Inverting the Viterbi Algorithm: An Abstract Framework for Structure Design

Michael Schnall-Levin MSCHNALL@MATH.MIT.EDU
Leonid Chindelevitch LEONIDUS@MATH.MIT.EDU
Bonnie Berger[1] BAB@MIT.EDU
Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge MA

Abstract

Probabilistic grammatical formalisms such as hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs) have been extensively studied and widely applied in a number of fields. Here, we introduce a new algorithmic problem on HMMs and SCFGs that arises naturally from protein and RNA design, and which has not been previously studied. The problem can be viewed as an inverse to the one solved by the Viterbi algorithm on HMMs or by the CKY algorithm on SCFGs. We study this problem theoretically and obtain the first algorithmic results. We prove that the problem is NP-complete, even for a 3-letter emission alphabet, via a reduction from 3-SAT, a result that has implications for the hardness of RNA secondary structure design. We then develop a number of approaches for making the problem tractable. In particular, for HMMs we develop a branch-and-bound algorithm, which can be shown to have fixed-parameter tractable worst-case running time, exponential in the number of states of the HMM but linear in the length of the structure. We also show how to cast the problem as a Mixed Integer Linear Program.

1. Introduction

Probabilistic grammatical formalisms such as hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs) have found many applications in areas such as computational biology and natural language processing. Because of their intuitive representation, their power to capture some of the essential relationships present in data, and the existence of polynomial-time algorithms (such as the Viterbi algorithm) and practical training procedures (such as the Baum-Welch algorithm), these formalisms have enjoyed tremendous popularity in the past decades.

Three natural problems for a model have been described: the decoding problem (given a model and a sequence, find the most likely derivation), the evaluation problem (given a model and a sequence, find the likelihood of the sequence being generated), and the learning problem (given a set of sequences, learn the parameters of the underlying model). In this paper, we identify another natural problem on HMMs and SCFGs, which is the inverse of the decoding problem: given a derivation and a model, find a sequence for which this derivation is the most likely one. Because the decoding problem is solved by the Viterbi algorithm in HMMs and by the CKY algorithm in SCFGs, we refer to our problem for these two models as the Inverse-Viterbi and the Inverse-CKY problem, respectively.

The motivation for our problem comes from protein and RNA design. The design of biological molecules with a desired structure is a long sought-after goal in computational biology. While a number of achievements have been made in protein structure design, the problem remains difficult (Butterfoss & Kuhlman, 2005; Park et al., 2004; Pokala & M., 2001). For RNA, there has been recent interest in secondary structure design (Breaker, 1996), and a number of fairly successful heuristics have been developed to solve this problem (Hofacker, 1994; Andronescu, 2004; Busch & Backofen, 2006). Generally, structure design can be divided into two goals: the positive-design aspect of finding a sequence that has low energy in the desired structure, and the negative-design aspect of blocking the sequence from having low energy in other struc-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

[1]To whom correspondence should be addressed

tures. While some work has explored the negative-design aspect in protein structure design (Butterfoss & Kuhlman, 2005), most work has focused solely on the positive-design aspect. In RNA secondary structure design, the positive-design aspect is largely trivial (desired paired positions in the secondary structure can simply be chosen to be complementary bases) and the negative-design aspect, which involves attempting to block erroneous base pairings in other structures, is central to solving the problem.

Our goal in formulating the Inverse-Viterbi and Inverse-CKY problems is to simultaneously capture the positive-design and negative-design aspects of the design problem. Our framework for viewing design within the context of HMMs or SCFGs is the following. When HMMs (SCFGs) are used for structure prediction, the emitted string represents the biological sequence and the goal is to find the hidden state-path (derivation tree for SCFGs) that represents the structure this sequence will adopt. A state-path of high probability for that sequence is the analogue of a structure with low energy. By inverting this problem, we can use the same HMM (SCFG) for design. Now, a state-path (derivation tree) representing the desired structure is known and the goal is to find a sequence which will adopt this structure (i.e. a string for which this state-path is optimal).

Our Contribution. We have defined a novel problem (the Inverse-Viterbi problem) on HMMs and its analogue on SCFGs, that as far as we know has never been studied before. We show that the problem is NP-hard for HMMs (and as a result for SCFGs). We then give approaches for making the problem tractable. In particular, for HMMs we give a branch-and-bound algorithm. This algorithm can be shown to have fixed-parameter tractable running time: if there are K states, the emission alphabet is Σ, the path length is n, and all of the log-probabilities in the model are greater than $-B$ (so that there are no 0 transition probabilities and all the probabilities in the model are greater than e^{-B}) and are defined to a precision δ, then the branch-and-bound algorithm has worst-case running time $O((2B/\delta)^{K-2}nK^2|\Sigma|)$, which is exponential in the number of states but linear in the path length. We also show how to cast the problem as a simple Mixed Integer Linear Program.

Our hardness proof shows that the RNA secondary structure design problem is hard in a certain sense: a polynomial-time algorithm that only depends on the energy model for RNA secondary structure being SCFG-like, as is the case for the Zuker energy model (the most successful model curently available for RNA secondary structure prediction (Zuker & Stiegler, 1981)), without making additional assumptions on the particular form of the energy model, is not possible unless $P = NP$.

