

WASHINGTON UNIVERSITY
DIVISION OF BIOLOGY AND BIOMEDICAL SCIENCES
Program in Molecular Genetics

Dissertation Examination Committee:

Sean R. Eddy, Chair

Steven Johnson

Rob Mitra

Tim Schedl

Jim Skeath

Gary Stormo

REMOTE PROTEIN HOMOLOGY DETECTION USING HIDDEN MARKOV
MODELS

by

Steven Johnson B.S.

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University
in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

December, 2006

Saint Louis, Missouri

Acknowledgments

The work described in this thesis is due in a large part to many people who have given me guidance and support through the years. First, I would like to thank my advisor, Dr. Sean Eddy, for accepting me into his lab and providing me with support and a fantastic intellectual environment. All of the Eddy lab members have provided me with encouragement, training, good laughs, and great fellowship. In particular, I would like to thank Drs. Robin Dowell and Alex Coventry. If this research represents anything, it is that it is possible for a student to reach beyond the boundaries of his/her previous training. I definitely started my graduate research as a pure biologist who wished to learn more about computational methods. Any ‘rounding’ of my abilities that I have obtained is due in large part to their patience and ability to explain concepts in mathematics and computer science. To them, I am very grateful.

I would also like to thank my committee members Drs. Steve Johnson, Rob Mitra, Tim Schedl, Jim Skeath, and Gary Stormo. It is a multidisciplinary committee and it matches my work. Their different perspectives on the important issues in my research has been invaluable.

However, I would not have even arrived here at Washington University if it were not due to the support and love from my parents, Lawery and Linda Johnson. From the earliest days of finding bathroom sinks full of mysterious liquids, which I gleefully exclaimed were my ‘spearmints’, to the twisted path that brought me to Washington University, they have unfailingly encouraged me.

There is an expression that, ‘the last miles are truly the hardest’. I can’t think of a more fitting way to sum up my thesis work. Those last miles were made bearable by one person, Carolyn Guild. I have seen her deal with the challenges in her life with strength and grace and then turn around and support me when I couldn’t follow her example. I am extremely thankful to have her in my life.

Finally, I would like to acknowledge and dedicate this thesis to the littlest people that made this possible:

'To all the cocktail napkins I've scribbled on through the years.'

Steven Johnson

Washington University in Saint Louis
September 2006

Contents

Acknowledgments	iii
List of Tables	ix
List of Figures	x
1 Background	1
1.1 Sequence Comparison and Remote Homology Detection	1
1.2 Early Pairwise Sequence Comparisons	2
1.3 Pairwise Sequence Comparison Algorithms	3
1.4 Local Alignment Score Interpretation	4
1.5 Information Revealed by Homologous Sequences	4
1.6 Position Specific Scoring Matrices and Profiles	5
1.7 Profile Hidden Markov Models	6
1.7.1 HMM Architecture	6
1.7.2 Domain Modeling with HMMs	8
1.7.3 Probabilistic Modeling	9
1.7.4 HMM Scoring Algorithms	9
1.8 Further Advances in Sequence Comparison Methods	10
1.8.1 Iterative Methods	10
1.8.2 Domain Databases	11
1.8.3 Model-Model Alignments	11
1.9 Outline of This Work	12
2 Remote Protein Homolog Benchmark	14
2.1 Background	14
2.1.1 Homolog Database Selection	14

2.1.2	SCOP Database	16
2.1.3	Benchmark Alignments and Models	16
2.1.4	Source of Non-Homologous Sequences	17
2.2	Results	17
2.2.1	Reproduction of Madera/Gough	17
2.2.2	Problems with Madera/Gough	20
2.2.3	Improved Benchmark	23
2.3	Concluding Remarks	25
2.4	Methods	26
2.4.1	Reproduction of Madera/Gough Benchmark	26
2.4.2	Test Database	27
2.4.3	WU-BLAST Family Pairwise Search	28
2.4.4	SAM	28
2.4.5	HMMER 2.3.2	28
2.4.6	Determination of Significant Performance Differences	29
2.4.7	Availability	29
3	Information Content Based Sequence Weighting	30
3.1	Background	30
3.1.1	Information Theory and Scoring Systems	32
3.2	Results	33
3.2.1	HMMER 2.3.2 Models Heavily Weight the Training Sequences	33
3.2.2	Entropy and Information Content	34
3.2.3	Information Content Based Sequence Weighting	35
3.2.4	Optimization of Information Content Based Sequence Weighting	36
3.2.5	Information Content-Based Sequence Weighting Improves HMMER Local Model Performance	37
3.2.6	Information Content Based Sequence Weighting Improves HMMER Glocal Model Performance	37
3.2.7	SAM 3.5 Demonstrates Improved Performance Over HMMER	38
3.3	Concluding Remarks	40
3.4	Methods	41
3.4.1	Optimization of Information Content	41
3.4.2	Optimization of SAM 3.5	41

3.4.3	Availability	41
4	Forward Algorithm	42
4.1	Background	42
4.1.1	Example of Forward Algorithm Sensitivity	44
4.2	Results	46
4.2.1	Fitting Local Forward Null Score Distributions	46
4.2.2	1% Tail of Forward Null Score Distributions Fit an Exponential Distribution	47
4.2.3	The Exponential Distribution Provides Robust P-Values	48
4.2.4	HMMER Local Forward Searches with Exponential P-Values are an improvement over Local Viterbi	49
4.3	Concluding Remarks	50
4.4	Methods	51
4.4.1	Posterior Probability Matrix	51
4.4.2	Fitting of Forward Null Score Distributions	52
4.4.3	HMMER 2.5	52
4.4.4	Availability	53
5	HMM Search Heuristic: HMMERHEAD	54
5.1	Background	54
5.1.1	HMM Scoring Algorithms	57
5.2	Results	58
5.2.1	HMMERHEAD Algorithm	58
5.2.2	HMMERHEAD is Faster Than Default Scoring Algorithms	60
5.2.3	HMMERHEAD's Loss in Sensitivity is Minimal	61
5.3	Concluding Remarks	63
5.4	Methods	65
5.4.1	HMMERHEAD and HMMER 2.5 Timings	65
5.4.2	Test Database Modifications	65
5.4.3	Availability	65
6	JackHMMER	66
6.1	Background	66
6.1.1	Position Specific Scoring Matrices and Profile	67

6.1.2	Pairwise Sequence Comparisons	68
6.1.3	Iterative HMM Searches	68
6.2	Results	69
6.2.1	JackHMMER Algorithm	69
6.2.2	General Iterative Method Concern	69
6.2.3	Sub-Databases Limit the Performance of Iterative HMM Searches	70
6.2.4	JackHMMER Outperforms NCBI's Psi-BLAST and SAM's t-02	71
6.3	Concluding Remarks	72
6.4	Methods	73
6.4.1	Iteration Database	73
6.4.2	Testset Modification	74
6.4.3	Benchmarking Procedure	74
6.4.4	Comparison Programs	74
6.4.5	Availability	75
	Appendix A Major Scripts/Programs	76
A.1	Overview	76
A.2	Accessory Scripts/Programs	76
	Appendix B Benchmarking Pipelines	80
B.1	Overview of My Benchmark	80
B.2	Madera and Gough Benchmark Overview	82
	References	83
	Vita	92

List of Tables

- 3.1 **Average score per alignment position using PAM matrices to score alignments between sequences of varying alignment distances.**

Data taken from Altschul 1991. Bold values are the optimal mean score per alignment position for the sequences of the given distance. 32

- 5.1 **HMMERHEAD parameter settings and corresponding speed improvements for the different model and search types.**

Fold speedup is measured as the mean default algorithm search time/mean HMMERHEAD search time. Bold parameter settings are those that achieve maximum speedup with a minimum loss in sensitivity relative to the default algorithm. (See section 5.2.3) . . . 61

