (APSP) and for the Max-Weight k-Clique problem in edge-weighted graphs are essentially tight. Finally, using a recent algorithm by Green Larsen and Williams for online Boolean matrix-vector multiplication, we get a $2^{\Omega(\sqrt{\log n})}$ speedup for the Viterbi algorithm when there are few distinct transition probabilities in the HMM.

1. Introduction

A Hidden Markov Model (HMM) is a simple model that describes a random process for generating a sequence of observations. A random walk is performed on an underlying graph (Markov Chain) and, at each step, an observation is drawn from a probability distribution that depends only on the current state (the node in the graph).

HMMs are a fundamental statistical tool and one of the most important questions in the applications of HMMs is computing the most likely sequence of states visited by the random walk in the HMM given the sequence of observations. Andrew Viterbi proposed an algorithm (Viterbi, 1967) for this problem that computes the solution in $O(Tn^2)$ time for any HMM with n states and an observation sequence of length T. This algorithm is known as the *Viterbi algorithm* and the problem of computing the most likely sequence of states is also known as the *Viterbi Path* problem.

The Viterbi algorithm has found wide applicability in machine learning. It is an important tool for structured prediction, used e.g., for structured perceptrons (Collins, 2002). Other applications include speech recognition (Rabiner, 1989; Nefian et al., 2002; Bengio, 2003), part-of-speech tagging (Collins, 2002), action planning (Attias, 2003), emotion recognition (Cohen et al., 2000), human activity classification (Mannini & Sabatini, 2010), and waveform classification (Kim & Smyth, 2006). Furthermore, it is often combined with other methods. For example, a combination of the Viterbi algorithm and neural networks is used for speech recognition (Mohamed et al., 2012; Abdel-Hamid et al., 2012; Bourlard & Morgan, 2012), handwriting recognition and protein secondary structure prediction (Lin et al., 2005; Peng et al., 2009). It also can be combined with Support Vector Machines (Altun et al., 2003). Finally, the Viterbi algorithm is used as a module in Graph Transformer Networks, with applications to speech recognition (LeCun et al., 1998; Collobert, 2011).

The quadratic dependence of the algorithm's runtime on the number of states is a long-standing bottleneck that limits its applicability to problems with large state spaces, particularly when the number of observations is large. A lot of effort has been put into improving the Viterbi algorithm to lower either the time or space complexity. Many works achieve speedups by requiring structure in the input, either explicitly by considering restricted classes of HMMs (Felzenszwalb et al., 2004; Siddiqi & Moore, 2005) or implicitly by using heuristics that improve runtime in certain cases (Esposito & Radicioni, 2009; Kaji et al., 2010). For the general case, in (Lifshits et al., 2009; Mahmud & Schliep, 2011) it is shown how to speed up the Viterbi algorithm by $O(\log n)$ when the number of distinct observations is constant using the Four Russians method or similar ideas. More recently, in (Cairo et al., 2016), the same logarithmic speed-up was shown to be possible for the general case. Despite significant effort, only log-

Authors ordered alphabetically. [1]MIT, US. Correspondence to: Arturs Backurs <backurs@mit.edu>, Christos Tzamos <tzamos@mit.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

Lower Bound	Complexity	Alphabet size	Assumption
Theorem 1	Tn^2	T	APSP Conjecture
Theorem 2	Tn^2	n^ε (for any constant $\varepsilon > 0$)	k-Clique Conjecture
Theorem 4	Tn^2 when $T \leq n$	1 (unary)	APSP Conjecture
Upper Bound	Complexity	Alphabet size	Assumption
Theorem 3	$Tn^2/2^{\Omega(\sqrt{\log n})}$	Any	$2^{\delta\sqrt{\log n}}$ distinct transition probabilities in HMM

Table 1: Summary of our upper and lower bounds for the Viterbi Path problem. n is the number of states in the underlying HMM and T is the number of observations. For a sparse HMM with m non-zero transition probabilities, we show a tight lower bound of Tm (see Theorems 6, 7 and 8 in Section 6).

arithmic improvements are known other than in very special cases. In contrast, the memory complexity can be reduced to almost linear in the number of states without significant overhead in the runtime (Grice et al., 1997; Tarnas & Hughey, 1998; Churbanov & Winters-Hilt, 2008).

In this work, we attempt to explain this apparent barrier for faster runtimes by giving evidence of the inherent hardness of the Viterbi Path problem. In particular, we show that getting a polynomial speedup[1] would imply a breakthrough for fundamental graph problems. Our lower bounds are based on standard hardness assumptions for the All-Pairs Shortest Paths and the Min-Weight k-Clique problems and apply even in cases where the number of distinct observations is small.

Before formally stating our results, let us give some background on the Min-Weight k-Clique problem. This fundamental graph problem asks to find the minimum weight k-clique in the given undirected weighted graph on n nodes and $O(n^2)$ weighted edges. This is the parameterized version of the NP-hard Min-Weight Clique problem (Karp, 1972). The Min-Weight k-Clique is amongst the most well-studied problems in theoretical computer science, and it is the canonical intractable problem in parameterized complexity.

A naive algorithm solves the Min-Weight k-Clique in $O(n^k)$ time and the best known algorithm still runs in $O(n^{k-o(1)})$ for any constant k. Obtaining a significantly faster algorithm for this problem is a longstanding open question.