In presenting an abstract formulation of the design problem and giving the theoretical results derived in this paper, our goal is not to provide methods that will necessarily be immediately applied to the protein or RNA structure design problems. Instead, we believe that the abstract framework given in this paper may prove to be useful in understanding the design problem and facilitating the development of new methods for design. The Inverse-Viterbi and Inverse-CKY problems are novel and natural problems on HMMs and SCFGs, and so we believe a theoretical exploration is interesting in its own right.

2. Problem Description and Hardness Results

2.1. Definition of the Models

An HMM consists of a set $\mathcal{N}$ of K states and an alphabet Σ, with $\mathcal{N} \cap \Sigma = \emptyset$. The symbols in Σ are emitted on transitions between the states. The probability of emitting the symbol a when transitioning from the state s_k to the state s_l is specified by the value of the parameter $p^a_{s_k,s_l}$. These parameters determine the HMM. We assume (without loss of generality) that there is a unique initial state S.

The normalization condition requires that

$$\sum_{s_l \in \mathcal{N}} \sum_{a \in \Sigma} p^a_{s_k,s_l} = 1 \, \text{for} \, k = 1, \ldots, K$$

Similarly, an SCFG consists of a set $\mathcal{N}$ of K non-terminal symbols, and a set Σ of terminal symbols, with $\mathcal{N} \cap \Sigma = \emptyset$. The non-terminals are rewritten according to a set $\mathcal{R}$ of rewriting rules. The probability of applying each rewriting rule α is specified by the value of the parameter p_α. These parameters determine the SCFG. We assume (without loss of generality) that there is a unique starting non-terminal symbol S.

Every rule α replaces a single non-terminal with a string γ of non-terminals and terminals:

$$\alpha = N_k \rightarrow \gamma$$

Here N_k (the terminal symbol being rewritten) is referred to as the left-hand side of the rule, abbreviated as $l(\alpha)$.

The normalization condition requires that

$$\sum_{\{\alpha\in\mathcal{R}|l(\alpha)=N_k\}} p_\alpha = 1, \text{ for } k = 1\ldots,, K$$

We do not insist that the SCFG be in Chomsky Normal Form (CNF) because in some applications (such as RNA secondary structure design), the correspondence between the design and inverse problem defined in this paper may only be natural if the SCFG is not converted to CNF.

We use boldface letters to indicate sequences of symbols. A state-path of length n in the HMM is written as $\boldsymbol{\pi} = \pi_1 \ldots \pi_n$, where each π_i is a state in the HMM. Such a path emits a sequence of $n-1$ emission symbols, $\boldsymbol{\omega} = \omega_1 \ldots \omega_{n-1}$ where each ω_i is a symbol from Σ. The joint probability of a state-path $\boldsymbol{\pi}$ and an emission sequence $\boldsymbol{\omega}$ is given by $\Pr(\boldsymbol{\pi}, \boldsymbol{\omega}) = \prod_{i=1}^{n-1} p^{\omega_i}_{\pi_i,\pi_{i+1}}$. It is frequently more convenient to deal with sums rather than products, and so we work in log-space, taking $q^a_{s_1,s_2} := \log(p^a_{s_1,s_2})$ and therefore $\log(\Pr(\boldsymbol{\pi}, \boldsymbol{\omega})) = \sum_{i=1}^{n-1} q^{\omega_i}_{\pi_i,\pi_{i+1}}$.

A derivation of length n in the SCFG is the successive application of rewriting rules, beginning with the starting symbol S, which generates a yield $\boldsymbol{\omega} = \omega_1 \ldots \omega_n$ where each ω_i is a symbol from Σ. The derivation can be summarized in the form of a tree $\mathcal{T}$. The joint probability of a derivation tree $\mathcal{T}$ and a yield $\boldsymbol{\omega}$ is given by $\Pr(\mathcal{T}, \boldsymbol{\omega}) = \prod_{\alpha\in\mathcal{R}(\mathcal{T})} p_\alpha$, where $\mathcal{R}(\mathcal{T})$ denotes the multiset of rewriting rules used to derive $\mathcal{T}$. As with HMMs, it is convenient to work instead with the log-probabilities, $q_\alpha := \log p_\alpha$, which gives $\log(\Pr(\mathcal{T}, \boldsymbol{\omega})) = \sum_{\alpha\in\mathcal{R}(\mathcal{T})} q_\alpha$.