List of Figures

1.1	Structural Alignment of Calpactin.	
	A partial multiple sequence alignment of the protein calpactin reveals considerable position-specific information about this protein. Boxed positions demonstrate strong residue preference, column 30, and increased indel probability, columns 43-49.	5
1.2	State Diagram of a HMMER Hidden Markov Model	7
2.1	Madera/Gough Reproduction. Local Model Performance	
	Reproduction of the Madera and Gough benchmark confirms their results. A) Figure taken from Madera/Gough publication. 1S-BL&CLW-HH is HMMER 2.3.2 local models' performance. 1S-BL&CLW-SS is SAM 3.2.1 local models' performance. B) My reproduction of the Madera/Gough benchmark.	18
2.2	Madera/Gough Benchmark. Glocal Model Performance	
	HMMER 2.3.2 glocal models outperform SAM 3.2.1 glocal models. .	19
2.3	WU-BLASTP Versus WU-BLAST Family Pairwise Search	
	WU-BLAST FPS outperforms WU-BLASTP. Error bars represent the minimum and maximum true positives identified from bootstrapping.	22
2.4	Improved Benchmark	
	HMMER 2.3.2 local models underperform on the improved benchmark. Error bars represent the minimum and maximum true positives identified from bootstrapping. A) HMMER 2.3.2 local models are still underperforming. B) HMMER and SAM glocal models identify an equivalent number of trues.	24

3.1	Mean Relative Entropy of Match State Emissions	
	The match state mean relative entropy was calculated for each of the HMMER 2.3.2 benchmark models.	34
3.2	Information Content Based Sequence Weighting. Local Model Performance	
	Information content based sequence weighting significantly improves local model performance. Error bars represent the minimum and maximum true positives identified from bootstrapping.	37
3.3	Information Content Based Sequence Weighting. Glocal Model Performance	
	Information content based sequence weighting modestly improves glocal model performance. Error bars represent the minimum and maximum true positives identified from bootstrapping.	38
3.4	SAM 3.5 Performance	
	The latest version of SAM shows improved performance over HMMER even with the new sequence weighting scheme. Error bars represent the minimum and maximum true positives identified from bootstrapping. A) Local Model Performance B) Glocal Model Performance.	39
4.1	Match State Posterior Probability Matrix of d1bqca_ HMM and the d1cz1a_ Sequence	
	The match state posterior probability matrix for d1bqca_ and d1cz1a_ reveals several highly probable alignment segments.	45
4.2	Fits to the 1% Tail of the d1b72a_ Forward Null Score Distribution	
	The 1% tail of the d1b72a_ Forward null score distribution fitted to the Gumbel, hyper-exponential, and exponential distributions.	47
4.3	Robustness of Exponential Based P-Values	
	Expected versus observed number of sequences with the given P-value.	48
4.4	Performance Benchmark of Local Forward Utilizing Exponential Based P-Values	
	HMMER 2.5 local Forward outperforms HMMER 2.5 local Viterbi. Error bars represent the minimum and maximum true positives identified from bootstrapping.	50

5.1	HMMERHEAD: Word Hit Identification	58
5.2	HMMERHEAD: Optimal Scoring Ungapped Word Extension	59
5.3	HMMERHEAD: Optimal Scoring Gapped Alignment	60
5.4	HMMERHEAD Local Forward Performance	
	HMMERHEAD local Forward 6k/2k/7k/20k has a minimal loss in sensitivity. While HMMERHEAD local Forward 7k/2k/8k/20k obtains a faster search speed, it displays a greater loss in sensitivity. Error bars represent the minimum and maximum true positives identified from bootstrapping.	62
5.5	HMMERHEAD Local Viterbi Performance	
	HMMERHEAD local Viterbi 6k/2k/7k/20k identifies a comparable number of trues as the Viterbi algorithm. While HMMERHEAD local Viterbi 7k/2k/8k/20k obtains a faster search speed, it displays a greater loss in sensitivity. Error bars represent the minimum and maximum true positives identified from bootstrapping.	63
5.6	HMMERHEAD Glocal Viterbi Performance	
	HMMERHEAD glocal Viterbi 7k/2k/8k/22k identifies a comparable number of trues as the Viterbi algorithm. While HMMERHEAD glocal Viterbi 8k/2k/9k/22k obtains a faster search speed, it displays a greater loss in sensitivity. Error bars represent the minimum and maximum true positives identified from bootstrapping.	64
6.1	Overview of an Iterative Search Strategy	66
6.2	Effect of Sub-Database Use on JackHMMER Searches	
	The utilization of a sub-database significantly impairs the performance of the JackHMMER iterative approach. Error bars represent the minimum and maximum true positives identified from bootstrapping.	71
6.3	JackHMMER Performance	
	JackHMMER outperforms Psi-BLAST and SAM's t-02. Error bars represent the minimum and maximum true positives identified from bootstrapping.	72

ABSTRACT OF THE DISSERTATION

Remote Protein Homology Detection Using Hidden Markov Models

by

Steven Johnson

Doctor of Philosophy in Molecular Genetics

Washington University in St. Louis

September, 2006

Sean R. Eddy, Chairman

The first sequence comparison algorithms were introduced over 30 years ago. The motivation behind that work still exists today, to connect the vast reservoir of existing protein knowledge to lesser known sequences. If two proteins are deemed homologous, then information regarding function and structure may be common between them. At the least, it provides a basis for hypotheses and experimentation. Sequence comparison methods, and the data access that they rely on, have grown considerably over the decades. This growth has primarily come from a constant cycle of incorporating biological sources of information into search algorithms, the identification of more distant homologies, and the further refinement of information gained from the new homologies into the next generation of algorithms and search strategies. The focus of this research has been in the use of Hidden Markov Models (HMMs) for remote protein homology detection. We have implemented information content sequence weighting to improve local alignment searches. We identified that the Forward null score

distribution can be approximated by the exponential distribution and utilized this to calculate accurate and robust P-values for HMM Forward scores. In addition, we have developed an HMM search heuristic to decrease the time required to search large databases. Further, we have incorporated this heuristic into an iterative HMM search method that significantly increases search sensitivity.

Chapter 1

Background

1.1 Sequence Comparison and Remote Homology Detection

The focus of my thesis research is the improvement of remote protein homology detection using Hidden Markov Models. It has been 40 years since the first sequence comparison algorithm was introduced [24]. The motivation behind that work still exists today, to connect the vast reservoir of existing protein knowledge to lesser known sequences. If two proteins are deemed homologous, then information regarding function and structure may be common between them. At the least, it provides a basis for hypotheses and experimentation.

Sequence comparison methods, and the data access that they rely on, have grown considerably over the decades. This growth has primarily come from a constant cycle of incorporating biological sources of information into search algorithms, the identification of more distant homologies, and the further refinement of information gained from the new homologies into the next generation of algorithms and search strategies. Many of the advances in sequence comparison that I outline below fit into this pattern.

1.2 Early Pairwise Sequence Comparisons

In 1961, Watson and Kendrew obtained the full length protein sequence of the sperm whale myoglobin [91]. In their analysis of this protein sequence, they wished to align their newly obtained sequence to a known protein. Therefore, they did what was common at the time. Armed with the protein structure and the knowledge of the biochemical properties of the different amino acids, they aligned their new myoglobin sequence with the human hemoglobin sequence by hand. In fact, during this era, it wasn't uncommon to print sequences on slips of paper and align the pieces on your living room floor [16].

It was still plausible to do this at that time, due to the limited number and familiarity of biologists with the less than 60 known protein sequences [16]. However, as the number of known protein sequences grew and outpaced the rate of protein structure determination, spurred in part by the advent of DNA sequencing in 1975, it was becoming completely impractical to hand align sequences to discover homologous protein regions. There needed to be an automated way to align protein sequences. The question at the heart of this problem is, 'How do you quantify the similarity between two proteins?'

