BIOINFORMATICS

Vol. 19 Suppl. 2 2003, pages ii36–ii41
DOI: 10.1093/bioinformatics/btg1057

HMM sampling and applications to gene finding and alternative splicing

Simon L. Cawley[1] *and Lior Pachter*[2,*]

[1]*Affymetrix, 6550 Vallejo St Suite 100, Emeryville, CA 94608, USA and* [2]*Department of Mathematics, U.C. Berkeley, CA 94720, USA*

Received on March 17, 2003; accepted on June 9, 2003

ABSTRACT

The standard method of applying hidden Markov models to biological problems is to find a Viterbi (maximal weight) path through the HMM graph. The Viterbi algorithm reduces the problem of finding the most likely hidden state sequence that explains given observations, to a dynamic programming problem for corresponding directed acyclic graphs. For example, in the gene finding application, the HMM is used to find the most likely underlying gene structure given a DNA sequence. In this note we discuss the applications of sampling methods for HMMs. The standard sampling algorithm for HMMs is a variant of the common forward-backward and backtrack algorithms, and has already been applied in the context of Gibbs sampling methods. Nevetheless, the practice of sampling state paths from HMMs does not seem to have been widely adopted, and important applications have been overlooked. We show how sampling can be used for finding alternative splicings for genes, including alternative splicings that are conserved between genes from related organisms. We also show how sampling from the posterior distribution is a natural way to compute probabilities for predicted exons and gene structures being correct under the assumed model. Finally, we describe a new memory efficient sampling algorithm for certain classes of HMMs which provides a practical *sampling* alternative to the Hirschberg algorithm for optimal alignment. The ideas presented have applications not only to gene finding and HMMs but more generally to stochastic context free grammars and RNA structure prediction.

Key words: suboptimal parses, sampling, hidden Markov model, conserved alternative splicing

Contact: lpachter@math.berkeley.edu

INTRODUCTION

Many problems in computational biology reduce to the prediction of an optimal 'annotation' of one or more biological sequences. An annotation could be a gene prediction for a DNA sequence, an assignment of secondary structure to a protein sequence, or in the case of multiple sequences it could be an alignment. In typical annotation problems an underlying model of the problem (with associated parameters) is developed, and it is in this context that one searches for an optimal solution. For example, in the case of alignment the model may be the edit distance model whose parameters are match, mismatch and gap scores. For gene finding, typical parameters include probabilities of different codons. The optimization algorithms that have been developed for these models usually return one optimal solution, and this solution may depend heavily on the particular parameters that are used. In the case of pairwise sequence alignment, there has been extensive work on understanding the dependence of optimal alignments on the parameters (Gusfield *et al.*, 1992) and it has been observed that there may be many biologically significant alignments suboptimal for a given parameter set.

These observations have led to a numerous efforts for finding suboptimal solutions to optimization problems in biology. The most studied problem has been alignment, where investigations by numerous authors (Naor and Brutlag, 1993, 1994; Saqi and Sternberg, 1991; Saqi *et al.*, 1992; Thiele *et al.*, 1995; Waterman, 1983; Waterman and Eggert, 1987; Waterman and Byers, 1985; Waterman *et al.*, 1992; Waterman, 1994; Zuker, 1991) have been extended and varied in many interesting ways. Alignment algorithms such as Needleman-Wunsch (Needleman and Wunsch, 1970) are based on dynamic programming, and this classical subject also contains a large literature on finding suboptimal solutions (Bellman and Kalaba, 1960; Clarke *et al.*, 1963; Fox, 1973; Hoffman and Pavley, 1959; Lawler, 1972; Perko, 1986; Pollack, 1961 and Shier, 1979).

Many dynamic programming alignment solutions along with the majority of current gene finding predictions can be interpreted as Viterbi solutions for a suitable HMM. In the hidden Markov model framework one considers an HMM with parameters θ. The observed sequence y can be either a DNA sequence (in the case

*To whom correspondence should be addressed.

of gene finding), or pairs of sequences in the case of alignment. Either way, a hidden state path z for the HMM corresponds to an annotation or alignment (z corresponds to the missing data). The mathematical problem is to solve the maximum likelihood problem: Given θ and y, find z maximizing $P(y|z,\theta)$. This problem can be solved efficiently with dynamic programming, and in the context of HMMs is called the Viterbi algorithm.