2.2. Definition of the Direct Problem

In the original Viterbi problem, one is given an emission sequence $\boldsymbol{\omega_0}$ from an HMM and the goal is to find the most likely state-path to have generated $\boldsymbol{\omega_0}$: the $\boldsymbol{\pi}$ that maximizes the conditional probability given the emission $\Pr(\boldsymbol{\pi}|\boldsymbol{\omega_0})$. Since $\Pr(\boldsymbol{\pi}|\boldsymbol{\omega_0}) = \frac{\Pr(\boldsymbol{\pi},\boldsymbol{\omega_0})}{\Pr(\boldsymbol{\omega_0})}$, and $\boldsymbol{\omega_0}$ is fixed, it is equivalent to simply maximize the joint probability $\Pr(\boldsymbol{\pi}, \boldsymbol{\omega_0})$. The Viterbi problem can therefore be expressed as: given $\boldsymbol{\omega_0}$, find an element of $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega_0})$ (we consider arg max as the *set* of all arguments maximizing the function). For an HMM with K states and an emission of length n, the Viterbi algorithm finds the best state-path using dynamic programming in time $O(nK^2|\Sigma|)$ (Viterbi, 1967).

Similarly, the direct problem for an SCFG is formulated as follows: given a yield $\boldsymbol{\omega}$, find the derivation tree $\mathcal{T}$ which maximizes the joint probability $\Pr(\mathcal{T}, \boldsymbol{\omega})$. In other words, given $\boldsymbol{\omega}$, we find an element of $\arg\max_{\mathcal{T}} \Pr(\mathcal{T}, \boldsymbol{\omega})$. The optimal derivation is referred to as the Viterbi parse of $\boldsymbol{\omega}$. For a derivation of length n in an SCFG with rewriting rules $\mathcal{R}$ in Chomsky Normal Form, the CKY algorithm finds the Viterbi parse in time $O(n^3|\mathcal{R}|)$ (Durbin et al., 1999). Modified versions of the CKY algorithm can also handle SCFGs in similar forms, such as those used in RNA structure prediction, with the same time complexity (for example see (Dowell & Eddy, 2004)).

2.3. Definition of the Inverse Problem

In the Inverse-Viterbi problem, a desired output of the Viterbi algorithm is known and the goal is to design an input to the Viterbi algorithm that will return this output. In mathematical terms the problem is: given a state-path $\boldsymbol{\pi_0}$, find an $\boldsymbol{\omega}$ so that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$, or determine that none exists.

In an HMM used for structure prediction, the above definition of the inverse problem captures what it means to do structure design: one knows the structure (state-path) and tries to find a sequence that has a higher score with that structure than with any other structure. It is important to emphasize that for many $\boldsymbol{\pi}$ there will be no such $\boldsymbol{\omega}$. In fact, it can be shown that only polynomially many paths are designable (Elizalde & Woods, 2006). This captures the notion that many structures are not designable: there is no sequence that will lead to these structures.

To illustrate this distinction, consider the 2-state HMM shown in Figure 1. Say that the desired state-path to design is $B^n = B \ldots B$. The most likely emission given this state-path is $a^{n-1} = a \ldots a$, but when run on such a path the Viterbi algorithm will not return B^n. In fact, the only sequence that the Viterbi algorithm will return B^n on is b^{n-1}. This simple case illustrates that to design a path of all B's it is important not just to pick emissions likely given this path, but to simulatenously block other possible paths, in this case those paths containing A's. Note further that the probability of b^{n-1} being emitted from B^n at random is $(0.2)^{n-1}$. Therefore, neither picking the most likely emission sequence nor randomly generating sequences from the state-path will in general solve the Inverse-Viterbi problem with probability greater than exponentially small in the length of the state-path.

We incorporate one generalization into our definition of the problem of inverting the Viterbi algorithm, because it seems natural to the design problem. We allow constraints on the emissions that can be chosen in any position (given as the Σ_i below). The algorithms we develop in this paper handle this generalization without any added complexity.

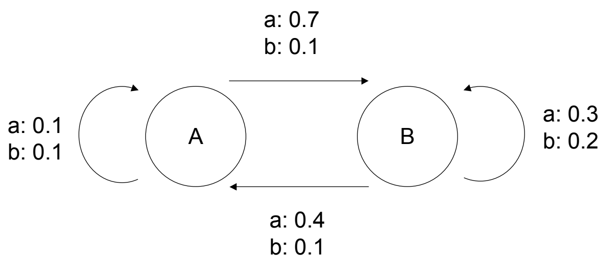

Figure 1. A 2-state HMM illustrating the distinction between the Inverse-Viterbi problem and the trivial problem of finding the most likely emission from a given state-path. The 2 states are A and B, while the 2 possible emissions are a and b. Each transition is marked with the possible emissions followed by their corresponding probabilities. In order to design B^n the only possible sequence is b^{n-1}, which is the least likely sequence to be produced by B^n.

INVERSE-VITERBI:
Input: An HMM, a state-path $\boldsymbol{\pi_0}$ of length n and for every position i in $1, \dots, n$ a set $\Sigma_i \subseteq \Sigma$ giving allowed emissions at position i.
Output: An $\boldsymbol{\omega}$ where each $\omega_i \in \Sigma_i$ so that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$, or $\emptyset$ if no such $\boldsymbol{\omega}$ exists.

Similarly, the inverse problem for an SCFG requires one to find an input that corresponds to a given output. In other words, given a derivation $\mathcal{T}_0$, we would like to find an $\boldsymbol{\omega}$ such that $\mathcal{T}_0$ is in $\arg\max_{\mathcal{T}} \Pr(\mathcal{T}, \boldsymbol{\omega})$, or determine that none exists. Note that this problem only makes sense if the tree $\mathcal{T}_0$ has had all of its leaves removed (we will call such a tree "naked"); in other words, the tree includes the specification of non-terminals but not the terminal symbols produced.