With a scoring scheme and penalties for the insertion of gaps, one can go through all possible alignments and calculate the overall score of each alignment by summing the scores of the individual alignment columns. Initial attempts involved the creation of residue scoring schemes that tried to duplicate the human process. Aligning identical residues and residues of similar biochemical properties were given positive scores, while aligning residues with conflicting properties, i.e. hydrophobic and hydrophilic, were given negative scores [24].

In 1978, Dr. Margaret Dayhoff pioneered an alternative scoring approach. Since 1965, Dr. Dayhoff had been compiling and publishing the known protein sequences in her 'Atlas of Protein Sequence and Structure' [12]. She observed biases in residue replacements in her collection of protein families. In an effort to take advantage of this information, she tabulated the relative mutability of the different amino acids and their frequencies to create the PAM substitution matrices. The important difference between her matrices and previous efforts were that they were derived from observed amino acid replacements in protein

families, as opposed to human dictated replacement rules. While the PAM matrices have generally been replaced by newer methods on updated sequence databases, they still rely on using observed residue replacements in real protein sequences [33,42].

1.3 Pairwise Sequence Comparison Algorithms

There have been a number of algorithms developed to identify highly scoring alignments between two sequences. I will briefly cover two of the main pairwise sequence comparison algorithms.

In 1970, Needleman and Wunsch published an algorithm that enabled the identification of the highest scoring global alignment between two sequences [59]. This was performed by filling in a $M \times N$ matrix, where M and N are the lengths of the two sequences. At each cell in the array, one chooses the highest scoring path that would have resulted in the alignment of the given residues. Once this matrix is filled in, one can traceback through the maximization steps to identify the highest scoring alignment.

In 1981, Smith and Waterman published a modification to the Needleman-Wunsch algorithm that allowed for the identification of the optimal alignment between two sub-sequences [78]. Their algorithm included the same maximizations as Needleman and Wunsch. However, they also required a score of zero to be considered when determining the maximum score. In this case, the traceback starts at the highest scoring cell and stops when a cell value of zero is reached. This allows one to ignore those regions between two sequences that poorly align and identify the highest scoring local alignment. The identification of high similarity regions in long sequences increased the sensitivity of sequence database searches [16,17]. Due to the usefulness of this type of sequence comparison and the growth of sequence databases, faster algorithms have been developed to identify high scoring local alignments [4,48,53,55,92].

1.4 Local Alignment Score Interpretation

As sequence comparison methods became more sensitive and database sizes grew, new exciting protein homologies, like the oncogenic v-sis and the platelet-derived growth factor, were identified [18]. As more biologists became aware of the benefit of such searches, sequence comparison searches grew increasingly more popular [16,17]. However, there was a considerable amount of confusion as to what defined a significant sequence similarity [53]. Initial efforts used the mean and standard deviation of scores from random sequences to ascertain the significance of the scores obtained from biological sequences [53,79]. While these methods were more useful measures than an alignment score, optimal local alignment scores do not follow the assumed underlying Gaussian distribution. This can result in the overestimation of the significance of some scores.

In 1990, Karlin and Altschul published a study regarding the distribution of optimal scores from ungapped local alignments [13,14,43]. They showed that given certain assumptions, this score distribution approaches a Gumbel distribution [43]. The crucial element that came from this work was that now there was a solid mathematical foundation for the calculation of P-values for local alignment scores. It is important to note that there were several assumptions made in their work. In particular, their work assumed sequences had no compositional bias, were infinitely long, and that the local alignments were ungapped.

1.5 Information Revealed by Homologous Sequences

As the number of known protein sequences grew, it became common to have multiple homologous sequences of your gene of interest. As can be seen in Figure 1.1, there is information that can be gathered from a multiple sequence alignment that is not apparent from a single sequence. From this partial structural alignment of calpactin, one can identify positions where there is a

```

          0          10          20          30          40
1a4pB PSQMEHAMETMMFTFHKFAGDKGYLTKEDLRVLMKEKEFPGFLENQKDPLA
1uwoA -SELEKAMVALIDVFHQYSGDKHKLKSELKELINNELSHFLEE--KEQE
5icb  -----SPEELKGIFEKYAADPNQLSKEELKLLLQTEFPSLLKGP----T
 1ncx  EDAK GKSEELANCFRIFDNADGFIDIEELGEILRA--TGEH----TEED
1djxB KMLTQ--RAEIDRAFEEAAGSAETLSVERLVTFLQHQQREEE-----AL
1cfpB MSELEKAVVALIDVFHQYSGDKHKLKSELKELINNELSHFLEIKEQ--
          60          70          80          90
1a4pB VDKIMKDLQaRDGKVGQSFSLIAGLTIACNDYFVVHMK-
1uwoA VDKVMETLDNDGDGECDFQEFMAFVAMTTACHEFFEHE----
5icb  LDELFEELDKNGDGEVSFEFQVLVKKISQ-----
 1ncx  IEDLMKSDKNNDGRIDFDEFKMMEGV-----
1djxB ALSLIEREPSEAQRQMTKDGFLMYLLSGNAFSL---HRRVY-
1cfpB -METLDSGD---GECDFQEFMAFVAMITTACFEHE-----

```

Figure 1.1: **Structural Alignment of Calpactin.**

A partial multiple sequence alignment of the protein calpactin reveals considerable position-specific information about this protein. Boxed positions demonstrate strong residue preference, column 30, and increased indel probability, columns 43-49.

strong preference for a particular amino acid, such as the leucine at position 30. In addition, one can identify positions that would be more or less likely to have insertions or deletions, such as positions 43-49 or positions 29-30, respectively. Given this, the question now becomes, ‘How do you utilize this alignment information in a database search?’

1.6 Position Specific Scoring Matrices and Profiles

One of the first approaches to try to take advantage of the information present in multiple homologous sequences was the concept of Position Specific Scoring Matrices (PSSM), or weight matrices, introduced by Stormo [85]. There have been many alternative strategies in the construction of PSSMs. The general idea is to construct a JxK matrix, where J is the number of possible residues, e.g. 20 amino acids, and K is the length of the alignment being modeled [34, 83, 84].

For each position in the alignment, the probability of each possible amino acid at that position is calculated. This PSSM is then slid along a query sequence and the score of the window at each position is calculated. In 1987, Gribskov *et al.* introduced the concept of profiles, whose design is similar to PSSMs. However, Gribskov's profiles are used in a Smith-Waterman type search allowing for affine gap penalties [28]. Therefore, Gribskov profiles share the features listed above with the addition of two position specific gap penalties (gap-open, gap-extend) for every alignment column modeled. However, these gap penalties were calculated in an *ad hoc* manner and had no theoretical foundation for their determination.

1.7 Profile Hidden Markov Models

Hidden Markov Models (HMMs) were extensively employed in voice recognition software starting in the 1970's [67]. In voice recognition, a speech sample is broken into small sounds and this pattern of sounds needs to be matched to a known word. Obviously, there are variations in tone and dialect that can make this a difficult problem. Krogh *et al.* was one of the first groups to recognize the similarity between identifying slightly varying spoken words and identifying protein sequences [51]. In their work, they describe an HMM that can be used to model a protein domain and show that it was effective at modeling the globin domain. Later the same year, Eddy *et al.* released HMMER, a software package that uses Hidden Markov Models to model protein domains and identify remote protein homologies. My research relies extensively on HMMER. I will now go into a more detailed discussion of its default construction and usage.

1.7.1 HMM Architecture

A Hidden Markov Model of a protein domain is typically built from a multiple sequence alignment of homologous domains. HMMs are commonly represented by state diagrams, such as Figure 1.2, which consist of 'states' and paths connecting these states which are referred to as 'transitions'. I will start by discussing the details of the core of a typical HMMER model which is made up of match states, insert states, delete states, and transitions between these states.