A different strategy for obtaining state paths is to **sample** from the distribution $p(z|y,\theta)$. This viewpoint is natural from the perspective of Gibbs sampling, where such a sampling step is a basic part of the algorithm (e.g. Eddy, 1995; Durbin *et al.*, 1998). This point of view leads us to a randomized rather than deterministic approach to finding suboptimal solutions for dynamic programming applications in biology. Our method is fast and can therefore be applied in practice to large biological sequences, and has the additional benefit that it allows for the exploration of the space of solutions in such a way that posterior probabilities can be assigned to annotations in cases where the underlying models are probabilistic.

We also highlight a number of applications of the idea to problems where hidden Markov models or stochastic context free grammars have been applied. In particular, we have implemented the method for the SLAM generalized pair hidden Markov model (Pachter *et al.*, 2001), which then allows us to sample for alternatively spliced gene structures that are the same in human and mouse syntenic sequences. We have found a few examples where alternatively spliced human gene structures exist in the orthologous mouse genes, and we conjecture that this is true for the many alternatively spliced human and mouse genes.

SAMPLING ALGORITHMS

The sampling algorithms we describe are most conveniently developed in the framework of graph theory, where our method can be seen as a randomized algorithm for sampling paths according to their weight in a directed acyclic graph.

The forward-backtrack algorithm

The following lemma appears in Zhu *et al.* (1998) and is mentioned in Durbin *et al.* (1998).

LEMMA 1. *Let G be a directed acyclic graph with source s and sink t. Let each edge $e = (v_i, v_j)$ of G have weight $w(e)$ (we also use the notation $w(e) = w(v_i, v_j)$), and each node have weight $w(v_i)$. Assume without loss of generality that*

$$\sum_{\text{paths } P=(v_1,\ldots,v_{k(P)})} w(v_1)\prod_{i=2}^{k(P)} w(v_{i-1},v_i)w(v_i) = 1.$$

It is possible to pick at random a path P consisting of $s = v_1, v_2, \ldots, v_{k(P)} = t$ in time $O(n)$ such that

$$Pr(picking P) = w(s)\prod_{i=2}^{k(P)} w(v_{i-1},v_i)w(v_i).$$

PROOF. The proof is by induction on the maximal length of a path between s and t. The base case $n = 1$ is trivial. Suppose the lemma is true for the case when the length of the longest path between s and t is n, and consider the case where the maximal path has length $n+1$. Suppose the edges out of s have weights $w(e_1), \ldots, w(e_k)$, and are adjacent to vertices $v_1, \ldots, v_k$ respectively. Suppose we have computed weights $\beta(v_i)$ for all the vertices adjacent to s, where $\beta(v_i)$ is the sum of the weight of all the paths from v_i to t (this step can be done using dynamic programming, and is the backward algorithm for HMMs). Our path picking algorithm is to pick an edge from s at random with probability $w(s)w(e_i)\beta(v_i)$, at which point the distance from v_i to t is at most n, so by induction we can choose from there a path P which has weight z, and which has been selected with probability $\frac{z}{\beta(v_i)}$. Observe that the weight of the path from s to t is $w(s)zw(e_i)$, and that it has been selected with probability $\frac{z}{\beta(v_i)}w(s)w(e_i)\beta(v_i) = w(s)zw(e_i)$.

The interpretation of this result for HMMs is as follows.

COROLLARY 1. *Given an HMM and an associated output sequence Y_1^T, it is possible to pick a sequence of hidden states (from which the sequence could have been produced) at random in time $O(T)$ so that the probability of picking the sequence of states X_1^L with durations d_1^L is equal to $Pr(X_1^L, d_1^L|Y_1^T)$.*

In this case the sampling algorithm works by first applying the backward algorithm to compute the β variables, and then forward-tracking. The algorithm can also be applied in the forward direction by computing the forward variables first, and then back-tracking through the HMM graph.

Thus, the sampling algorithm is essentially a probabilistic back-track for HMMs, in contrast to the optimal backtrack in the Viterbi algorithm. Similarly, it is a probabilistic back-track for SCFGs that replaces the optimal backtrack in the CYK algorithm.