INVERSE-CKY:
Input: An SCFG, a naked derivation tree $\mathcal{T}_0$ that corresponds to an emitted string of n terminals and for every position i in $1, \dots, n$ a set $\Sigma_i \subseteq \Sigma$ giving the allowed emissions at position i.
Output: An $\boldsymbol{\omega}$ where each $\omega_i \in \Sigma_i$ so that $\mathcal{T}_0$ is in $\arg\max_{\mathcal{T}} \Pr(\mathcal{T}, \boldsymbol{\omega})$, or $\emptyset$ if no such $\boldsymbol{\omega}$ exists.

2.4. NP-hardness of the Inverse Problem

We now prove that the Inverse-Viterbi problem is NP-hard. To do so, we introduce the decision problem corresponding to Inverse-Viterbi:

DESIGNABLE:
Input: An HMM and a state-path $\boldsymbol{\pi_0}$
Output: YES if there is an $\boldsymbol{\omega}$ so that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$, otherwise NO.

An algorithm that solves Inverse-Viterbi would also solve Designable and so by proving Designable is NP-complete, we show that Inverse-Viterbi is NP-hard.

Theorem 1. *Designable is NP-Complete.*

Proof. Clearly Designable is in NP so we just need to show Designable is NP-hard. We do so by presenting a polynomial-time reduction from 3-SAT to Designable.

In outline, the construction is achieved by creating an HMM with one component that can emit all possible non-satisfying assignments for the 3-SAT problem along with a special state outside of this component that can emit all binary strings, but that does so with smaller probability. Because this probability is small, the path consisting of repeatedly being in the special state is only designable if a specific sequence of 0's and 1's could not possibly be emitted by the component corresponding to the 3-SAT formula. And such a sequence is, by the construction, a satisfying assignment of the 3-SAT formula.

In full detail, the construction is as follows (see Figure 2 for an illustration). Assume the 3-SAT formula consists of m variables and r clauses. The HMM consists of a begin state B, two special states S and T and $r(m+1)$ states labelled $X_{i,j}$ where $1 \le i \le r$ and $1 \le j \le m+1$. The emission alphabet consists of 0, 1, and the special symbol #. The state B transitions to either S or any of $X_{i,1}$ with equal probability, $\frac{1}{r+1}$, while emitting #. The state S transitions to itself while emitting 0 or 1, each with probability $\frac{1}{2}$. The state T transitions to itself with probability 1 while emitting #. The r sets of states $X_{i,1}, \dots, X_{i,m+1}$ for $1 \le i \le r$ are arranged in independent chains, each corresponding to the ith clause, that emit all strings $\{0,1\}^m$ that do not satisfy the ith clause. Such a chain is constructed by the following: if the ith clause contains the jth variable un-negated then $X_{i,j}$ transitions to $X_{i,j+1}$ while emitting 0 with probabilty 1, if the ith clause contains the jth variable negated then $X_{i,j}$ transitions to $X_{i,j+1}$ while emitting 1 with probabilty 1, and if the ith clause doesn't contain the jth variable then $X_{i,j}$ transitions to $X_{i,j+1}$ while emitting 0 or 1 each with probability $\frac{1}{2}$. Finally, $X_{i,m+1}$ transitions to T while emitting # with probability 1.

The state-path to design is BS^{m+1}. We observe that the joint probability of this state-path and an emission sequence of the form $\#\{0,1\}^m$ is $(\frac{1}{r+1})(\frac{1}{2})^m$, and that only emissions of this form have non-zero probability for this state-path. We further observe that the only other state-path that could emit such a sequence must be of the form $BX_{i,1} \dots X_{i,m+1}$, and the joint probability of such a sequence and such a state-path is $(\frac{1}{r+1})(\frac{1}{2})^{m-3}$ if the emission sequence contains a # followed by a non-satisfying assignment to the 3-SAT formula, but the joint probability is zero if the emission sequence contains a # followed by a satisfying assignment. Since $(\frac{1}{r+1})(\frac{1}{2})^{m-3} > (\frac{1}{r+1})(\frac{1}{2})^{m}$, the only sequence that could design BS^{m+1} is a # followed by a satisfying assignment and therefore BS^{m+1} is designable if and only if there is a satisfying assignment to the 3-SAT formula.