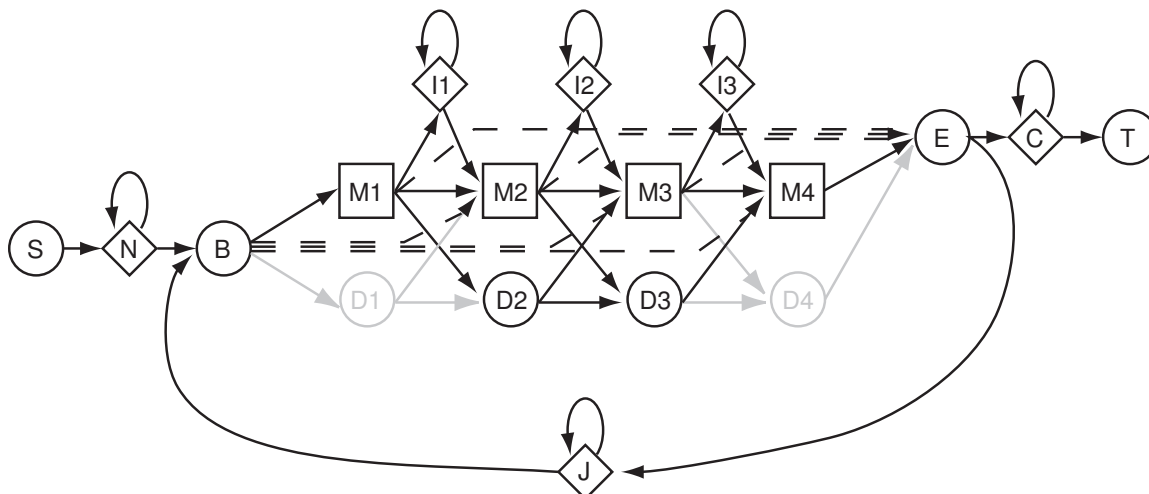


Figure 1.2: **State Diagram of a HMMER Hidden Markov Model**

Core States and Transitions

Match states, ‘M’ labeled squares in Figure 1.2, represent the conserved columns in the training alignment. The match states ‘emit’ amino acid residues according to probabilities usually based on the normalized combination of the observed amino acid counts in the alignment column they represent and pseudocounts from the prior. In a default HMMER construction, those alignment columns with < 50% gap characters are designated as match states.

Insert states, ‘I’ labeled diamonds in Figure 1.2, allow for the emission of residues between the match states. The emission probabilities for the insert state emission probabilities are derived primarily from residue frequencies calculated from an early version of Pfam. (For more information on Pfam, see section 1.8.2.)

Delete states, ‘D’ labeled circles in Figure 1.2, are an example of a non-emitting state or silent state. They can be thought of as placeholders that allow for paths through the HMM that do not involve emitting a residue of the target sequence.

I define a node of a HMMER HMM to consist of a set of alignable match, delete, and insert states as one progresses left to right along the HMM. e.g. M2, D2, and I2 are a node.

Transitions, represented as arrows in Figure 1.2, are the possible paths to and from the individual states in the HMM. Each transition has a probability, and all transition probabilities out of a state must sum to one. The transition probabilities to and from the match, insert, and delete states typically come from the normalized observed counts from the training alignment and a single component Dirichlet prior.

Additional States

N and C states, ‘N’ and ‘C’ labeled diamonds, are emission states that allow for the modeling of N and C terminal sequence around a domain. These emission probabilities correspond to background database frequencies. The J state allows the HMM to be able to detect multiple copies of the protein domain in a target sequence.

Local and Glocal HMMs

The two most commonly used types of HMMER HMMs are local and global/local (glocal) models. Local models generate an alignment of a segment of a target sequence to some subset of the core HMM states. Glocal models generate an alignment of the entire core model to some segment of the target sequence. These two model types are achieved by controlling the begin state, B labeled circle, to match state transitions and the match state to end state, E labeled circle, transitions. (Dashed lines in Figure 1.2.) In local models, these transitions are allowed, while in glocal models they are set to zero.

1.7.2 Domain Modeling with HMMs

Given the flexibility of HMMER’s HMMs we can re-visit the types of information that can be revealed from a multiple sequence alignment. By having match state emission probabilities for each modeled alignment column, HMMs can easily incorporate position-specific residue preferences. Additionally, by calculating transition probabilities that utilize the observed times the training

sequences use those paths, HMMs can incorporate position-specific information on insertion and deletion penalties.

1.7.3 Probabilistic Modeling

At this point, the reader may recognize some similarities between profiles and HMMs. While HMM match states and profiles are quite similar in their construction, this is where the similarity ends. The match state emission scores incorporated into HMMER's HMMs are typically derived from a combination of observed and prior data and have a probabilistic basis. This is also true for the column scores for most profiles [5, 28]. However, profile gap penalties do not. Profile gap penalties come from empirical studies that have demonstrated which values perform the best over a wide range of proteins. There are no underlying probabilities for most profile and pairwise sequence comparison methods' gap penalties. Thus, their alignment scores can not be considered likelihoods. In contrast, HMMER HMM's allow for a fully probabilistic modeling of insertions, deletions, and N and C terminal sequences flanking a modeled domain. Since HMMs are fully probabilistic, the scores returned by the various scoring algorithms are actual likelihoods.

1.7.4 HMM Scoring Algorithms

Once a HMMER HMM is constructed, it can be used to identify potentially homologous sequences. Two main algorithms that are used for this purpose are the Viterbi and Forward algorithms.

In 1963, Dr. Andrew Viterbi accepted a teaching position at the University of California Los Angeles. Around that time, the information theory work of Dr. Claude Shannon was gaining in popularity. While Dr. Viterbi had no formal background in information theory, he began to read published articles in the field and made it a topic of the classes he was teaching. He found teaching the subject quite challenging. "When doing research one gets into a mindset where it becomes second nature, but when it comes to teaching someone else's research, that's when it becomes clear whether one understands it fully. I found

information theory difficult to teach, so I started developing some tools.” [58]
The Viterbi algorithm was one of those teaching tools.

The Viterbi algorithm is a dynamic programming algorithm that calculates the $P(\text{Seq}, \hat{\pi} \mid \text{HMM}, \beta)$; where $\hat{\pi}$ is the optimal path over all paths π , and β is the parameterization of the HMM [67]. This is accomplished by filling in a $L \times M$ matrix where L and M are the lengths of the HMM and target sequence, respectively. In particular, L is the number of nodes in the HMM. The Viterbi algorithm is very similar to the Needleman-Wunsch and Smith-Waterman algorithms in that one selects the most probable of path ending at a particular position in the matrix. When the matrix is filled in, one can traceback from the maximum scoring state for the final target sequence position through the individual maximization steps to identify the highest scoring alignment between the HMM and the target sequence.

The Forward algorithm calculates the overall $P(\text{Seq} \mid \text{HMM}, \beta)$ where β is the parameterization of the HMM [67]. This is accomplished by again filling in a $L \times M$ matrix. However, in the Forward algorithm the maximization step of the Viterbi algorithm is replaced by a summation over all possible paths. Therefore, the score in the lower right-hand corner represents the probability that the observed sequence was generated by the model, considering all possible paths through the model that could have generated the sequence. Since the Forward score is a summation of the probabilities of the alternative paths and not a maximization, there is no traceback. Obviously, this also means there is no alignment between the sequence and the HMM returned from the Forward algorithm.

1.8 Further Advances in Sequence Comparison Methods

1.8.1 Iterative Methods

It is clear that the addition of homologous sequence information improved the sensitivity of sequence comparisons [63]. Therefore, increasing the number

and diversity of the aligned homologous sequences became a focus for further improvement. Iterative methods were introduced, particularly for profiles and HMMs [5, 15, 45]. In these methods, a profile or HMM is built from an initial alignment and is searched against a sequence database to identify homologs. These new homologs are then incorporated into the alignment, a new model is built and again searched against the database. Iterative methods are discussed further in Chapter 6.