A conjecture in graph algorithms and parameterized complexity is that it there is no $O(n^{k-\varepsilon})$ algorithm for any constant $\varepsilon > 0$. The special case of the conjecture with $k = 3$ says that finding the minimum weight triangle in a weighted graph cannot be solved in $O(n^{3-\delta})$ time for any constant $\delta > 0$. There are many negative results that intuitively support this conjecture: a truly subcubic algorithm for Min-Weight 3-Clique implies such algorithm for the All-Pairs Shortest Paths as well (Williams & Williams, 2010). The latter is a well studied problem and no truly subcubic algorithm is known for it despite significant effort (Williams, 2014). Unconditional lower bounds for k-Clique are known for various computational models, such as $\Omega(n^k)$ for monotone circuits (Alon & Boppana, 1987). The planted Clique problem has also proven to be very challenging (e.g. (Alon et al., 2007; 1998; Hazan & Krauthgamer, 2011; Jerrum, 1992)). Max-Clique is also known to be hard to efficiently approximate within nontrivial factors (Håstad, 1999).

We complement our lower bounds with an algorithm for Viterbi Path that achieves speedup $2^{\Omega(\sqrt{\log n})}$ when there are few distinct transition probabilities in the underlying HMM. We summarize our results in Table 1.

Our results and techniques Our first lower bound shows that the Viterbi Path problem cannot be computed in time $O(Tn^2)^{1-\varepsilon}$ for a constant $\varepsilon > 0$ unless the APSP conjecture is false. The APSP conjecture states that there is no algorithm for the All-Pairs Shortest Paths problem that runs in truly subcubic[2] time in the number of vertices of the graph. We obtain the following theorem:

Theorem 1. *The* VITERBI PATH *problem requires* $\Omega(Tn^2)^{1-o(1)}$ *time assuming the APSP Conjecture.*

The proof of the theorem gives a reduction from All-Pairs Shortest Paths to the Viterbi Path problem. This is done by encoding the weights of the graph of the APSP instance as transition probabilities of the HMM or as probabilities of seeing observations from different states. The proof requires a large alphabet size, i.e. a large number of distinct observations, which can be as large as the number of total steps T.

A natural question question to ask is whether there is a faster algorithm that solves the Viterbi Path problem when

[1] Getting an algorithm running in time, say $O(Tn^{1.99})$.

[2] Truly subcubic means $O(n^{3-\delta})$ for constant $\delta > 0$.

the alphabet size is much smaller than T, say when $T = n^2$ and the alphabet size is n. We observe that in such a case, the input size to the Viterbi Path problem is only $O(n^2)$: we only need to specify the transition probabilities of the HMM, the probabilities of each observation in each state and the sequence of observations. The Viterbi algorithm in this setting runs in $\Theta(Tn^2) = \Theta(n^4)$ time. Showing a matching APSP based lower bound seems difficult because the runtime in this setting is quadratic in the input size while the APSP conjecture gives only $N^{1.5}$ hardness for input size N. To our best knowledge, all existing reduction techniques based on the APSP conjecture do not achieve such an amplification of hardness. In order to get a lower bound for smaller alphabet sizes, we need to use a different hardness assumption.

For this purpose, we consider the k-Clique conjecture. It is a popular hardness assumption which states that it is not possible to compute a minimum weight k-clique on an edge-weighted graph with n vertices in time $O(n^{k-\varepsilon})$ for constant k and $\varepsilon > 0$. With this assumption, we are able to extend Theorem 1 and get the following lower bound for the Viterbi Path problem on very small alphabets:

Theorem 2. *For any $C, \varepsilon > 0$, the* VITERBI PATH *problem on $T = \Theta(n^C)$ observations from an alphabet of size $\Theta(n^\varepsilon)$ requires $\Omega(Tn^2)^{1-o(1)}$ time assuming the k-Clique Conjecture for $k = \lceil \frac{C}{\varepsilon} \rceil + 2$.*

To show the theorem, we perform a reduction from the Min-Weight k-Clique problem. Given a Min-Weight k-Clique instance, we create an HMM with two special nodes, a start node and an end node, and enforce the following behavior of the optimal Viterbi path: Most of the time it stays in the start or end node, except for a small number of steps, during which it traverses the rest of the graph to move from the start to the end node. The time at which the traversal happens corresponds to a clique in the original graph of the Min-Weight k-Clique instance. We penalize the traversal according to the weight of the corresponding k-clique and thus the optimal path will find the minimum weight k-clique. Transition probabilities of the HMM and probabilities of seeing observations from different states encode edge-weights of the Min-Weight k-Clique instance. Further, we encode the weights of smaller cliques into the sequence of observations according to the binary expansion of the weights.

Our results of Theorems 1 and 2 imply that the Viterbi algorithm is essentially optimal even for small alphabets. We also study the extreme case of the Viterbi Path problem with unary alphabet where the only information available is the total number of steps T. We show a surprising behavior: when $T \leq n$ the Viterbi algorithm is essentially optimal, while there is a simple much faster algorithm when $T > n$. See Section 5 for more details.

We complement our lower bounds with an algorithm for Viterbi Path that achieves speedup $2^{\Omega(\sqrt{\log n})}$ when there are few distinct transition probabilities in the underlying HMM. Such a restriction is mild in applications where one can round the transition probabilities to a small number of distinct values.

Theorem 3. *When there are fewer than $2^{\varepsilon\sqrt{\log n}}$ distinct transition probabilities for a constant $\varepsilon > 0$, there is a $Tn^2/2^{\Omega(\sqrt{\log n})}$ randomized algorithm for the* VITERBI PATH *problem that succeeds whp.*

We achieve this result by developing an algorithm for online $(\min, +)$ matrix-vector multiplication for the case when the matrix has few distinct values. Our algorithm is presented in Section 7 and is based on a recent result for online Boolean matrix-vector multiplication by Green Larsen and Williams (Larsen & Williams, 2017).

The results we presented above hold for dense HMMs. For sparse HMMs that have at most m edges out of the n^2 possible ones, i.e. the transition matrix has at most m nonzero probabilities, the VITERBI PATH problem can be easily solved in $O(Tm)$ time. The lower bounds that we presented above can be adapted directly for this case to show that no faster algorithm exists that runs in time $O(Tm)^{1-\varepsilon}$. See the corresponding discussion in Section 6.