The above construction is done in polynomial time, and therefore we have successfully given a polynomial reduction from 3-SAT to Designable. $\square$

Corollary 1. *Inverse-CKY is NP-hard.*

Proof. An HMM can be thought of as an SCFG with a non-terminal corresponding to each state and a terminal to each letter in the emission alphabet. Every branching rule rewrites a state as a letter and another state, so that all derivation trees are right-branching. Since the problem is hard on HMMs it is also hard on the extended class of SCFGs. $\square$

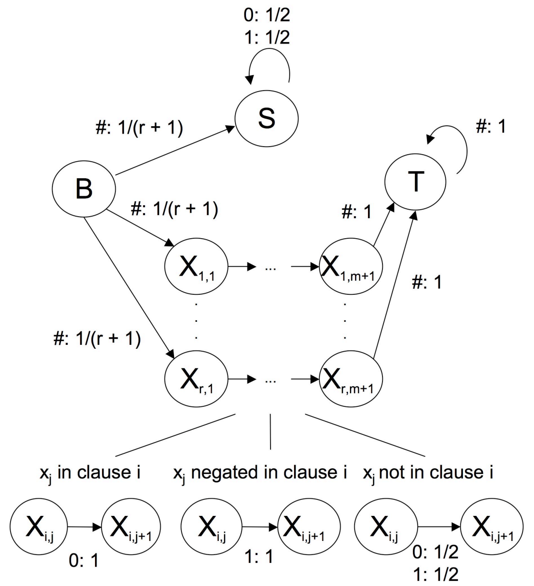

Figure 2. The reduction from 3-SAT to DESIGNABLE. Each transition is marked with all non-zero probability emissions followed by their corresponding probabilities.

3. Algorithmic Results

In this section, we give two approaches for finding a solution to the inverse problem, a branch-and-bound algorithm and a formulation of the problem as a Mixed Integer Linear Program. Both of these are derived from the same basic approach, based on a set of constraints we develop that are satisfied by an $\boldsymbol{\omega}$ if and only if it is a solution to the inverse problem. Below we first develop these constraints. Similar constraints and a Mixed Integer Linear Program can be developed for SCFGs. For reasons of space and simplicity of presentation, we only give the details for HMMs in this section. We illustrate the formulation of constraints and a Mixed Integer Linear Program for an SCFG used for RNA secondary structure prediction in a supplement.[2]

3.1. Constraint Formulation

Conceptually, the set of inequalities for HMMs is derived by looking at how the Viterbi algorithm works and enforcing constraints on $\boldsymbol{\omega}$ so that the Viterbi algorithm is forced to return the desired state-path $\boldsymbol{\pi_0}$.

The Viterbi algorithm calculates an n by K table of values $M_{i,s}$ of the best log-probability scores for the state-path from positions 1 to i with final state s. Because of the special form of the HMM score, this table can be filled in iteratively:
(1) $M_{1,S} = 0$ and $M_{1,s} = -\infty$ for all $s \neq S$
(2) $M_{i,s} = \max_{s'}(M_{i-1,s'} + q^{\omega_{i-1}}_{s',s})$ for $2 \leq i \leq n$ and all s

The best state in the nth position is then read off as $\pi_n \in \arg\max_s(M_{n,s})$, and the earlier ones are read off by a traceback routine: the best state in position $n-1$ is an s' that maximized $(M_{n-1,s'} + q^{\omega_{n-1}}_{s',\pi_n})$, and so on.

From the above we can directly read off the constraints on the emission symbol ω_i in position i for $1 \leq i \leq n-1$, that need to be satisfied in order to design a state-path with states π_i. For the Viterbi algorithm to return the desired path, we need for every state in this path to traceback to the previous state in the desired path and for the last state in this path to have the best log-probability score:
(3) $M_{i,\pi_i} + q^{\omega_i}_{\pi_i,\pi_{i+1}} \geq \max_{s\neq\pi_i}(M_{i,s} + q^{\omega_i}_{s,\pi_{i+1}})$ for $1 \leq i \leq n-1$
(4) $M_{n,\pi_n} \geq \max_{s\neq\pi_n}(M_{n,s})$

3.2. Branch-and-Bound Algorithm

What is particularly nice about inequalities (1)-(4) is that they allow for an inductive method for choosing possible ω_i in an emission sequence based only on the choices of ω_j for $1 \leq j \leq i-1$. This is because the inequality constraining the choice of ω_i (inequality 3 above) only depends on the values for $M_{i,s}$. And the

[2]See http://groups.csail.mit.edu/cb/inv_viterbi/scfg.pdf

values for $M_{i,s}$ only depend on the choices made for ω_1 through ω_{i-1}. This naturally leads to a branch-and-bound algorithm. Branch-and-bound algorithms are frequently useful in solving computationally hard problems. A branch-and-bound algorithm is complete (it always finds the correct answer) and frequently efficient on many problem instances.

The branch-and-bound algorithm steps through position i from 1 to $n-1$, at each step maintaining a list of emission sequences of length i that could be extended to possible length $n-1$ sequences the algorithm will ultimately return. At each step i, the algorithm forms emission sequences of length i from the emission sequences of length $i-1$ stored in the previous stage by appending possible emission symbols onto the sequences from the previous stage. In order to avoid performing an exhaustive search, at every stage the algorithm prunes the search space by applying two elimination rules. The first elimination rule ensures that for a given length $i-1$ sequence from the previous stage, an ω_i is only appended onto this sequence to form a length i sequence if the traceback constraint (constraint 3) is satisfied by the choice ω_i. The second elimination rule examines pairs $\boldsymbol{\omega}$ and $\tilde{\boldsymbol{\omega}}$ of partial strings of length i that remain after the application of the first elimination rule. It eliminates $\boldsymbol{\omega}$ due to $\tilde{\boldsymbol{\omega}}$, if given that $\boldsymbol{\omega}$ can be extended to a solution to the design problem, then $\tilde{\boldsymbol{\omega}}$ must also be able to be extended to a solution.