1.8.2 Domain Databases

One result from the building of more advanced models of protein domains is the compilation of these domain models into databases. Using the PSSM method of Henikoff and Henikoff, the BLOCKS database was built in an automated fashion from sequences in SWISS-PROT that contained ungapped sequence motifs from Prosite [32, 66]. The Pfam database is a manually curated database of protein domains that have been built into HMMs using the HMMER software [6, 7, 23, 81]. These databases are particularly useful when one has a single sequence of interest and wishes to take advantage of these more sensitive models. Pfam, in particular, has been vital to the annotation of open reading frames from many large scale sequencing projects [21, 22, 47, 52].

1.8.3 Model-Model Alignments

Several groups have decided that if incorporating the information from a single alignment increases sequence comparison sensitivity, then utilizing the information present in more than one alignment might provide further sensitivity.

Recent work has investigated aligning models against each other to further increase sensitivity. One approach has been to compare sequence profiles against each other [70, 71]. In COMPASS, a profile-profile comparison method, the score for aligning the columns from two different profiles is derived from the relative entropy between the probability distributions of the two profiles. The profiles are aligned using these scores and the Smith-Waterman local alignment algorithm. Significance estimates of these scores are calculated by assuming the null

distribution for this method fits a Gumbel distribution according to Karlin and Altschul.

Alternatively, there has also been work in performing HMM-HMM alignments [19,80]. In HHsearch, a modified version of Viterbi is performed to identify the most probable co-emitted path given both HMMs in either a local or global fashion. HHsearch calculates a P-value for the score of this HMM-HMM alignment by assuming that null scores follow a Gumbel distribution.

It does appear that profile-profile and HMM-HMM methods are capable of detecting quite distant protein homologies [70,80]. However, recent work by Pearson and Sierk indicates that there is serious concern regarding the accuracy of their statistical estimates, and [65].

1.9 Outline of This Work

My work has been to improve the performance of Hidden Markov Models in the detection of remote protein homologies. Outlined below is the focus of the individual chapters in this thesis.

Chapter 2 discusses the design and implementation of the major benchmarks I have utilized during the course of my work. Benchmarks are vital tools that enable one to measure the effects of algorithm design and implementation. I reproduce the work of Madera/Gough that claimed that HMMER's local models were underperforming. Additionally, I discuss concerns I had about Madera/Gough's benchmark and describe the modified benchmark I use in my work.

Chapter 3 examines the underperformance of HMMER's local models. I describe issues involved in sequence weighting and my implementation of an information content based external sequence weighting strategy. The performance of this strategy is then demonstrated using my benchmark.

Chapter 4 describes my contribution to the collaborative effort to utilize the Forward algorithm in HMM/sequence scoring. I discuss the strengths and weaknesses of the Forward and Viterbi algorithms, as well as an example where

the Forward algorithm is more sensitive. I discuss the good fits between the Forward null score distributions of several models and the exponential distribution. I then show the performance of a HMMER implementation utilizing the exponential distribution for the calculation of Forward score P-values.

Chapter 5 addresses the issue of speed in HMM scoring algorithms. HMM scoring algorithms, such as Viterbi and Forward, are rigorous but slow. I describe a heuristic, HMMERHEAD, that my colleague Elon Portugaly and myself developed for Hidden Markov Models. I then show the speed improvement and sensitivity loss of this heuristic on my benchmark.

Chapter 6 describes iterative search approaches with HMMs. Utilizing the speed heuristic HMMERHEAD, I demonstrate the loss in performance of an iterative HMM approach that utilizes a sub-database for iteration. I then compare the performance of my iterative approach, JackHMMER, against other profile and HMM iterative programs.

Chapter 2

Remote Protein Homolog Benchmark

2.1 Background

In 2002, Madera and Gough published a study comparing several different homology detection programs [56]. Their critical review of HMMER performance was an initial focus of my research. Their results were surprising given previous internal benchmarks that had shown HMMER performance had been comparable or better than SAM. I will now discuss several issues in protein homology benchmarks that came to light during the course of reproducing their work.

2.1.1 Homolog Database Selection

At the core of any protein homology detection benchmark is a database where the homology between the sequences is known. Simply defined, two proteins are homologous if they share a common ancestor [62]. There have been several different approaches to compiling large sets of homologous protein sequences [11, 82]. The majority of these databases infer homology on the basis of either sequence or structural similarity.

Caveats of Structural-Based Homology Databases

As with sequence similarity, there are many degrees of structural similarity. One commonly used classification is the structural fold. The SCOP structural database defines two proteins as sharing the same fold when they have the same major secondary structures in the same topological arrangements [11]. It has been well established that proteins are able to undergo considerable primary sequence mutation while still maintaining the same overall fold. Thus, structure has been seen as a more sensitive indicator of homology [11,61]. Alternatively, there is a concern that protein folds represent stable thermodynamic structures that may have arisen multiple times by convergent evolution [11]. Therefore, solely sharing the same fold can be a problematic indicator of homology.

Caveats of Sequence-Based Homology Databases

One of the major concerns in using a database of homologs based on sequence similarity is the circularity involved. There is the concern that good performance on the benchmark is due to a program's ability to mimic the sequence similarity measure used to construct the database. If a new method were actually more sensitive and detected remote homologies not identified by the creating method, these new homologies would be incorrectly called false positives.

A second concern is in regards to the evolutionary distance of this type of database. It is a generally observed fact that protein structure evolves more slowly than protein sequence [11,35]. Thus, using sequence-based homology could limit the evolutionary distances between the sequences in this type of testset.

Finally, there is also an assumption that if two sequences possess sequence similarity, then they are homologous. However, there are several examples of proteins with dissimilar structural folds that possess similar sequence motifs, such as the KH and ferredoxin domains [29,50,54]. I go into this issue in more depth later in the chapter.

Given the aforementioned caveats, it would be better to test sequence-based homology programs on a testset not created through the sole use of sequence information. In fact, it appears that the sequence comparison field

has accepted using structure based definitions of homology to test sequence comparison methods [56, 63, 64].

2.1.2 SCOP Database

The database used by Madera and Gough was the SCOP database. The SCOP database consists of single domain protein sequences from PDB [11]. Each sequence in the SCOP database has been put into a hierarchical classification scheme on the basis of its structure. The SCOP classification scheme in descending order is Class, Fold, Superfamily, and Family. The top level, Class, groups proteins according their overall fraction of different secondary structures. ie. All alpha helix. Again, the Fold classification is determined by the order and topological connections of the main secondary structures. The next level is the Superfamily level. Sequences that share the same Superfamily may have low levels of sequence identity, but their tertiary structures and function imply that they share a common evolutionary origin. Proteins share the same Family classification if they possess greater than 30% identity or their functions and tertiary structures are very similar. For future reference, when I capitalize Fold or Superfamily, I am referring specifically to the SCOP classifications.

The Madera and Gough benchmark is structured similar to a pairwise sequence comparison benchmark by Green *et al.* [27]. Their protein testset were those sequences that shared Superfamily classifications. It was clear that the Fold level wouldn't represent a good set of clearly homologous, and the Family level, due to their higher degree of sequence similarity, would be too easy of a testset. To further increase the difficulty of the benchmark, the sequences used were from the Astral Compendium's version of SCOP that filters out all sequences with greater than or equal to 40% identity [8, 9].

2.1.3 Benchmark Alignments and Models

Ideally, profile HMMs are built from multiple sequence alignments of homologous domains. Thus, one needs to obtain alignments for each of the testset sequences. There are a number of trusted alignment databases such as

Pfam and FSSP [37,82]. However, none of these had adequate coverage of the Astral test sequences.

Alternatively, an initial set of proteins homologous to each testset sequence could be identified in a sequence database and the alignments could be generated for the benchmark. The downside of this approach is that the alignments may be of lower quality than those created from structural alignments or hand-curation. However, this would be appropriate since many users do not have access to these higher quality alignments of their proteins of interest. In the Madera and Gough benchmark, alignments were generated for each of the test sequences by identifying an initial set of homologs from a non-redundant database using Wu-BLAST and aligning the homologous segments using ClustalW.