2. Preliminaries

Notation For an integer m, we denote the set $\{1, 2, \ldots, m\}$ by $[m]$.

Definition 1 (Hidden Markov Model)**.** *A* Hidden Markov Model *(HMM) consists of a directed graph with n distinct hidden states $[n]$ with transition probabilities $\tilde{A}(u, v)$ of going from state u to state v. In any given state, there is a probability distribution of symbols that can be observed and $\tilde{B}(u, s)$ gives the probability of seeing symbol s on state u. The symbols come from an alphabet $[\sigma]$ of size σ. An HMM can thus be represented by a tuple $(\tilde{A}, \tilde{B})$.*

2.1. The Viterbi Path Problem

Given an HMM and a sequence of T observations, the Viterbi algorithm (Viterbi, 1967) outputs a sequence of T states that is most likely given the T observations. More precisely, let $S = (s_1, \ldots, s_T)$ be the given sequence of T observations where symbol $s_t \in [\sigma]$ is observed at time $t = 1, \ldots, T$. Let $u_t \in [n]$ be the state of the HMM at time $t = 1, \ldots, T$. The Viterbi algorithm finds a state sequence $U = (u_0, u_1, \ldots, u_T)$ starting at $u_0 = 1$ that maximizes $\Pr[U|S]$. The problem of finding the sequence U is known as the *Viterbi Path* problem. In particular, the Viterbi Path

problem solves the optimization problem

$$\arg\max_{u_0=1,u_1,\ldots,u_T} \prod_{t=1}^{T} \left[\tilde{A}(u_{t-1},u_t)\cdot\tilde{B}(u_t,s_t)\right].$$

The Viterbi algorithm solves this problem in $O(Tn^2)$ by computing for $t = 1\ldots T$ the best sequence of length t that ends in a given state in a dynamic programming fashion. When run in a word RAM model with $O(\log n)$ bit words, this algorithm is numerically unstable because even representing the probability of reaching a state requires linear number of bits. Therefore, log probabilities are used for numerical stability since that allows to avoid underflows (Young et al., 1997; Amengual & Vidal, 1998; Li & Tang, 2009; Lee et al., 2007; Huang et al., 2001). To maintain numerical stability and understand the underlying combinatorial structure of the problem, we assume that the input is given in the form of log-probabilities, i.e. the input to the problem is $A(u,v) = -\log\tilde{A}(u,v)$ and $B(u,s) = -\log\tilde{B}(u,s)$ and focus our attention on the Viterbi Path problem defined by matrices A and B.

Definition 2 (Viterbi Path Problem)**.** *The* VITERBI PATH *problem is specified by a tuple (A,B,S) where A and B are $n\times n$ and $n\times\sigma$ matrices, respectively, and $S = (s_1,\ldots,s_T)$ is a sequence of $T = n^{\Theta(1)}$ observations $s_1,\ldots,s_T\in[\sigma]$ over an alphabet of size σ. Given an instance (A,B,S) of the* VITERBI PATH *problem, our goal is to output a sequence of vertices $u_0,u_1,\ldots,u_T\in[n]$ with $u_0 = 1$ that solves*

$$\arg\min_{u_0=1,u_1,\ldots,u_T} \sum_{t=1}^{T} \left[A(u_{t-1},u_t) + B(u_t,s_t)\right].$$

We can assume that log probabilities in matrices A and B are arbitrary positive numbers without the restriction that the corresponding probabilities must sum to 1. See Appendix C for a discussion.

A simpler special case of the VITERBI PATH problem asks to compute the most likely path of length T without any observations.

Definition 3 (Shortest Walk Problem)**.** *Given an integer T and a weighted directed graph (with possible self-loops) on n vertices with edge weights specified by a matrix A, the* SHORTEST WALK *problem asks to compute a sequence of vertices $u_0 = 1,u_1,\ldots,u_T\in[n]$ that solves*

$$\arg\min_{u_0=1,u_1,\ldots,u_T} \sum_{t=1}^{T} A(u_{t-1},u_t).$$

It is easy to see that the SHORTEST WALK problem corresponds to the VITERBI PATH problem when $\sigma = 1$ and $B(u,1) = 0$ for all $u\in[n]$.

2.2. Hardness assumptions

We use the hardness assumptions of the following problems.

Definition 4 (ALL-PAIRS SHORTEST PATHS (APSP) problem)**.** *Given an undirected graph $G = (V,E)$ with n vertices and positive integer weights on the edges, find the shortest path between u and v for every $u,v\in V$.*

The *APSP conjecture* states that the ALL-PAIRS SHORTEST PATHS problem requires $\Omega(n^3)^{1-o(1)}$ time in expectation.

Conjecture 1 (APSP conjecture)**.** *The* ALL-PAIRS SHORTEST PATHS *problem on a graph with n vertices and positive integer edge-weights bounded by $n^{O(1)}$ requires $\Omega(n^3)^{1-o(1)}$ time in expectation.*

There is a long list of works showing conditional hardness for various problems based on the All-Pairs Shortest Paths conjecture (Roditty & Zwick, 2004; Williams & Williams, 2010; Abboud & Williams, 2014; Abboud et al., 2015b;c).

Definition 5 (MIN-WEIGHT k-CLIQUE problem)**.** *Given a complete graph $G = (V,E)$ with n vertices and positive integer edge-weights, output the minimum total edge-weight of a k-clique in the graph.*

This is a very well studied computational problem and despite serious efforts, the best known algorithm for this problem still runs in time $O(n^{k-o(1)})$, which matches the runtime of the trivial algorithm up to subpolynomial factors. The *k-Clique conjecture* states that this problem requires $\Omega(n^k)^{1-o(1)}$ time and it has served as a basis for showing conditional hardness results for several problems on sequences (Abboud et al., 2015a; 2014; Bringmann et al., 2016) and computational geometry (Backurs et al., 2016).