Specifically, the second elimination rule is based on the following observation. If for all states s, $M_{i+1,\pi_{i+1}} - M_{i+1,s}$ is at least as large under $\tilde{\boldsymbol{\omega}}$ as it is under $\boldsymbol{\omega}$ (i.e. if for all states s, the relative preference of $\tilde{\boldsymbol{\omega}}$ for π_i to state s is at least as large as that of $\boldsymbol{\omega}$), then the traceback constraints (inequality 3 above) on all positions j for $j > i$ and the ending constraints (inequality 4 above) can only be easier to satisfy when extending $\tilde{\boldsymbol{\omega}}$ than when extending $\boldsymbol{\omega}$.

It is important to note that for the case of a 2-state HMM the branch-and-bound is an exact polynomial-time algorithm. This is because there is only one $M_{i+1,\pi_{i+1}} - M_{i+1,s}$ value to compare the choices for ω_i on (there is only one state s other than π_i at every position since there are only 2 states to choose from), and so there is always a *best* choice for ω_i at every position based on the past choices.

The above branch-and-bound algorithm is exact for all HMMs, but has no guaranteed worst-case running time. If we make additional assumptions about our HMM, however, we can show that the algorithm also has fixed-parameter tractable running time. Specifically, we assume that all q values (the log-probabilities) satisfy $q \geq -B$ and that there are no zero probabilities in the model. Furthermore, we assume that these q values have been rounded off to precision δ.

Algorithm 1 Branch-and-Bound Algorithm

```
**Input:** An HMM, a desired state-path $\boldsymbol{\pi_0}$ of length $n$, and for every position $i$ in $1,\dots,n$ a set $\Sigma_i \subseteq \Sigma$ giving the allowed emissions at position $i$
**Output:** A sequence $\boldsymbol{\omega}$ such that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi},\boldsymbol{\omega})$ or $\emptyset$ if no such sequence exists.
**Variables:** A list $L_i$ of all partial sequences of length $i$ considered at the $i$th iteration each together with its corresponding $K$-vector of values $M_{i,s}$.
**Initialize:** $L_0 = \{(\epsilon, \mathbf{0})\}$
**for** $i = 1$ **to** $n-1$ **do**
  Set $L_i = \emptyset$
  **for** all $(\boldsymbol{\omega^{i-1}}, \boldsymbol{v^{i-1}}) \in L_{i-1}$ and all $\omega_i \in \Sigma_i$ **do**
    Form $\boldsymbol{\omega^i} = \boldsymbol{\omega^{i-1}}\omega_i$ by concatenation
    Compute the $K$-vector $\boldsymbol{v^i}$ of values $M_{i+1,s}$
    Add $(\boldsymbol{\omega^i}, \boldsymbol{v^i})$ to $L_i$ iff Elim Rule 1 doesn't apply
  **end for**
  **for** all $(\boldsymbol{\omega^i}, \boldsymbol{v^i}) \in L_i$ **do**
    From $\boldsymbol{v^i}$ compute and store the $(K-1)$-vector $\boldsymbol{u}$ of values $M_{i+1,\pi_{i+1}} - M_{i+1,s}$ for $s \neq \pi_{i+1}$
  **end for**
  Apply Elim Rule 2 to all pairs of entries of $L_i$
**end for**
**for** all $(\boldsymbol{\omega^{n-1}}, \boldsymbol{v^{n-1}}) \in L_{n-1}$ **do**
  **if** $M_{n,\pi_n} < \max_{s\neq\pi_n}(M_{n,s})$ **then**
    Remove $(\boldsymbol{\omega^{n-1}}, \boldsymbol{v^{n-1}})$ from $L_{n-1}$
  **end if**
**end for**
**Return:** An element of $L_{n-1}$ or $\emptyset$ if $L_{n-1}$ is empty.
**Elim Rule 1:** Eliminate $\boldsymbol{\omega^i}$ if $M_{i,\pi_i} + q^{\omega_i}_{\pi_i,\pi_{i+1}} < \max_{s\neq\pi_i}(M_{i,s} + q^{\omega_i}_{s,\pi_{i+1}})$
**Elim Rule 2:** Eliminate $\boldsymbol{\omega^i}$ due to $\tilde{\boldsymbol{\omega}}^{\boldsymbol{i}}$ if $\tilde{\boldsymbol{\omega}}^{\boldsymbol{i}} \in L_i$ has $(K-1)$-vector $\boldsymbol{u}$ componentwise $\geq$ that of $\boldsymbol{w^i}$
```