2.1.4 Source of Non-Homologous Sequences

Performance on a protein homology detection benchmark should be measured as a balance between a method's ability to detect remote protein homolog sequences, while rejecting non-homologous sequences. Thus, there is a need for non-homologous sequences in the test database. These sequences can either be biological non-homologous sequences or artificially generated sequences. As I point out in the next section, there are some serious issues with using biological sequences as a source of non-homologous sequences.

2.2 Results

2.2.1 Reproduction of Madera/Gough

Agreement with Madera/Gough Local Model Result

In order to better understand Madera/Gough's results, I reproduced their benchmark. Due to slight variations in database versions and alignment protocols, I would need the exact alignments used by Madera and Gough to identically reproduce their results. I requested these alignments but

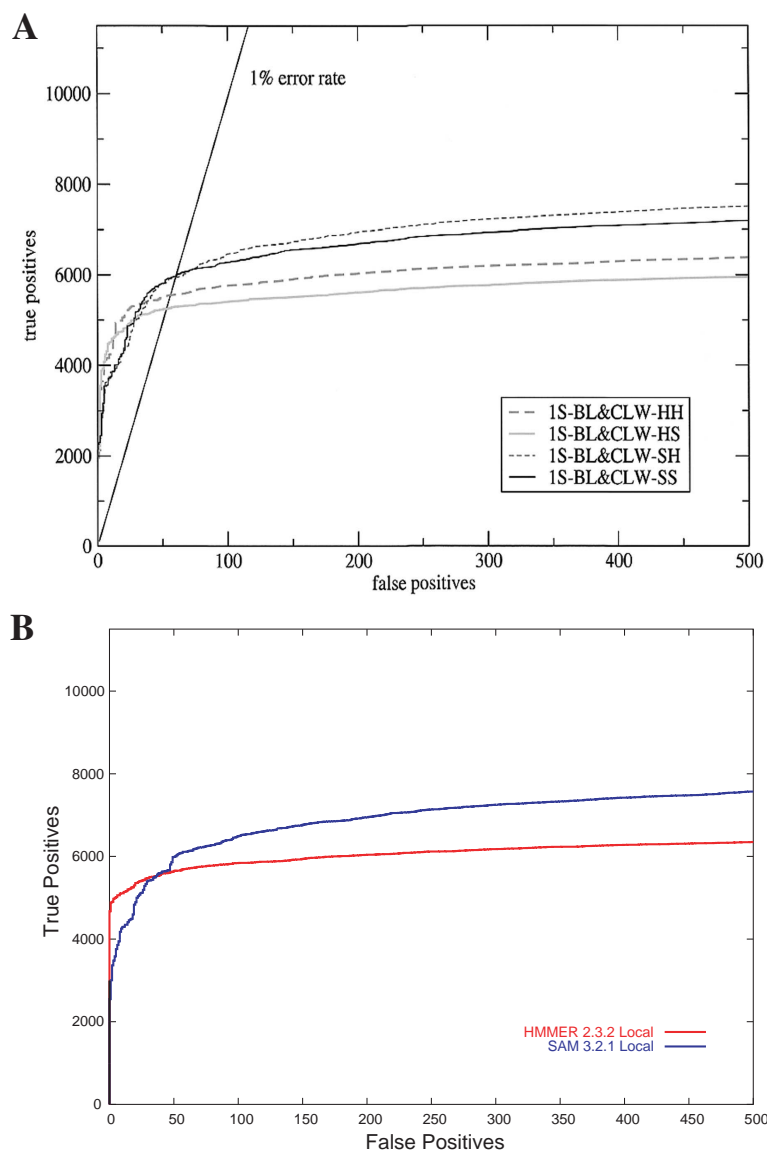


Figure 2.1: **Madera/Gough Reproduction. Local Model Performance** Reproduction of the Madera and Gough benchmark confirms their results. **A)** Figure taken from Madera/Gough publication. 1S-BL&CLW-HH is HMMER 2.3.2 local models' performance. 1S-BL&CLW-SS is SAM 3.2.1 local models' performance. **B)** My reproduction of the Madera/Gough benchmark.

unfortunately they were not saved by the authors. Figure 2.1A is taken from their study. Figure 2.1B shows the results of my reproduction. While there is some variation, both results are quite similar. The main differences are that it seems HMMER 2.3.2 local models show an improved performance at low false positive values and that SAM 3.2.1 shows an overall performance improvement, particularly at higher false positive values. In my reproduction, it is clear that HMMER's local models are underperforming relative to SAM. This was quite surprising because it conflicted with a previous internal benchmark. This benchmark, profmark, used sequences from the Pfam 2.0 full alignments. The families were split into sets of training and test sequences. Glocal models were built from the training sequences and then searched against a test database that consisted of the remaining Pfam test sequences and random sequences. On this benchmark, HMMER slightly outperformed SAM.

HMMER 2.3.2 Glocal Models Outperform and SAM 3.2.1 Glocal Models on the Madera/Gough Benchmark

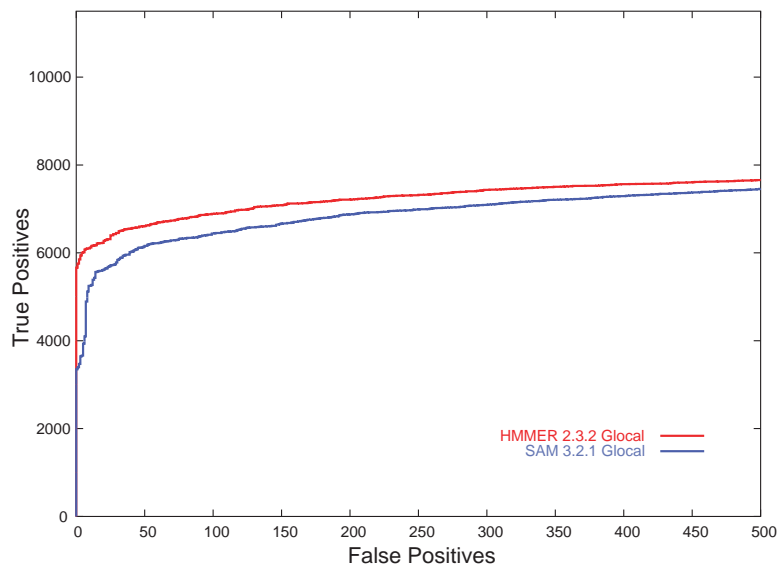


Figure 2.2: Madera/Gough Benchmark. Glocal Model Performance HMMER 2.3.2 glocal models outperform SAM 3.2.1 glocal models.

The focus of the profmark benchmark had been on glocal model performance. It was assumed that local and glocal models should perform

similarly. In Madera and Gough’s study, they did not examine the performance of HMMER’s glocal models. To test the assumption that local and glocal models perform similarly, I measured the performance of the HMMER 2.3.2 glocal models on the Madera/Gough benchmark (Figure 2.2). HMMER 2.3.2 glocal models outperform HMMER 2.3.2 local models, as well as the glocal models of SAM 3.2.1. This explains the conflict between the two studies.

2.2.2 Problems with Madera/Gough

During the reproduction of Madera and Gough’s work, I came across several problems with their benchmark. In this section, I describe the issues that I have with their benchmark and explain the modifications I made that resulted in the creation of a new benchmark.

SCOP Misclassifications

The previous benchmark by Madera and Gough used a test database that consisted solely of the SCOP database sequence. Homology and non-homology was defined by the SCOP Superfamily and Fold classification of the sequences. If two sequences shared the same Superfamily, they were considered homologous. If the sequences were in different Folds, they were considered non-homologous. Inspection of the top scoring false positives in their benchmark reveals highly significant scores between sequences from different folds. Thus, under Madera and Gough’s scoring scheme, these were considered false positives.