Memory efficient sampling

Memory efficient divide-and-conquer variants of the Viterbi algorithm (Hirschberg, 1975) have been critical in applications such as alignment, or more generally for pairHMMs. In the case of a standard (non-pair) HMMs with N states for a sequence of length T, Hirschberg's algorithm reduces the memory requirements from $O(NT)$ to $O(N)$ and increases the running time from $O(NT)$ to

$O(NTlogT)$. For pairHMMs (using T and U to denote the lengths of two sequences), Hirschberg's algorithm reduces the memory requirements from $O(NTU)$ to $O(NT)$ while leaving the time requirements at $O(NTU)$.

The forward-backtrack sampling algorithm described above for a standard HMM has time complexity $O(NT + kT)$ where k is the number of samples we wish to obtain. The memory requirements are $O(NT)$, consisting of the forward variables. An analog of Hirschberg's algorithm for sampling can be obtained by *sampling* during the divide step, rather than maximizing. If we denote the forward variables by $\alpha(t, i)$ and the backward variables by $\beta(t, i)$, then we sample t according to the weights $\alpha(t, i)+\beta(t, i)$. A divide-and-conquer analog for sampling can also be obtained for pairHMMs using the same idea.

Unfortunately, a drawback of the algorithm is that repeated sampling requires the re-calculation of the α and β variables, since these are discarded to save memory during the divide and conquer algorithm. Thus, obtaining k samples requires only $O(N)$ memory, but $O(kNTlogT)$ time. We therefore introduce an alternative memory efficient approach for sampling which is as memory efficient as divide-and-conquer, yet allows for efficient resampling as well. The algorithm is most suited for pairHMMs where the reduction in time for resampling is larger, but for the sake of simplicity we explain the method for a non-pair HMM (although it can be generalized).

We assume that the generalized HMM has a probability $l_d(i)$ of outputting d symbols in state i and that $d \leq D$. The algorithm begins with the computation of the forward variables one 'column' at a time, until the last column $\alpha(T, i)$ is reached. At this stage we have discarded all but the previous $D-1$ columns, and so the calculation of $\alpha(T, i)$ requires only $O(N)$ space. We now backtrack and produce k samples at the same time. Consider the problem of constructing $\alpha(t-D, i)$ (for every i) given $\alpha(u, i)$ (for $t-D+1 \leq u \leq t$ and $1 \leq i \leq N$). The forward variable recursions allow us to write

$$\alpha(t, j) = b_j(t) \sum_{d=1}^{D} \sum_{i=1}^{N} l_d(i) c_{ij} \alpha(t-d, i)$$

where the c_{ij} are transition probabilities and $b_j(t)$ is the probability of outputting the symbol at t in state j. Let $v_r = (\alpha(r, 1), \ldots, \alpha(r, N))$ denote the vector of α variables at position r. Solving for v_{t-D} we obtain

$$v_{t-D} = C^{-1}B^{-1}v_t - \sum_{d=1}^{D-1} L_d v_{t-d}$$

where C is the transition matrix $\{c_{ij}\}$, B is the diagonal matrix with $(b_1(t), \ldots, b_N(t))$ on the diagonal and L_d is the diagonal matrix with $(l_d(1), \ldots, l_d(N))$ on the diagonal. The matrix B is invertible as long as every symbol has a nonzero output probability in every state. The matrix C is a stochastic matrix, and may or may not be invertible. We do not know of general conditions for a stochastic matrix to be invertible, but sufficient conditions exist, for example.

THEOREM 2. *A matrix C is invertible if C is strictly diagonally dominant, that is if*

$$|c_{ii}| > \sum_{i \neq j} |c_{ij}|$$

for every i.

This follows immediately from Gersgorin's theorem (e.g. Horn and Johnson, 1985).

COROLLARY 3. *A stochastic matrix is invertible if all the self transition probabilities are greater than a half.*

This is a useful condition, valid in many practical situations of interest.

Having calculated $v_{t-D} = (\alpha(t-D, 1), \ldots, \alpha(t-D, N))$ we then sample one step for each of k paths using the method of Lemma 1.