Conjecture 2 (k-Clique conjecture)**.** *The* MIN-WEIGHT k-CLIQUE *problem on a graph with n vertices and positive integer edge-weights bounded by $n^{O(k)}$ requires $\Omega(n^k)^{1-o(1)}$ time in expectation.*

For $k = 3$, the MIN-WEIGHT 3-CLIQUE problem asks to find the minimum weight triangle in a graph. This problem is also known as the MINIMUM TRIANGLE problem and under the 3-Clique conjecture it requires $\Omega(n^3)^{1-o(1)}$ time. The latter conjecture is equivalent to the APSP conjecture (Williams & Williams, 2010).

We often use the following variant of the MIN-WEIGHT k-CLIQUE problem:

Definition 6 (MIN-WEIGHT k-CLIQUE problem for k-partite graphs)**.** *Given a complete k-partite graph $G = (V_1\cup\ldots\cup V_k,\ E)$ with $|V_i| = n_i$ and positive integer weights on the edges, output the minimum total edge-weight of a k-clique in the graph.*

If for all i, j we have that $n_i = n_j^{\Theta(1)}$, it can be shown that the MIN-WEIGHT k-CLIQUE problem for k-partite graphs requires $\Omega\left(\prod_{i=1}^{k} n_i\right)^{1-o(1)}$ time assuming the k-Clique conjecture. We provide a simple proof of this statement in the appendix.

3. Hardness of VITERBI PATH

We begin by presenting our main hardness result for the VITERBI PATH problem.

Theorem 1. *The* VITERBI PATH *problem requires* $\Omega(Tn^2)^{1-o(1)}$ *time assuming the APSP Conjecture.*

To show APSP hardness, we will perform a reduction from the MINIMUM TRIANGLE problem (described in Section 2.2) to the VITERBI PATH problem. In the instance of the MINIMUM TRIANGLE problem, we are given a 3-partite graph $G = (V_1 \cup V_2 \cup U,\ E)$ such that $|V_1| = |V_2| = n, |U| = m$. We want to find a triangle of minimum weight in the graph G. To perform the reduction, we define a weighted directed graph $G' = (\{1, 2\} \cup V_1 \cup V_2,\ E')$. E' contains all the edges of G between V_1 and V_2, directed from V_1 towards V_2, edges from 1 towards all nodes of V_1 of weight 0 and edges from all nodes of V_2 towards 2 of weight 0. We also add a self-loops at nodes 1 and 2 of weight 0.

We create an instance of the VITERBI PATH problem (A, B, S) as described below. Figure 1 illustrates the construction of the instance.

- Matrix A is the weighted adjacency matrix of G' that takes value $+\infty$ (or a sufficiently large integer) for non-existent edges and non-existent self-loops.
- The alphabet of the HMM is $U \cup \{\perp, \perp_F\}$ and thus matrix B has $2n + 2$ rows and $\sigma = m + 2$ columns. For all $v \in V_1 \cup V_2$ and $u \in U$, $B(v, u)$ is equal to the weight of the edge (v, u) in graph G. Moreover, for all $v \in V_1 \cup V_2$, $B(v, \perp) = +\infty$ (or a sufficiently large number) and for all $v \in V_1 \cup V_2 \cup \{1\}$, $B(v, \perp_F) = +\infty$. Finally, all remaining entries corresponding to nodes 1 and 2 are 0.
- Sequence S of length $T = 3m + 1$ is generated by appending the observations u, u and $\perp$ for all $u \in U$ and adding a $\perp_F$ observation at the end.

Given the above construction, the theorem statement follows directly from the following claim.

Claim 1. *The weight of the solution to the* VITERBI PATH *instance is equal to the weight of the minimum triangle in the graph G.*

Proof. The optimal path for the VITERBI PATH instance begins at node 1. It must end in node 2 since otherwise when observation $\perp_F$ arrives we collect cost $+\infty$. Similarly, whenever an observation $\perp$ arrives the path must be either on node 1 or 2. Thus, the path first loops in node 1 and then goes from node 1 to node 2 during three consecutive observations u, u and $\perp$ for some $u \in U$ and stays in node 2 until the end. Let $v_1 \in V_1$ and $v_2 \in V_2$ be the two nodes visited when moving from node 1 to node 2. The only two steps of non-zero cost are:

1. Moving from node 1 to node v_1 at the first observation u. This costs $A(1, v_1) + B(v_1, u) = B(v_1, u)$.
2. Moving from node v_1 to node v_2 at the second observation u. This costs $A(v_1, v_2) + B(v_2, u)$.

Thus, the overall cost of the path is equal to $B(v_1, u) + A(v_1, v_2) + B(v_2, u)$, which is equal to the weight of the triangle (v_1, v_2, u) in G. Minimizing the cost of the path in this instance is therefore the same as finding the minimum weight triangle in G. □

4. Hardness of VITERBI PATH with small alphabet

The proof of Theorem 1 requires a large alphabet size, which can be as large as the number of total steps T. In the appendix, we show how to get a lower bound for the VITERBI PATH problem on alphabets of small size by using a different hardness assumption.