Examination of later versions of SCOP revealed that several of these sequences were reclassified as being in the same Superfamily. For example, in SCOP 1.50, several members of the L-2-Haloacid dehalogenase Superfamily hit a sequence in the epoxide hydrolase Superfamily, d1cr6a1, with expectation values, E-values, below $1e-10$. In later SCOP releases, the classification of these sequences was changed and all the sequences were placed in the HAD-like Superfamily. The SCOP database, like many databases, is under a constant state of manual revision.

Different Folds with Sequence Similarity

Some of the issues with using biological sequences as a source of non-homology are not due to classification errors, but are due to potential conflicts in sequence versus structural inference of homology. A recent paper by Krishna *et al.* reveals that there are ferredoxin functional domains that share a highly conserved 14 residue sequence motif. However, these sequences can be found within two different protein folds [50]. The pyruvate-ferredoxin fold consists of 4 beta sheets interspersed by 2 alpha helices. The quinol-fumarate fold, a functional ferredoxin, is made up of 4 alpha helices. Thus, from a structural standpoint these two sequences have different folds. However, functionally and sequence similarity-wise, they are similar. Conserved sequence motifs have been observed in other domains with dissimilar folds [29, 54]. Regardless of whether these cases arose from convergent evolution or ‘structural drift’, these are clear examples of conflict between structural and sequence based homology that are difficult to interpret [49]. This type of scenario would further complicate using SCOP structural classifications to define our non-homologous sequences.

In fact, further inspection of the high-ranking false positives in Madera and Gough’s benchmark reveals cross-Fold hits between several fumarate reductases and pyruvate-ferredoxin sequences with E-values as low as $1e-05$.

In the Madera and Gough benchmark, they excluded hits between the Rossmann and Rossmann-like Folds due to the number of cross-hits they observed between these two Folds. Instead of selectively excluding hits between certain Folds, I decided to seek out an alternative source of non-homologous sequences for my test database. I decided to use shuffled biological sequences in my test database as a trusted source of non-homologous sequences. I use shuffled biological sequences instead of random sequences because shuffled biological sequences still possess the same overall compositional residue biases that can be found in biological sequences.

Improvement of WU-BLAST Comparison

The comparison in Madera and Gough’s study between HMMs and the pairwise search method, WU-BLASTP, was misleading. Hidden Markov Models have the advantage of using multiple homologous sequences as input, while WU-BLASTP searches can only utilize single SCOP sequences. To better account for this advantage in input data, I replaced the single WU-BLASTP search with a WU-BLAST Family Pairwise Search. This method has been previously described and consists of comparing a ‘family’ of sequences to a database using a pairwise sequence comparison method [30]. In a Family Pairwise Search, when more than one ‘family’ sequence hits the same database sequence, their similarity measures are combined and represents the similarity of the family to that database sequence. It is clear that the WU-BLAST FPS approach outperforms a single query WU-BLASTP search (Figure 2.3).

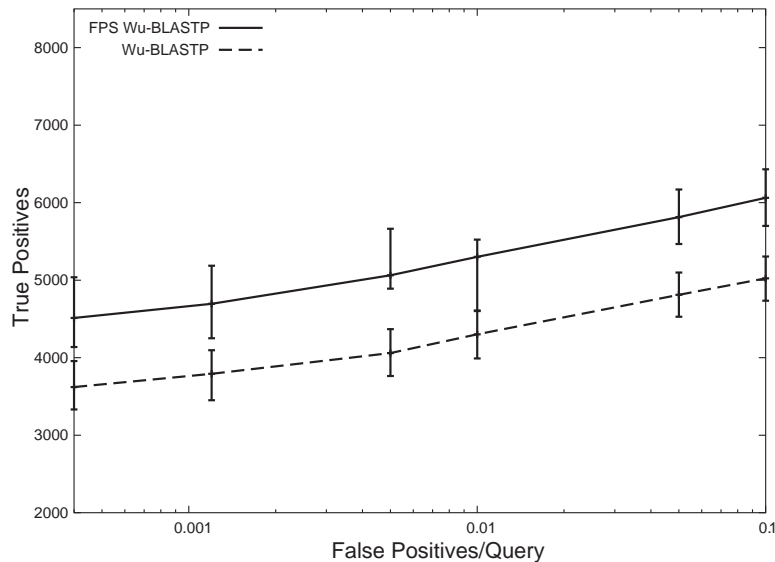


Figure 2.3: **WU-BLASTP Versus WU-BLAST Family Pairwise Search**
WU-BLAST FPS outperforms WU-BLASTP. Error bars represent the minimum and maximum true positives identified from bootstrapping.

Determination of Significant Performance Differences

Often it is unclear how to interpret differences in program performance. I employed a bootstrap re-sampling method similar to that of Brenner [27]. Significance is determined by using the percentile method for defining a confidence interval on a statistic [20]. Basically, I create alternative testsets by sampling with replacement from the original list of test sequences. I then measure the differences in performance between two different methods on each of these alternative testsets and determine the distribution of score differences. If the middle 95% of the values of this distribution does not overlap zero, I say that there is a statistically significant difference at the 0.05 level in the performance of the two methods.

2.2.3 Improved Benchmark

HMMER Local Models Are Underperforming

I measured HMMER 2.3.2's performance on the new benchmark (Figure 2.4). Unfortunately, the local model results reconfirm those found in Madera and Gough's benchmark. HMMER's local models still demonstrate a marked underperformance at a statistical level compared to SAM at all false positives per query levels.

HMMER 2.3.2 and SAM 3.2.1 Glocal Models Identify an Equivalent Number of Homologs

SAM 3.2.1 and HMMER 2.3.2 glocal models find equivalent numbers on the new benchmark. The optimization of SAM's model building scripts, described below, increased SAM performance and is a better representation of the performance of the two programs.

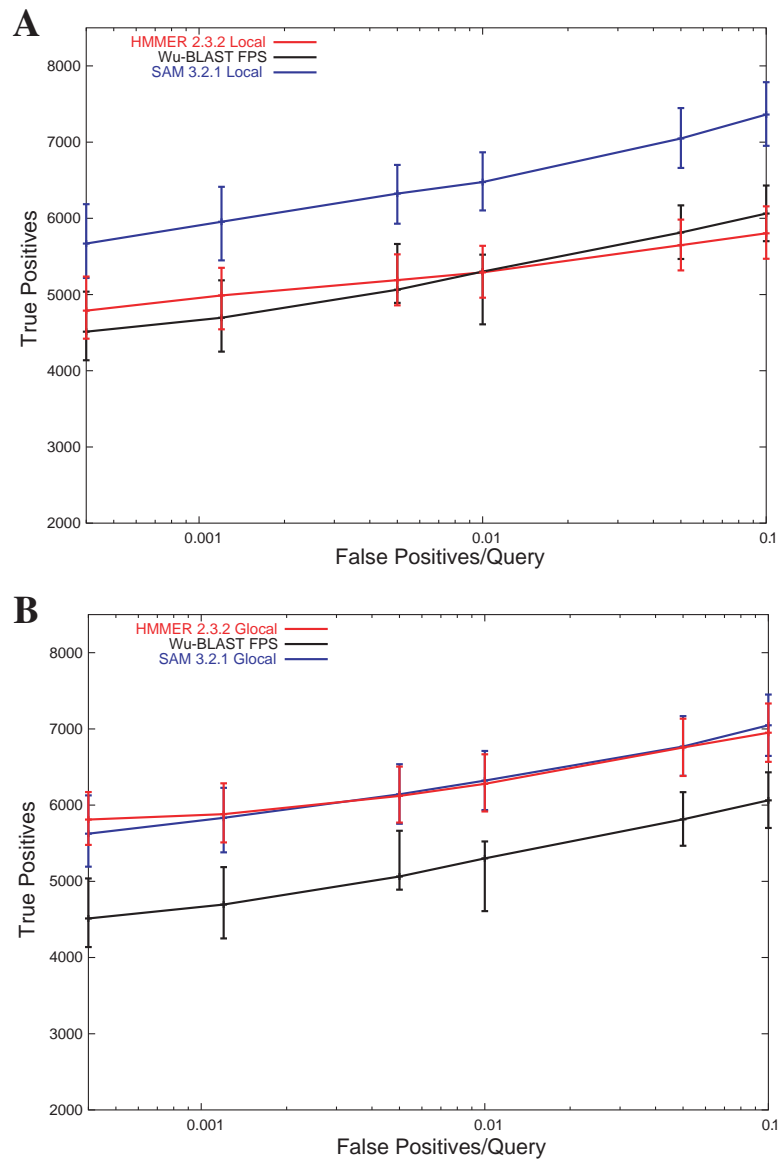


Figure 2.4: **Improved Benchmark**

HMMER 2.3.2 local models underperform on the improved benchmark. Error bars represent the minimum and maximum true positives identified from bootstrapping.