The algorithm runs in time $O(kNT + MT)$ (where M is the time for an $N \times N$ matrix inversion) and requires $O(N)$ memory. In the case of pairHMMs, our sampling algorithm runs in time $O(NTU + kN(T+U))$ and memory $O(N(T+U))$ rather than time $O(kNTU)$ for the sampling analog of the Hirschberg algorithm. The specific case of Needleman-Wunsch alignment can always be sampled efficiently (unless the gap score is 0) since the matrices that need to be inverted are nonsingular. Because of the immediate practical applications of memory efficient alignment sampling we provide full details of the algorithm in the appendix.

CONSERVED ALTERNATIVE SPLICING

The problem of identifying alternative splicing structures for genes is a particularly suitable application of HMM sampling. The sampling approach can be thought of as an ab-initio method for finding alternative splicings of genes, that is independent of EST evidence. Alternatively, sampling can be used with HMM models that incorporate EST evidence (e.g. `GENIE` Reese *et al.*, 2000).

We used sampling in an application of the SLAM generalized pairHMM (Alexandersson *et al.*, 2003) to the beta-tropomyosin (TPM2) gene in human and mouse. SLAM models exon/intron structure and conserved noncoding sequences in a pair of syntenic DNA sequences and is well-suited for predicting protein-coding genes in the human and mouse genomes. The model performs both alignment and genefinding at the same time, and when an ORF is found it is found in both organisms. TPM2 was selected because it has two well-characterized

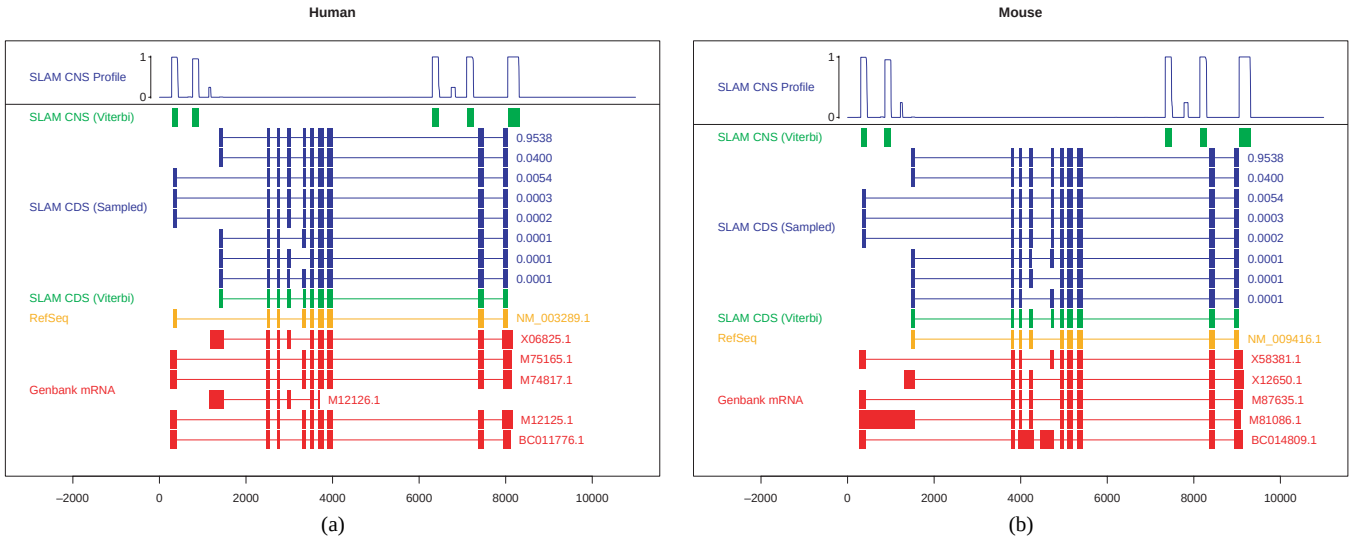

Fig. 1. Gene structure of the beta-tropomyosin (TPM2) gene in human (a) and mouse (b). 10,000 parses were sampled, sampled results are presented in blue. The CNS profile shows the frequency at which each base is predicted to be a member of a CNS over the sampled parses. 8 distinct CDS predictions were generated in the course of the samples, each is shown along with its frequency. The CDS and CNS predictions for the Viterbi parse are presented in green. Experimentally-derived transcript structures (along with their identifiers) are presented in the Genbank mRNA and RefSeq tracks.