Theorem 2. *For any $C, \varepsilon > 0$, the* VITERBI PATH *problem on $T = \Theta(n^C)$ observations from an alphabet of size $\Theta(n^\varepsilon)$ requires $\Omega(Tn^2)^{1-o(1)}$ time assuming the k-Clique Conjecture for $k = \lceil \frac{C}{\varepsilon} \rceil + 2$.*

5. Complexity of VITERBI PATH for unary alphabet

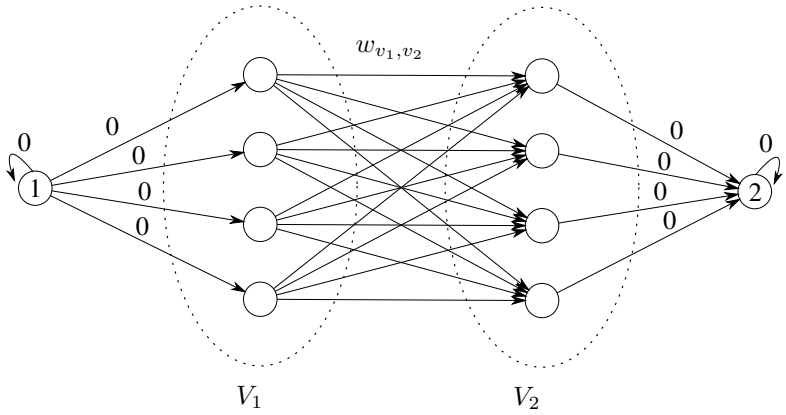
In this section, we focus on the extreme case of VITERBI PATH with unary alphabet.

Theorem 4. *The* VITERBI PATH *problem requires $\Omega(Tn^2)^{1-o(1)}$ time when $T \leq n$ even if the size of the alphabet is $\sigma = 1$, assuming the APSP Conjecture.*

The above theorem follows from APSP-hardness of the SHORTEST WALK problem that we present next.

Theorem 5. *The* SHORTEST WALK *problem requires $\Omega(Tn^2)^{1-o(1)}$ time when $T \leq n$, assuming the APSP Conjecture.*

Proof. We will perform a reduction from the MINIMUM TRIANGLE problem to the VITERBI PATH



(a) The graph specified by transition matrix A. Every edge (v_1, v_2) in $V_1 \times V_2$ has the original edge-weight as in graph G.

Node	$u \in U$	$\perp$	$\perp_F$
1	0	0	∞
$v \in V_1 \cup V_2$	$w_{v,u}$	∞	∞
2	0	0	0

(b) The cost of seeing a symbol at every node given by matrix B.

Figure 1: The construction of matrices A and B for the reduction in the proof of Theorem 1. The notation $w_{v,u}$ denotes the weight of the edge (v, u) in the original graph G.

problem. In the instance of the MINIMUM TRIANGLE problem, we are given a 3-partite undirected graph $G = (V_1 \cup V_2 \cup U,\ E)$ with positive edge weights such that $|V_1| = |V_2| = n$, $|U| = m$. We want to find a triangle of minimum weight in the graph G. To perform the reduction, we define a weighted directed and acyclic graph $G' = (\{1, 2\} \cup V_1 \cup V_2 \cup U \cup U',\ E')$. Nodes in U' are in one-to-one correspondence with nodes in U and $|U'| = m$. E' is defined as follows. We add all edges of G between nodes in U and V_1 directed from U towards V_1 and similarly, we add all edges of G between nodes in V_1 and V_2 directed from V_1 towards V_2. Instead of having edges between nodes in V_2 and U, we add the corresponding edges of G between nodes in V_2 and U' directed from V_2 towards U'. Moreover, we add additional edges of weight 0 to create a path P of $m + 1$ nodes, starting from node 1 and going through all nodes in U in some order. Finally, we create another path P' of $m + 1$ nodes going through all nodes in U' in the same order as their counterparts on path P and ending at node 2. These edges have weight 0 apart from the last one, entering node 2, which has weight $-C$ (a sufficiently large negative constant)[3].

We create an instance of the SHORTEST WALK problem by setting $T = m + 4$ and A to be the weighted adjacency matrix of G' that takes value $+\infty$ (or a sufficiently large integer) for non-existent edges and self-loops.

The optimal walk of the SHORTEST WALK instance must include the edge of weight $-C$ entering node 2 since otherwise the cost will be non-negative. Moreover, the walk must reach node 2 exactly at the last step since otherwise the cost will be $+\infty$ as there are no outgoing edges from node 2. By the choice of T, the walk leaves path P at some node $u \in U$, then visits nodes v_1 and v_2 in V_1 and V_2, respectively, and subsequently moves to node $u' \in U'$ where u' is the counterpart of u on path P'. The total cost of the walk is thus the weight of the triangle (u, v_1, v_2) in G, minus C. Therefore, the optimal walk has cost equal to the weight of the minimum triangle up to the additive constant C. □

Notice that when $T > n$, the runtime of the Viterbi algorithm is no longer optimal. We now present a faster algorithm with a total running time $\log T \cdot n^3/2^{\Omega(\sqrt{\log n})}$.

As we show in Section 7, the general VITERBI PATH problem reduces, according to Equation 2, to computing $(\min, +)$ matrix-vector products. In the case of unary alphabet, it corresponds to computing $(\min, +)$ matrix-vector product T times as follows: $A \oplus A \oplus ... \oplus A \oplus z$. This can be equivalently performed by first computing all $(\min, +)$ matrix-matrix products $A^{\oplus T} = A \oplus A \oplus ... \oplus A$ using exponentiation with repeated squaring and then multiplying the resulting matrix with the vector z. This requires only $O(\log T)$ matrix $(\min, +)$-multiplications. Using the currently best algorithm for $(\min, +)$ matrix product (Williams, 2014), we get an algorithm with total running time $\log T \cdot n^3/2^{\Omega(\sqrt{\log n})}$.

6. Hardness for sparse HMMs

The VITERBI PATH lower-bounds we have provided apply to the case where the HMM has all n^2 possible edges.

