A. Hardness of VITERBI PATH with small alphabet: Proof

Throughout the proof, we set $p = \lceil \frac{C}{\varepsilon} \rceil$ and $\alpha = \frac{C}{p} \leq \varepsilon$.

We will perform a reduction from the MIN-WEIGHT k-CLIQUE problem for $k = p + 2$ to the VITERBI PATH problem. In the instance of the MIN-WEIGHT k-CLIQUE problem, we are given a k-partite graph $G = (V_1 \cup V_2 \cup U_1 \ldots \cup U_p,\ E)$ such that $|V_1| = |V_2| = n$ and $|U_1| = \ldots = |U_p| = m = \Theta(n^\alpha)$. We want to find a clique of minimum weight in the graph G. Before describing our final VITERBI PATH instance, we first define a weighted directed graph $G' = (\{1, 2, 3\} \cup V_1 \cup V_2,\ E')$ similar to the graph in the proof of Theorem 1. E' contains all the edges of G between V_1 and V_2, directed from V_1 towards V_2, edges from node 1 towards all nodes in V_1 of weight 0 and edges from all nodes in V_2 towards node 2 of weight 0. We also add a self-loop at nodes 1 and 3 of weight 0 as well as an edge of weight 0 from node 2 towards node 3. We obtain the final graph G'' as follows:

- For every node $v \in V_1$, we replace the directed edge $(1, v)$ with a path $1 \to a_{v,1} \to \ldots \to a_{v,p} \to v$ by adding p intermediate nodes. All edges on the path have weight 0.
- For every node $v \in V_2$, we replace the directed edge $(v, 2)$ with a path $v \to b_{v,1} \to \ldots \to b_{v,p} \to 2$ by adding p intermediate nodes. All edges on the path have weight 0.
- Finally, we replace the directed edge $(2, 3)$ with a path $2 \to c_1 \to \ldots \to c_Z \to 3$ by adding Z intermediate nodes, where 2^Z is a strict upper bound on the weight of any k-clique[6]. All edges on the path have weight 0.

We create an instance of the VITERBI PATH problem (A, B, S) as described below. Figure 2 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 $U_1 \cup \ldots \cup U_p \cup \{\perp, \perp_0, \perp_1, \perp_F\}$ and thus matrix B has $O(n)$ rows and $\sigma = p \cdot m + 4 = O(n^\alpha)$ columns.

  For all $v \in V_1$, every $i \in [p]$ and every $u \in U_i$, $B(a_{v,i}, u)$ is equal to the weight of the edge (v, u) in graph G. Similarly, for all $v \in V_2$, every $j \in [p]$ and every $u \in U_j$, $B(b_{v,j}, u)$ is equal to the weight of the edge (v, u) in graph G.

  Moreover, for all $i \in \{1, \ldots, Z\}$, $B(c_i, \perp_1) = 2^{i-1}$ and $B(c_i, \perp_0) = 0$. Finally, $B(v, \perp) = +\infty$ for all nodes $v \notin \{1, 3\}$ while $B(v, \perp_F) = +\infty$ for all nodes $v \neq 3$. All remaining entries of matrix B are 0.
- Sequence S is generated by appending for every tuple $(u_1, \ldots, u_p) \in U_1 \times \ldots \times U_p$ the following observations in this order: Initially we add the observations $(u_1, \ldots, u_p, \perp_0, \perp_0, u_1, \ldots, u_p, \perp_0)$. Moreover, let W be the total weight of the clique $(u_1, \ldots, u_p)$ in the graph G. We add Z observations encoding W in binary[7] starting with the least significant bit. For example, if $W = 11$ and $Z = 5$, the binary representation is 01011_2 and the observations we add are $\perp_1, \perp_1, \perp_0, \perp_1, \perp_0$ in that order. Finally, we append a $\perp$ observation at the end.

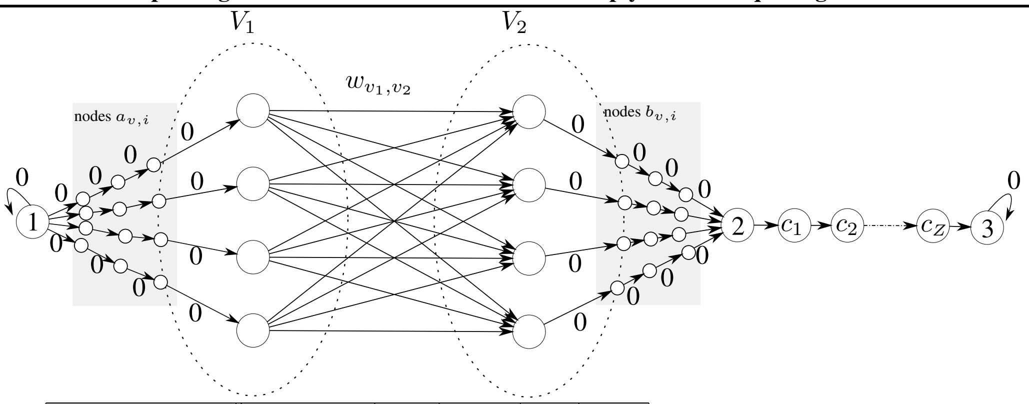
  Notice, that for each tuple, we append exactly $Z + 2p + 4 = Z + 2k$ observations. Thus, the total number of observations is $m^p(Z + 2k)$. We add a final $\perp_F$ observation at the end and set $T = m^p(Z + 2k) + 1$.