alternative transcripts and the pattern of alternative splicing is conserved in both human and mouse. It is difficult enough to find well-annotated alternatively spliced transcripts in human, requiring also that there be alternative splicing in mouse greatly limits the number of choices. Genomic DNA for the TPM2 locus was obtained for human (chr9:35993981-36002362 on the June 2002 human assembly at UCSC) and for mouse (chr4:42348486-42357821 on the Feb 2002 mouse assembly at UCSC). SLAM was used to predict an optimal paired gene structure with the Viterbi algorithm and to predict 10,000 sampled paired parses with the sampling method of Lemma 1 (Fig. 1). The Viterbi parse does fairly well, but tries to predict both the 6th and 7th exons (and in trying to do so, is forced to shift one of the exon boundaries) whereas according to RefSeq and Genbank mRNA annotations only one of the two exons should be used. As expected, the most likely sampled parse is in fact the same as the Viterbi parse (occuring at a frequency of greater than 95%). Looking through the other sampled parses it is clear that the sampling provides valuable information beyond that provided by the Viterbi parse. The samples include variants where each of the 6th and 7th exons are left out, the terminal alternative exon is included in some of the samples, and one of two splice variants is recovered exactly.

In addition to predicting coding sequence (CDS) SLAM predicts conserved non-coding sequence (CNS), regions of sequence having significant conservation but poor coding statistics. Often well-conserved untranslated regions end up being predicted as CNS. Each sampled parse may yield CNS regions, and the frequency of a particular base being predicted as CNS can be used as a measure of confidence in the base being a CNS. Not surprisingly, the CNS regions predicted as part of the Viterbi parse are frequently sampled as such, but interestingly the sampled parses suggest an extra CNS in the second exon of the gene.

DISCUSSION

The sampling algorithm for HMMs is particularly important in the context of biology where suboptimal parses may be of interest, and where posterior probabilities are important for obtaining confidence measures for biological predictions. The speed of the sampling algorithm (indeed, it is as fast as the backtrack algorithm for the Viterbi), combined with its low memory cost, makes it a practical method for any type of HMM where the Viterbi is being used. We feel that the sampling algorithm should be standard in HMM implementations where currently only the forward-backward and Viterbi algorithms are typically implemented.

In the case of alternative splicing, the method provides a natural algorithm for predicting different transcripts in a gene region. Suboptimal methods such as finding the k best parses significantly increase memory requirements.

Our analysis of alternative splicings that are conserved in human and mouse genes suggest that the mouse genome may provide a new window to finding alternative splicings in the human genome (and vice versa). With the recent publication of the mouse genome (Waterston *et al.*, 2002) and associated RIKEN cDNAs (Okazaki *et al.*, 2002) it should be possible to analyze the extent of conservation in alternative splicing on a genome wide scale. Preliminary analysis of human and mouse ESTs has been inconclusive as to the extent of conservation of alternative splicing (Ewan Birney, personal communication). Our de-novo approach does not take advantage of EST information, however it is an interesting problem to incorporate EST evidence into our framework. The de-novo approach is interesting in its own right since it may allow for the detection of rare (alternative) splice forms not obvious from EST alignments alone.

ACKNOWLEDGMENTS

The authors would like to thank Michael Siani-Rose for useful discussions. This work was partially supported by NIH grant R01-HG02362-01.

REFERENCES

Alexandersson,M., Cawley,S. and Pachter,L. (2003) SLAM - Cross-species genefinding and alignment wihth a generalized pair hidden Markov model. *Genome Res.*, **13**, 496–502.

Bellman,R. and Kalaba,R. (1960) On *K*th best policies. *J. SIAM*, **8(4)**, 582–588.

Clarke,S., Krikorian,A. and Rausen,J. (1963) Computing the *N* best loopless paths in a network. *J. SIAM.*, **11(4)**, 1096–1102.

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis*. Cambridge UP.

Eddy,S. R. (1995) Multiple alignment using hidden Markov models. In *ISMB-95*. **3**, pp. 114–120.

Fox,B. (1973) Calculating the Kth shortest paths. *Can. J. Oper. Res. Inf. Process.*, **11**, 66–70.