For sparse HMMs that have at most m edges out of the

[3] Since the definition of SHORTEST WALK doesn't allow negative weights, we can equivalently set its weight to be 0 and add C to all the other edge weights.

n^2 possible ones, i.e. the transition matrix has at most m non-zero probabilities, the VITERBI PATH problem can be easily solved in $O(Tm)$ time. The lower bounds that we presented in the paper can be adapted directly for this case to show that no faster algorithm exists that runs in time $O(Tm)^{1-\varepsilon}$. This can be easily seen via a padding argument. Consider a hard instance for VITERBI PATH on a dense HMM with $\sqrt{m}$ states and m edges. Adding $n-\sqrt{m}$ additional states with self-loops, we obtain a sparse instance with n states and $m+n-\sqrt{m} = O(m)$ edges. Thus, any algorithm that computes the optimal Viterbi Path in $O(Tm)^{1-\varepsilon}$ time for the resulting instance would solve the original instance with $\sqrt{m}$ states in $O\left(T(\sqrt{m})^2\right)^{1-\varepsilon}$ time contradicting the corresponding lower bound.

This observation directly gives the following lower bounds for VITERBI PATH problem, parametrized by the number m of edges in an HMM with n states.

Theorem 6. *The* VITERBI PATH *problem requires* $\Omega(Tm)^{1-o(1)}$ *time for an HMM with* m *edges and* n *states, assuming the APSP Conjecture.*

Theorem 7. *For any* $C, \varepsilon > 0$*, the* VITERBI PATH *problem on* $T = \Theta(m^C)$ *observations from an alphabet of size* $\Theta(m^\varepsilon)$ *requires* $\Omega(Tm)^{1-o(1)}$ *time assuming the* k*-Clique Conjecture for* $k = \lceil \frac{C}{\varepsilon} \rceil + 2$*.*

Theorem 8. *The* VITERBI PATH *problem requires* $\Omega(Tm)^{1-o(1)}$ *time when* $T \leq \sqrt{m}$ *even if the size of the alphabet is* $\sigma = 1$*, assuming the APSP Conjecture.*

7. A faster VITERBI PATH algorithm

In this section, we present a faster algorithm for the VITERBI PATH problem, when there are only few distinct transition probabilities in the underlying HMM.

Theorem 3. *When there are fewer than* $2^{\varepsilon\sqrt{\log n}}$ *distinct transition probabilities for a constant* $\varepsilon > 0$*, there is a* $Tn^2/2^{\Omega(\sqrt{\log n})}$ *randomized algorithm for the* VITERBI PATH *problem that succeeds whp.*

The number of distinct transition probabilities is equal to the number of distinct entries in matrix $\tilde{A}$ in Definition 1. The same is true for matrix A in the additive version of VITERBI PATH, in Definition 2. So, from the theorem statement we can assume that matrix A has at most $2^{\varepsilon\sqrt{\log n}}$ different entries for some constant $\varepsilon > 0$.

To present our algorithm, we revisit the definition of VITERBI PATH. We want to compute a path $u_0 = 1, u_1, \ldots, u_T$ that minimizes the quantity:

$$\min_{u_0=1,u_1,\ldots,u_T} \sum_{t=1}^{T} \left[A(u_{t-1}, u_t) + B(u_t, s_t)\right]. \quad (1)$$

Defining the vectors $b_t = B(\cdot, s_t)$, we note that (1) is equal to the minimum entry in the vector obtained by a sequence of T $(\min, +)$ matrix-vector products[4] as follows:

$$A \oplus (\ldots (A \oplus (A \oplus (A \oplus z + b_1) + b_2) + b_3) \ldots) + b_T \quad (2)$$

where z is a vector with entries $z_1 = 0$ and $z_i = \infty$ for all $i \neq 1$. Vector z represents the cost of being at node i at time 0. Vector $(A \oplus z + b_1)$ represents the minimum cost of reaching each node at time 1 after seeing observation s_1. After T steps, every entry i of vector (2) represents the minimum minimum cost of a path that starts at $u_0 = 1$ and ends at $u_T = i$ after T observations. Taking the minimum of all entries gives the cost of the solution to the VITERBI PATH instance.

To evaluate (2), we design an online $(\min, +)$ matrix-vector multiplication algorithm. In the online matrix-vector multiplication problem, we are given a matrix and a sequence of vectors in online fashion. We are required to output the result of every matrix-vector product before receiving the next vector. Our algorithm for online $(\min, +)$ matrix-vector multiplication is based on a recent algorithm for online Boolean matrix-vector multiplication by Green Larsen and Williams (Larsen & Williams, 2017):

Theorem 9 (Green Larsen and Williams (Larsen & Williams, 2017))**.** *For any matrix* $M \in \{0,1\}^{n\times n}$ *and any sequence of* $T = 2^{\omega(\sqrt{\log n})}$ *vectors* $v_1, \ldots, v_T \in \{0,1\}^n$*, online Boolean matrix-vector multiplication of* M *and* v_i *can be performed in* $n^2/2^{\Omega(\sqrt{\log n})}$ *amortized time whp. No preprocessing is required.*

We show the following theorem for online $(\min, +)$ matrix-vector multiplication, which gives the promised runtime for the VITERBI PATH problem[5] since we are interested in the case where T and n are polynomially related, i.e. $T = n^{\Theta(1)}$.

Theorem 10. *Let* $A \in \mathbb{R}^{n\times n}$ *be a matrix with at most* $2^{\varepsilon\sqrt{\log n}}$ *distinct entries for a constant* $\varepsilon > 0$*. For any sequence of* $T = 2^{\omega(\sqrt{\log n})}$ *vectors* $v_1, \ldots, v_T \in \mathbb{R}^n$*, online* $(\min, +)$ *matrix-vector multiplication of* A *and* v_i *can be performed in* $n^2/2^{\Omega(\sqrt{\log n})}$ *amortized time whp. No preprocessing is required.*

Proof. We will show the theorem for the case where $A \in \{0, +\infty\}^{n\times n}$. The general case where matrix A has $d \leq 2^{\varepsilon\sqrt{\log n}}$ distinct values $a^1, ..., a^d$ can be handled by creating d matrices $A^1, ..., A^d$, where each matrix A^k has entries $A^k_{ij} = 0$ if $A_{ij} = a^k$ and $+\infty$ otherwise. Then, vector

[4] A $(\min, +)$ product between a matrix M and a vector v is denoted by $M \oplus v$ and is equal to a vector u where $u_i = \min_j(M_{i,j} + v_j)$.