Under these assumptions, we can see that any two values $M_{i,s}$ and $M_{i,s'}$ satisfy $|M_{i,s} - M_{i,s'}| \leq B$. This follows from the definitions:
$M_{i,s} = \max_{s'}(M_{i-1,s'} + q^{\omega_{i-1}}_{s',s})$ and
$M_{i,s'} = \max_{s}(M_{i-1,s} + q^{\omega_{i-1}}_{s,s'})$.
Let the maximum in the expression for $M_{i,s}$ be attained with s_0. Then

$$M_{i,s'} \geq M_{i-1,s_0} + q^{\omega_{i-1}}_{s_0,s'}$$

$$= M_{i-1,s_0} + q^{\omega_{i-1}}_{s_0,s} + (q^{\omega_{i-1}}_{s_0,s'} - q^{\omega_{i-1}}_{s_0,s})$$

$$= M_{i,s} + (q^{\omega_{i-1}}_{s_0,s'} - q^{\omega_{i-1}}_{s_0,s}),$$

so that, upon rearranging,

$$M_{i,s} - M_{i,s'} \leq q^{\omega_{i-1}}_{s_0,s} - q^{\omega_{i-1}}_{s_0,s'} \leq 0 - (-B) = B,$$

and by symmetry, we also get $M_{i,s'} - M_{i,s} \leq B$, so finally, $|M_{i,s} - M_{i,s'}| \leq B$.

In particular, only $2B/\delta$ distinct values are possible for each of the $(K-1)$ possible $M_{i,\pi_i} - M_{i,s}$ values. In the branch-and-bound algorithm, it is only impossible to remove either $\boldsymbol{\omega}$ or $\tilde{\boldsymbol{\omega}}$ (both of length i) due to the other if they are incomparable: the values one gives for $M_{i,\pi_i} - M_{i,s}$ are larger for some s and smaller for some other s. But there are only $(2B/\delta)^{K-2}$ incomparable values: for two sequences that share the first $(K-2)$ $M_{i,\pi_i} - M_{i,s}$ values, any values for the last $M_{i,\pi_i} - M_{i,s}$ will make them comparable.

Therefore, in the branch-and-bound algorithm there are at most $(2B/\delta)^{K-2}$ sequence possibilities that must be retained at any stage, and so with a careful implementation the running time of the algorithm is $O((2B/\delta)^{K-2} nK^2|\Sigma|)$. This bound is exponential in the number of states, but linear in the length of the structure to be designed. (This bound is independent of the base used to get the q values (log-probabilities), because changing the base introduces a factor into both B and δ that cancels.)

For SCFGs in CNF, a similar idea allows one to obtain an exact algorithm that runs in polynomial time if there are only 2 non-terminal symbols. However, the idea used above for candidate string elimination does not immediately generalize to SCFGs because of their non-linear nature; an HMM outputs one symbol per state, but a non-terminal in an SCFG can generally end up producing any substring of the output string.

3.3. Casting the Problem as a Mixed Integer Linear Program

We can also start with the inequalities that must be satisfied for $\boldsymbol{\omega}$ and cast the inverse problem as the problem of finding a feasible solution to a Mixed Integer Linear Program. We provide this simple formulation because it allows both practical and theoretical tools developed for integer programming to be applied directly to our problem.

The formulation as a Mixed Integer Linear Program is done by defining 0-1 variables $\epsilon_{i,j}$, where $\epsilon_{i,j} = 1$ indicates that the jth emission symbol is chosen for ω_i. Enforcing that there is only one emission choice made at every position is equivalent to requiring $\sum_j \epsilon_{i,j} = 1$ for $i = 1$ to $n-1$. Each maximum in the constraints is replaced by $\geq$, while the traceback constraints are enforced by additional equalities.

Integer Linear Program For HMMs:
Objective: Feasible Solution

Variables:
$\epsilon_{i,j}$, 0-1 valued, for $1 \leq i \leq n-1$ and $1 \leq j \leq |\Sigma|$
$M_{i,s}$, for $1 \leq i \leq n$ and $1 \leq s \leq K$

Constraints:
$\sum_j \epsilon_{i,j} = 1$ for all $1 \leq i \leq n-1$
$\epsilon_{i,j} = 0$ if $j \notin \Sigma_i$ for all $1 \leq i \leq n-1$
$M_{1,S} = 0$ and $M_{1,s} = -\infty$ for all $s \neq S$
$M_{i,s} \geq \sum_j \epsilon_{i-1,j}(M_{i-1,s'} + q^j_{s',s})$ for all s, s' and all $i \geq 2$
$M_{i,\pi_i} = \sum_j \epsilon_{i-1,j}(M_{i-1,\pi_{i-1}} + q^j_{\pi_{i-1},\pi_i})$ for all $i \geq 2$
$M_{n,s} \leq M_{n,\pi_n}$ for all $s \neq \pi_n$

4. Simulations

We implemented our branch-and-bound algorithm and examined its running time on synthetic data in order to demonstrate that in practice the algorithm frequently runs fast when an exhaustive search would be infeasible. In order to do this, we randomly generated HMMs by drawing each-transition-emission pair probability from the uniform distribution and then normalizing the values, rounding off to precision $\delta = 0.01$. We then separately generated both arbitrary state-paths and designable state-paths at random from this HMM (the latter by randomly sampling emission sequences and running the Viterbi algorithm on these sequences) and timed our branch-and-bound algorithm on these instances. We found that our algorithm ran significantly faster on arbitrary paths, the majority of which are not designable, than on arbitrary designable paths (taking milliseconds rather than seconds per run).