Gusfield,D., Balasubramanian,K. and Naor,D. (1992) Parametric optimization of sequence alignment. In *Proceedings of the third annual ACM-SIAM Joint Symposium Discrete Algorithms*. Orlando Florida.

Hirschberg,D. S. (1975) A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, **18(6)**, 341–343.

Hoffman,W. and Pavley,R. (1959) A method for the solution of the *N*th best path problem. *J. ACM*, **6**, 506–514.

Horn,R.A. and Johnson,C.R. (1985) *Matrix Analysis*. Cambridge Universtiy Press, Cambridge.

Kan,Z., Rouchka,E.C., Gis,W.R. and States,D.J. (2001) Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. *Genome Res.*, **11**, 889–900.

Lawler,E. L. (1972) A Procedure for computing the *K*-best solutions to discrete optimization problems and its appplications to the shortest paths problem. *Manage. Sci.*, **18**, 401–405.

Modrek,B. and Lee,C. (2002) A genomic view of alternative splicing. *Nat. Genet.*, **30**, 13–19.

Naor,D. and Brutlag,D.L. (1993) On suboptimal alignments of biological sequences. In *Fourth International Symposium on Combinatorial Pattern Matching*. Springer-Verlag, Padova, Italy, pp. 179–196.

Naor,D. and Brutlag,D. L. (1994) On near-optimal alignments of biological sequences. *J. Comput. Biol.*, **1(4)**, 349–366.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol*, **48**, 443–453.

Okazaki,Y. *et al.* (2002) Analysis of the mouse transcriptome based on functional annotation of 60 770 full-length cDNAs. *Nature*, **420**, 563–573.

Pachter,L., Alexandersson,M. and Cawley,S. (2001) In *RECOMB 2001: Proceedings of the Fifth International Conference on Computational Molecular Biology*.

Perko,A. (1986) Implementation of algorithms for K shortest loopless paths. *Networks*, **16**, 149–160.

Pollack,M. (1961) The kth best route through a network. *Oper. Res.*, **9(4)**, 578–580.

Reese,M.G., Kulp,D., Tammana,H. and Haussler,D. (2000) `Genie`—gene finding in *Drosophila melanogaster*. *Genome Res.*, **10(4)**, 529–538.

Saqi,M.A.S. and Sternberg,M. (1991) A simple method to generate non-trivial alternate alignments of protein sequences. *J. Mol. Biol.*, **219**, 727–732.

Saqi,M.A.S., Bates,P. and Sternberg,M.J.E. (1992) Towards an automatic method of predicting protein structure by homology: an evaluation of suboptimal sequence alignments. *Protein Eng.*, **5(4)**, 305–311.

Shier,D.R. (1979) On algorithms for finding the *K* shortest paths in a network. *Networks*, **9**, 195–214.

Thiele,R., Zimmer,R. and Lengauer,T. (1995) Recursive dynamic programming for adaptive sequence and structure alignment. In *ISMB-95*. **3**, pp. 384–392.

Vingron,M. and Argos,P. (1990) Determination of reliable regions in protein sequence alignments. *Protein Eng.*, **3**, 565–569.

Waterman,M.S. (1983) Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proc. Natl Acad. Sci. USA*, **80**, 3123–3124.

Waterman,M.S. and Eggert,M. (1987) A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol.*, **197**, 723–728.

Waterman,M.S. and Byers,T.H. (1985) A dynamic programming algorithm to find all solutions in a neighborhood of the optimum. *Math. Biosci.*, **77**, 179–188.

Waterman,M.S., Eggert,M. and Lander,E. (1992) Parametric sequence comparisons. *Proc. Natl Acad. Sci. USA*, **89(13)**, 6090–6093.

Waterman,M.S. (1994) Parametric and ensemble sequence alignment algorithms. *Bull. Math. Biol.*, **56(4)**, 743–767.

Waterston,R.H. *et al.* (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature*, **420**, 520–562.

Zhu,J., Liu,J.S. and Lawrence,C.E. (1998) Bayesian adaptive sequence alignment algorithms. *Bioinformatics*, **14**, 25–39.

Zuker,M. (1991) Suboptimal sequence alignment in molecular biology: alignment with error analysis. *J. Mol. Biol.*, **221**, 403–420.