[5] Even though computing all $(\min, +)$ products does not directly give a path for the VITERBI PATH problem, we can obtain one at no additional cost by storing back pointers. This is standard and we omit the details.

$r = A \oplus v$ can be computed by computing $r^k = A^k \oplus v$ for every k and setting $r_i = \min_k(r_i^k + a^k)$. This introduces a factor of $2^{\varepsilon\sqrt{\log n}}$ in amortized runtime but the final amortized runtime remains $n^2/2^{\Omega(\sqrt{\log n})}$ if $\varepsilon > 0$ is sufficiently small. From now on we assume that $A \in \{0, +\infty\}^{n\times n}$ and define the matrix $\bar{A} \in \{0,1\}^{n\times n}$ whose every entry is 1 if the corresponding entry at matrix A is 0 and 0 otherwise.

For every query vector v, we perform the following:

- Sort indices $i_1, ..., i_n$ such that $v_{i_1} \leq ... \leq v_{i_n}$ in $O(n \log n)$ time.
- Partition the indices into $p = 2^{\alpha\sqrt{\log n}}$ sets, where set S_k contains indices $i_{(k-1)\lceil \frac{n}{p} \rceil + 1}, ..., i_{k\lceil \frac{n}{p} \rceil}$.
- Set $r = (\perp, ..., \perp)^T$, where $\perp$ indicates an undefined value.
- For $k = 1...p$ fill the entries of r as follows:
  - Let $\mathbb{I}_{S_k}$ be the indicator vector of S_k that takes value 1 at index i if $i \in S_k$ and 0 otherwise.
  - Compute the Boolean matrix-vector product $\pi^k = \bar{A} \odot \mathbb{I}_{S_k}$ using the algorithm from Theorem 9.
  - Set $r_j = \min_{i\in S_k}(A_{j,i} + v_i)$ for all $j \in [n]$ such that $r_j = \perp$ and $\pi_j^k = 1$.
- Return vector r.

Runtime of the algorithm per query The algorithm performs $p = 2^{\alpha\sqrt{\log n}}$ Boolean matrix-vector multiplications, for a total amortized cost of $p \cdot n^2/2^{\Omega(\sqrt{\log n})} = n^2/2^{\Omega(\sqrt{\log n})}$ for a small enough constant $\alpha > 0$. Moreover, to fill an entry r_j the algorithm requires going through all elements in some set S_k for a total runtime of $O(|S_k|) = n/2^{\Omega(\sqrt{\log n})}$. Thus, for all entries p_j the total time required is $n^2/2^{\Omega(\sqrt{\log n})}$. The runtime of the other steps is dominated by these two operations so the algorithm takes $n^2/2^{\Omega(\sqrt{\log n})}$ amortized time per query.

Correctness of the algorithm To see that the algorithm correctly computes the $(\min, +)$ product $A \oplus v$, observe that the algorithm fills in the entries of vector r from smallest to largest. Thus, when we set a value to entry r_j we never have to change it again. Moreover, if the value r_j gets filled at step k, it must be the case that $\pi_j^{k'} = 0$ for all $k' < k$. This means that for all indices $i \in S_1 \cup ... \cup S_{k-1}$ the corresponding entry $A_{j,i}$ was always $+\infty$. □

Acknowledgments

We thank Piotr Indyk for many helpful discussions, for comments on an earlier version of the writeup and for suggestion on how to improve the presentation. We also thank the anonymous reviewers for their careful reviews. This work was supported in part by an IBM PhD Fellowship, the NSF and the Simons Foundation.

References

Abboud, Amir and Williams, Virginia Vassilevska. Popular conjectures imply strong lower bounds for dynamic problems. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pp. 434–443. IEEE, 2014.

Abboud, Amir, Williams, Virginia Vassilevska, and Weimann, Oren. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming*, pp. 39–51. Springer, 2014.

Abboud, Amir, Backurs, Arturs, and Williams, Virginia Vassilevska. If the Current Clique Algorithms are Optimal, so is Valiant's Parser. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pp. 98–117. IEEE, 2015a.

Abboud, Amir, Grandoni, Fabrizio, and Williams, Virginia Vassilevska. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1681–1697. SIAM, 2015b.

Abboud, Amir, Williams, Virginia Vassilevska, and Yu, Huacheng. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pp. 41–50. ACM, 2015c.

Abdel-Hamid, Ossama, Mohamed, Abdel-rahman, Jiang, Hui, and Penn, Gerald. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4277–4280. IEEE, 2012.

Alon, Noga and Boppana, Ravi B. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1): 1–22, 1987.

Alon, Noga, Krivelevich, Michael, and Sudakov, Benny. Finding a large hidden clique in a random graph. *Random Struct. Algorithms*, 13(3-4):457–466, 1998.

Alon, Noga, Andoni, Alexandr, Kaufman, Tali, Matulef, Kevin, Rubinfeld, Ronitt, and Xie, Ning. Testing k-wise and almost k-wise independence. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pp. 496–505, 2007.

Altun, Yasemin, Tsochantaridis, Ioannis, Hofmann, Thomas, et al. Hidden markov support vector machines. In *ICML*, volume 3, pp. 3–10, 2003.

Amengual, Juan C and Vidal, Enrique. Efficient error-correcting viterbi parsing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(10):1109–1116, 1998.

Attias, Hagai. Planning by probabilistic inference. In *AISTATS*, 2003.

Backurs, Arturs, Dikkala, Nishanth, and Tzamos, Christos. Tight Hardness Results for Maximum Weight Rectangles. In *International Colloquium on Automata, Languages, and Programming*, 2016.