Figure 3 shows running times of simulations on random designable state-paths for different numbers of states K and path lengths n, with fixed emission alphabet of size $|\Sigma| = 20$. For each pair of K and n values, 10 HMMs were generated at random and for each of these HMMs, 10 designable paths were generated at random, as described above. The branch-and-bound algorithm was then run and the average time to design a sequence over these 100 runs was recorded. On these problem instances, the running time of the algorithm scales roughly linearly with path length n. Interestingly, while the running times initially increased with increasing K values, the running times were lower for $K = 50$ and $K = 100$ than for $K = 20$, an observation that was repeated for multiple experiments. The longest run of the algorithm took 80 seconds. A solution by exhaustive search would require examining $|\Sigma|^n$ possible sequences, which for that run would have been 20^{400} sequences. All code was implemented in Matlab and run on a 3.06 GHz Intel Xeon PC.

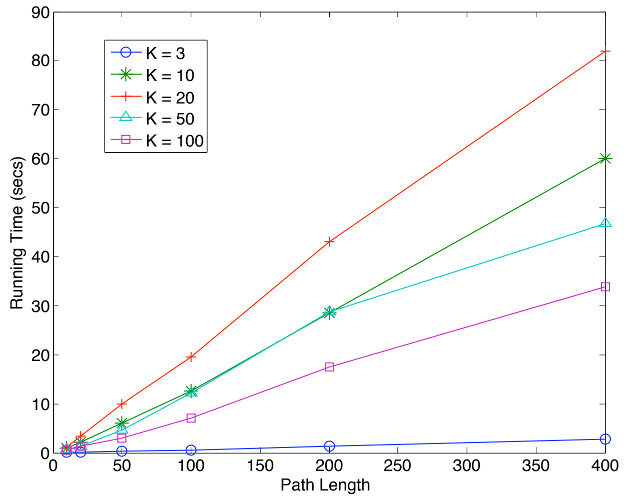

Figure 3. Running times of the branch-and-bound algorithm on designable paths. Simulations shown for number of states $K = 3, 10, 20, 50, 100$, path lengths $n = 10, 20, 50, 100, 200, 400$ and emission alphabet size $|\Sigma| = 20$.

5. Conclusions

We have introduced a novel problem on HMMs and SCFGs, the Inverse-Viterbi problem, inspired by protein and RNA structure design, and have given a number of theoretical results for the problem. In particular, our hardness result demonstrates that a polynomial time algorithm for RNA secondary structure design that only exploits the general form of the Zuker energy or similar SCFG models (and not the particulars of a specific model) is not possible unless $P = NP$.

There are a number of possible extensions to this work. Developing more efficient algorithms on both HMMs and SCFGs may be possible and in particular, extending our branch-and-bound algorithm to SCFGs would be useful. It is also possible to explore extensions of the problem to more general probabilistic models such as Markov Random Fields. The framework given here may also be useful for developing new algorithms for design in specific applications. Areas where the negative-design aspect plays a large role, such as RNA secondary structure design, are the most likely candidates to benefit from such an approach. Given the widespread use of grammars, the inverse problem we have defined here may find applications to other fields.

6. Acknowledgements

The authors thank Andreas Schulz, Michael Baym, Jérôme Waldispühl, Mathieu Blanchette, and particularly Michael Collins for advice and helpful discussions. M. Schnall-Levin is supported by NDSEG and Hertz Foundation fellowships and L. Chindelevitch is supported in part by an NSERC PGS-D Scholarship.

References

Andronescu, M. e. a. (2004). A new algorithm for RNA secondary structure design. *Journal of Molecular Biology*, *336*, 607–624.

Breaker, R. R. (1996). Are engineered proteins getting competition from RNA? *Current Opinion in Biotechnology*, *7*, 442–448.

Busch, A., & Backofen, R. (2006). INFO-RNA- a fast approach to inverse RNA folding. *Bioinformatics*, *22*, 1823–1831.

Butterfoss, G., & Kuhlman, B. (2005). Computer-based design of novel protein structures. *Annual Review of Biophysics and Biomolecular Structure*, *35*, 49–65.

Dowell, R., & Eddy, S. (2004). Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, *5*.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1999). *Biological sequence analysis: Probablistic models of proteins and nucleic acids*. Cambridge University Press.

Elizalde, S., & Woods, K. (2006). Bounds on the number of inference functions of a graphical model. *ArXiv Mathematics e-prints*, *math/0610233*.

Hofacker, I. L. e. a. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte f. Chemie*, *125*, 167–188.

Park, S., Yang, X., & Saven, J. G. (2004). Advances in computational protein design. *Current Opinion in Structural Biology*, *14*, 487–494.

Pokala, N., & M., H. T. (2001). Review: Protein design- where we were, where we are, where we're going. *Journal of Structural Biology*, *134*, 269–281.

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

Zuker, M., & Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, *9*, 133–148.
]