Size of graph G'' Since the edge-weights in the k-Clique conjecture are upper bounded by $n^{O(k)}$, the length of the path $2 \to c_1 \to \ldots \to c_Z \to 3$ can be upper bounded by $Z + 2 \leq O(k \log n)$. The length of every path $1 \to a_{v,1} \to \ldots \to a_{v,p} \to v$ or $v \to b_{v,1} \to \ldots \to b_{v,p} \to 2$ is $p + 2 = k$. The number of vertices of G'' can now be upper bounded by $O(k \log n + kn) \leq O(n)$ since $k = O(1)$.

Correctness of the reduction Since the MIN-WEIGHT k-CLIQUE instance requires $\Omega\left(|V_1| \cdot |V_2| \cdot \prod_{i=1}^{p} |U_i|\right)^{1-o(1)} = \Omega(Tn^2)^{1-o(1)}$ time, the following claim implies that the above VITERBI PATH instances require $\Omega(Tn^2)^{1-o(1)}$ time. The alphabet size used is at most $O(n^\alpha)$ and $\alpha \leq \varepsilon$ and the theorem follows.

[6] A trivial such upper bound is k^2 times the weight of the maximum edge.

[7] Since 2^Z is a strict upper-bound on the clique size at most Z digits are required.



Node	$u \in U_j$	$\perp_0$	$\perp_1$	$\perp$	$\perp_F$
1	0	0	0	0	∞
$a_{v,i}$ for $v \in V_1$	$w_{u,v}\mathbb{1}_{i=j}$	0	0	∞	∞
$v \in V_1 \cup V_2$	0	0	0	∞	∞
$b_{v,i}$ for $v \in V_2$	$w_{u,v}\mathbb{1}_{i=j}$	0	0	∞	∞
2	0	0	0	∞	∞
c_i	0	0	2^{i-1}	∞	∞
3	0	0	0	0	0

Figure 2: The construction of matrices A and B for the reduction in the proof of Theorem 2.

Claim 2. *The weight of the solution to the* VITERBI PATH *instance is equal to the minimum weight of a k-clique in the graph G.*

Proof. The optimal path for the VITERBI PATH instance begins at node 1. It must end in node 3 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 3. Thus, the path first loops in node 1 and then goes from node 1 to node 3 during the sequence of $Z + 2k$ consecutive observations corresponding to some tuple $(u_1, ..., u_p) \in U_1 \times ... \times U_p$ and stays in node 3 until the end. Let v_1 and v_2 be the nodes in V_1 and V_2, respectively, that are visited when moving from node 1 to node 3. The only steps of non-zero cost happen during the subsequence of observations corresponding to the tuple $(u_1, ..., u_p)$:

1. When the subsequence begins with u_1, the path jumps to node $a_{v_1,1}$ which has a cost $B(a_{v_1,1}, u_1)$ equal to the edge-weight (v_1, u_1) in graph G. It then continues on to nodes $a_{v_1,2}, ..., a_{v_1,p}$ when seeing observations $u_2, ..., u_p$. The total cost of these steps is $\sum_{i=1}^{p} B(a_{v_1,i}, u_i)$ which is the total weight of edges $(v_1, u_1), ..., (v_1, u_p)$ in graph G.

2. For the next two observations $\perp_0, \perp_0$, the path jumps to nodes v_1 and v_2. The first jump has no cost while the latter has cost $A(v_1, v_2)$ equal to the weight of the edge (v_1, v_2) in G.

3. The subsequence continues with observations $u_1, ..., u_p$ and the path jumps to nodes $b_{v_2,1}, ..., b_{v_2,p}$ which has a total cost $\sum_{i=1}^{p} B(b_{v_2,i}, u_i)$ which is equal to the total weight of edges $(v_2, u_1), ..., (v_2, u_p)$ in graph G.

4. The path then jumps to node 2 at no cost at observation $\perp_0$.

5. The path then moves on to the nodes $c_1, ..., c_Z$. The total cost of those moves is equal to the total weight of the clique $(u_1, ..., u_p)$ since the observations $\perp_0$ and $\perp_1$ that follow encode that weight in binary.