Bengio, Samy. An asynchronous hidden markov model for audio-visual speech recognition. *Advances in Neural Information Processing Systems*, pp. 1237–1244, 2003.

Bourlard, Herve A and Morgan, Nelson. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

Bringmann, Karl, Grønlund, Allan, and Larsen, Kasper Green. A dichotomy for regular expression membership testing. *arXiv preprint arXiv:1611.00918*, 2016.

Cairo, Massimo, Farina, Gabriele, and Rizzi, Romeo. Decoding Hidden Markov Models Faster Than Viterbi Via Online Matrix-Vector (max,+)-Multiplication. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Churbanov, Alexander and Winters-Hilt, Stephen. Implementing EM and Viterbi algorithms for Hidden Markov Model in linear memory. *BMC bioinformatics*, 9(1):1, 2008.

Cohen, Ira, Garg, Ashutosh, Huang, Thomas S, et al. Emotion recognition from facial expressions using multilevel hmm. In *Neural information processing systems*, volume 2. Citeseer, 2000.

Collins, Michael. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 1–8. Association for Computational Linguistics, 2002.

Collobert, Ronan. Deep learning for efficient discriminative parsing. In *AISTATS*, volume 15, pp. 224–232, 2011.

Esposito, Roberto and Radicioni, Daniele P. Carpediem: Optimizing the viterbi algorithm and applications to supervised sequential learning. *Journal of Machine Learning Research*, 10(Aug):1851–1880, 2009.

Felzenszwalb, Pedro F, Huttenlocher, Daniel P, and Kleinberg, Jon M. Fast algorithms for large-state-space

HMMs with applications to web usage analysis. *Advances in NIPS*, 16:409–416, 2004.

Grice, J Alicia, Hughey, Richard, and Speck, Don. Reduced space sequence alignment. *Computer applications in the biosciences: CABIOS*, 13(1):45–53, 1997.

Håstad, Johan. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

Hazan, Elad and Krauthgamer, Robert. How hard is it to approximate the best nash equilibrium? *SIAM J. Comput.*, 40(1):79–91, 2011.

Huang, Xuedong, Acero, Alex, Hon, Hsiao-Wuen, and Foreword By-Reddy, Raj. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice Hall PTR, 2001.

Jerrum, Mark. Large cliques elude the metropolis process. *Random Struct. Algorithms*, 3(4):347–360, 1992.

Kaji, Nobuhiro, Fujiwara, Yasuhiro, Yoshinaga, Naoki, and Kitsuregawa, Masaru. Efficient staggered decoding for sequence labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 485–494. Association for Computational Linguistics, 2010.

Karp, Richard M. Reducibility among combinatorial problems. In *Complexity of computer computations*, pp. 85–103. Springer, 1972.

Kim, Seyoung and Smyth, Padhraic. Segmental hidden markov models with random effects for waveform modeling. *Journal of Machine Learning Research*, 7(Jun): 945–969, 2006.

Larsen, Kasper Green and Williams, Ryan. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2182–2189. SIAM, 2017.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, Peng, Dong, Ming, Liang, Weiqian, and Liu, Runsheng. Design of Speech Recognition Co-Processor for the Embedded Implementation. In *Electron Devices and Solid-State Circuits, 2007. EDSSC 2007. IEEE Conference on*, pp. 1163–1166. IEEE, 2007.

Li, Peng and Tang, Hua. Design a co-processor for Output Probability Calculation in speech recognition. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pp. 369–372. IEEE, 2009.

Lifshits, Yury, Mozes, Shay, Weimann, Oren, and Ziv-Ukelson, Michal. Speeding up HMM decoding and training by exploiting sequence repetitions. *Algorithmica*, 54(3):379–399, 2009.

Lin, Kuang, Simossis, Victor A, Taylor, Willam R, and Heringa, Jaap. A simple and fast secondary structure prediction method using hidden neural networks. *Bioinformatics*, 21(2):152–159, 2005.

Mahmud, Md Pavel and Schliep, Alexander. Speeding up Bayesian HMM by the four Russians method. In *International Workshop on Algorithms in Bioinformatics*, pp. 188–200. Springer, 2011.

Mannini, Andrea and Sabatini, Angelo Maria. Machine learning methods for classifying human physical activity from on-body accelerometers. *Sensors*, 10(2):1154–1175, 2010.

Mohamed, Abdel-rahman, Dahl, George E, and Hinton, Geoffrey. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

Nefian, Ara V, Liang, Luhong, Pi, Xiaobo, Xiaoxiang, Liu, Mao, Crusoe, and Murphy, Kevin. A coupled hmm for audio-visual speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 2, pp. II–2013. IEEE, 2002.

Peng, Jian, Bo, Liefeng, and Xu, Jinbo. Conditional neural fields. In *Advances in neural information processing systems*, pp. 1419–1427, 2009.

Rabiner, Lawrence R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Roditty, Liam and Zwick, Uri. On dynamic shortest paths problems. In *Algorithms–ESA 2004*, pp. 580–591. Springer, 2004.

Siddiqi, Sajid M and Moore, Andrew W. Fast inference and learning in large-state-space hmms. In *Proceedings of the 22nd international conference on Machine learning*, pp. 800–807. ACM, 2005.

Tarnas, Christopher and Hughey, Richard. Reduced space hidden Markov model training. *Bioinformatics*, 14(5): 401–406, 1998.

Viterbi, Andrew J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

Williams, Ryan. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pp. 664–673. ACM, 2014.

Williams, Virginia Vassilevska and Williams, Ryan. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pp. 645–654. IEEE, 2010.

Young, Steve, Evermann, Gunnar, Gales, Mark, Hain, Thomas, Kershaw, Dan, Liu, Xunying, Moore, Gareth, Odell, Julian, Ollason, Dave, Povey, Dan, et al. *The HTK book*, volume 2. Entropic Cambridge Research Laboratory Cambridge, 1997.