APPENDIX: MEMORY EFFICIENT SAMPLING OF ALIGNMENTS

Consider the alignment of two sequences of lengths T and U, with a gap score of g and a match/mismatch score of m_{ij} for element i from the first sequence with element j from the second ($1 \leq i \leq T, 1 \leq j \leq U$). In order to simplify the presentation, we will work with exponentiated scores $w = e^g$ and $s_{ij} = e^{m_{ij}}$. The weight of an alignment path is therefore just the product of the match, mismatch and gap scores along the path.

Let a_{ij} denote the the sum of the weights of all the alignment paths from $(0,0)$ to (i,j) (the forward variables). Observe that the a_{ij}s can be computed using the standard Needleman-Wunsch algorithm with max replaced by sum. The recursion is

$$a_{ij} = wa_{i-1,j} + wa_{i,j-1} + s_{ij}a_{i-1,j-1} \tag{1}$$

where $i, j \geq 1$ and $a_{0j} = a_{j0} = wj$. Let $v_j = (a_{1j}, a_{2j}, \ldots, a_{Tj})$ be the jth column vector ($0 \leq j \leq U$). The notation $v_j(i)$ is used to denote a_{ij}. We will fix a constant r, and write v_{r+1} in terms of v_r. First, consider the lower-triangular $(T+1)\times(T+1)$ matrix W where the ijth entry of W is $w^{|i-j|+1}$, $i \geq j$ and $w_{ij} = 0$ otherwise. Similarly, let S_r be the $(T+1)\times(T+1)$ matrix with 1 on the diagonal, and $S_r[ij] = s_{jr}$ for $i = j+1$ and $S_r[ij] = 0$ otherwise. Observe that recursion (1) can be rewritten as

$$v_r = WS_r v_{r-1}.$$

In order to sample efficiently, we will want to compute v_{r-1} in terms of v_r:

$$v_{r-1} = S_r^{-1}W^{-1}v_r. \tag{2}$$

The matrix W is invertible iff $w \neq 0$, which is always the case since $w = e^g$ for some real number g. All the matrices S_r are invertible. Thus, we can solve for v_{r-1} given v_r. Matrix inversion and muliplication is expensive, however in our case equation (2) yields an effective recursive procedure for computing a_{ir} because of the special structure of the matrices. It is easily seen that W^{-1} is a banded matrix with nonzero entries only on the diagonal and off-diagonal, and S_r^{-1} is a lower triangular matrix. By multiplying them we obtain the following set of equations for computing v_{r-1} from v_r:

$$v_{r-1}(0) = \frac{1}{w}v_r(0) \tag{3}$$

$$v_{r-1}(1) = \frac{v_r(1)}{w} + v_r(0) - \frac{s_{1r}}{w}v_{r-1}(0) \tag{4}$$

$$\vdots$$

$$v_{r-1}(k) = \frac{v_r(k)}{w} + v_r(k-1) - \frac{s_{kr}}{w}v_{r-1}(k-1) \tag{5}$$

$$\vdots$$

$$v_{r-1}(T) = \frac{v_r(T)}{w} + v_r(T-1) - \frac{s_{Tr}}{w}v_{r-1}(T-1) \tag{6}$$

Thus, the vector v_{r-1} can be computed in time $O(T)$ from the vector v_r by computing the $v_{r-1}(k)$ in the order $v_{r-1}(0), v_{r-1}(1), \ldots, v_{r-1}(T)$. Although it is possible to derive the recursions (3) - (8) directly from (1), we have included the matrix derivation since it is useful in generalizing the backward computation of forward variables to other problems.

The sampling algorithm for randomly selecting an alignment path according to its weight is therefore to first compute v_U using the forward algorithm variation of the Needleman-Wunsch algorithm (max replaced by sum). This is done using only $O(T)$ memory by keeping only v_{r-1} in memory for the computation of v_r. The samples can then be computed simultaneously according to Lemma 1 by computing $v_{U-1}, \ldots, v_0$ one column at a time and always throwing away columns after they have been used (to keep the computations at $O(T)$ memory). The running time of the algorithm for generating k samples is therefore $O(TU + k(T+U))$. The memory requirement, like the Hirschberg algorithm, is only $O(T+U)$.

More general alignment frameworks (for example allowing for gap open and extension penalties) can be sampled using minor midifications of the above algorithm.