The overall cost of the path is exactly equal to the weight of the k-clique $(v_1, v_2, u_1, ..., u_p)$ in G. Minimizing the cost of the path in this instance is therefore the same as finding the minimum weight k-clique in G. □

B. Reduction from MIN-WEIGHT k-CLIQUE to MIN-WEIGHT k-CLIQUE in k-partite graphs

In this section, we show the following lemma using standard techniques.

Lemma 1. *Consider the* MIN-WEIGHT k-CLIQUE *problem in k-partite graphs $G = (V_1 \cup \ldots \cup V_k,\ E)$ with $|V_i| = n_i$. If for all i, j we have that $n_i = n_j^{\Theta(1)}$, then the* MIN-WEIGHT k-CLIQUE *problem for this class of instances requires $\Omega\left(\prod_{i=1}^{k} n_i\right)^{1-o(1)}$ time assuming the k-Clique conjecture.*

Proof. Without loss of generality assume that $n_1 \geq n_i$ for all i and let $n = n_1$. Assume, that there is an $O\left(\prod_i n_i\right)^{1-\varepsilon}$ algorithm that finds a minimum weight k-clique in k-partite graphs with $|V_i| = n_i$ for all i. We can use this faster algorithm to find a k-clique in a graph $G = (V,\ E)$ where $|V| = n$, as follows: Let $\mathcal{V}^i$ be a partition of V into $\frac{n}{n_i}$ sets of size n_i. For all $(V_1, \ldots, V_k) \in \mathcal{V}^1 \times \cdots \times \mathcal{V}^k$, we create a k-partite graph $G' = (V_1 \cup \ldots \cup V_k, E')$ by adding edges corresponding to the edges of graph G between nodes across partitions and find the minimum weight k-clique in the graph G' using the faster algorithm. Computing the minimum weight k-clique out of all the graphs we consider gives the solution to the MIN-WEIGHT k-CLIQUE instance on G. The total runtime is $\prod_{i=1}^{k} \frac{n}{n_i} \cdot O\left(\prod_{i=1}^{k} n_i\right)^{1-\varepsilon} = n^{k-\Omega(\varepsilon)}$ which would violate the k-Clique conjecture. The previous equality holds because of the assumption that $n_i = n_j^{\Theta(1)}$. □

C. Sum of Probabilities

In the definition of the additive version of VITERBI PATH, we didn't impose any constraint on the weights. In the multiplicative version where weights correspond to probabilities, we have the restriction that probabilities of transition from each vertex sum to 1.

To convert an instance $\mathcal{I}_{Add}$ of the additive VITERBI PATH formulation to an equivalent instance $\mathcal{I}_{Mul}$ in the multiplicative setting, we add a shift of $\log n$ to all entries of A and a shift of $\log T$ to entries of matrix B. This doesn't change the optimal solution but only changes its value by an additive shift of $T \log n + T \log T$. This transformation makes all probabilities in the $\mathcal{I}_{Mul}$ instance small enough such that transition probabilities sum to less than 1 and similarly probabilities of outputting observations sum to less than 1. To handle the remaining probability, we introduce an additional node α and an additional symbol γ in the alphabet of observations. Every original transitions to node α with its remaining transition probability and outputs observation γ with its remaining transition probability. We require that node α outputs observation γ with 100% probability. As we never observe γ in the sequence of observations, the optimal solution must never go through node α and thus the optimal solution remains the same.

The transformation above requires introducing an additional symbol in the alphabet. For our reduction of ALL-PAIRS SHORTEST PATHS to VITERBI PATH when $\sigma = 1$, we don't want the alphabet to increase. We describe an alternative transformation for $\sigma = 1$ that doesn't introduce additional symbols. This case corresponds to the SHORTEST WALK instance and matrix B is irrelevant.

We first scale all weights in matrices A, by dividing by some large weight W, so that all values are between 0 and 1 and then add a shift of $\log n$ to all of them. This doesn't change the optimal solution but only changes its value by a multiplicative factor W and an additive shift of $T \log n$. After this transformation all values are between $\log n$ and $1 + \log n$ and thus the corresponding probabilities in the $\mathcal{I}_{Mul}$ instance are at most $1/n$. This causes transition probabilities to sum to less than 1. To assign the remaining probability, we introduce a clique of $4n$ additional nodes. All nodes in the clique have probability $\frac{1}{4n}$ of transition to any other node in the clique and 0 probability of transition to any of the original node. For every original node, we spread its remaining transition probability evenly to the $4n$ nodes of the clique. It is easy to see that the optimal solution to the VITERBI PATH problem will not change, as it is never optimal to visit any of the nodes in the clique. This is because all edges in the original graph have weight at most $1 + \log n$ while if a node in the clique is visited the path must stay in the clique at a cost of $2 + \log n$ per edge.