A) HMMER 2.3.2 local models are still underperforming. **B)** HMMER and SAM global models identify an equivalent number of trues.

2.3 Concluding Remarks

I reproduced the work of Madera and Gough and found the same major conclusion as their study. The local models of HMMER 2.3.2 are underperforming relative to SAM 3.2.1 local models. This was a confusing result because internal benchmarks had indicated that HMMER performance was comparable to that of SAM. Further experiments explained the reason for this conflicting evidence. HMMER's previous benchmarks focused on glocal model performance, while the Madera and Gough study focused on local model performance. Madera and Gough's benchmark, as well as my own benchmark, reveals that HMMER 2.3.2 local models are underperforming relative to SAM as well as HMMER's glocal models. The reason for the local model performance difference is explained in more detail in the next chapter.

In the course of reproducing Madera and Gough's work, I discovered several issues I had with their benchmark, which I then used to create my own benchmark. One of the main issues was regarding non-homologous sequences in the test database. In Madera and Gough's work, they used biological sequences as false positive or non-homologous sequences. They utilized the SCOP classification scheme to define whether sequences were homologous despite the fact that this classification is continuously being refined. A general concern with using biological sequences as sources of non-homology is that a sensitive method would be penalized for identifying a homology that is not detected in the database. As a specific example of this, there were several high scoring false positives between sequences from the L-2-Haloacid dehalogenase Superfamily and a sequence from the epoxide hydrolase Superfamily in SCOP 1.50. In later versions of SCOP, these sequences were combined into one Superfamily. Therefore, in my benchmark, I employ shuffled biological sequences as sources of non-homology. While this might not be ideal, since shuffled sequences do not contain the same types of regional compositional bias or short-period repeats as biological sequences, I decided that this would be better than using a clearly flawed scheme.

Another issue I had with Madera and Gough's benchmark was their use of single query searches to compare the performance of pairwise sequence

comparison methods to that of HMMs. I replaced their single query WU-BLAST searches with a family pairwise approach to better account for the advantage in input data that HMMs possess. Also, I utilized bootstrapping and a 5% confidence interval to more quantitatively measure the performance difference among different search programs.

One point that should be made in regards to both Madera and Gough's and my benchmark is that they do not necessarily reflect E-value accuracy. E-values are used in ranking the reported hits, but the performance measures are based off of true and false positive counts. Since rank order is used, these benchmarks will not indicate whether the E-values associated with the reported hits are accurate.

2.4 Methods

2.4.1 Reproduction of Madera/Gough Benchmark

SCOP/Astral Sequences

I used the Astral Compendium's filtered version of SCOP 1.50. Sequences in this dataset are filtered so that there are no sequences with $\geq 40\%$ identity. There are 2,893 sequences in this dataset. This database can be found at: <http://astral.berkeley.edu>

Alignment Generation

In order to build HMMs, I needed a multiple sequence alignment of homologs for each test sequence. Putative homologs for each test sequence were identified in a non-redundant database, NRDB90 (10/2002), using WU-BLAST 2.0MP [26, 38]. Database sequences with E-values $\leq 1e-05$ were considered putative homologs. Homologous regions were extracted and aligned using ClustalW 1.82 [40]. Several extremely large homolog sets, 23, could not be aligned after 3 days. For each of these, 1000 homolog sequences were randomly selected and aligned.

Since we wanted to identify homologs at the SCOP Superfamily level and we considered matches to the query sequence as too easy, 372 sequences from Superfamilies with one member were excluded. The final set of 2,521 alignments was then used to build HMMs using the various approaches.

Benchmarking Procedures

Each model is searched against the test database and a master list is created from all 2,521 search results. A typical line from this master list consists of the name of the query model, the target database sequence identified, and the E-value of the score. This list is then ranked by E-value in ascending order. Thus, those query/target hits with the most significant E-values are at the top of the list. Each hit in the list is scored as true, false, or ignored.

For the reproduction of Madera and Gough's benchmark, I used their scoring scheme. Hits to sequences from the same Superfamily are scored as true. Hits between sequences from different Folds are considered false. Hits between sequences that are from different Superfamilies but are within the same Fold are considered to have uncertain homology and ignored.

For my benchmark, I considered hits between sequences from the same Superfamily as true. Hits to shuffled sequences were scored as false and hits to SCOP sequences not in the same Superfamily were considered to have uncertain homology and ignored. Scripts used in this benchmark are discussed further in appendix A/B.

Performance is graphed as the number of true homologies detected at 1, 3, 12, 25, 125, and 250 false positives or 0.0004, 0.001, 0.005, 0.01, 0.05, 0.1 false positives per query (fpq), respectively.

2.4.2 Test Database

These sequences were generated by running SQUID's Shuffle program with the 2,521 Astral SCOP sequences as input. I generated five shuffled copies of each query sequence. These shuffled sequences were combined with the 2,521

Astral sequences to create a total test database size of 15,126 sequences. SQUID is included in the current HMMER version or can be found at:
<http://selab.janelia.org/software/>

2.4.3 WU-BLAST Family Pairwise Search

The pairwise sequence comparisons were performed using WU-BLASTP 2.0MP. I compared the performances of several methods of combining the E-values in this approach, such as mean log E-value, mean E-value, and minimum E-value [30]. Since they provided the best performance on my benchmark, I used minimum E-values.

2.4.4 SAM

SAM has several automated model building scripts. For the Madera and Gough benchmark, I plotted the performance of models built with the fw0.7 script, the same as they used.

For my benchmark, I wanted to ensure that I got an optimal performance out of SAM. Therefore, I benchmarked the performance of all 6 model building scripts in SAM 3.2.1. I found that the w0.5 script obtained the optimal performance and it is used in my benchmarks. The SAM software can be found at:

<http://www.cse.ucsc.edu/compbio/sam.html>

2.4.5 HMMER 2.3.2

This codebase can be found at:
<http://hmmer.janelia.org/>

2.4.6 Determination of Significant Performance Differences

In order to compare two different methods, 1000 query sets were generated by sampling with replacement from the original set of 2,521 queries, and the performance of each method was assessed on each re-sampled set. For each value of false positive per query, we calculated a 95% confidence interval on the difference in true positives detected by the two different methods on each of the 1000 query sets. This was accomplished by taking the $0.95 \times 1000 = 950$ middle values. i.e., eliminating 2.5% of the values at each tail of the distribution. Scripts used in this procedure are discussed further in appendix A/B.

2.4.7 Availability

All benchmarking scripts, alignments, and the test databases used in these experiments can be found one in a compressed tarball at:
<http://selab.wustl.edu/people/steve/thesis/bm.tar.gz>

Chapter 3

Information Content Based Sequence Weighting

3.1 Background

HMMER derives its match state emission probabilities from combining the observed residues with prior data regarding the residue types that commonly occur in multiple sequence alignment columns. This information comes from a 9 component Mixture Dirichlet prior [77]. Mathematically, the final emission probability of an amino acid i is represented as;

$$P(i) = \sum_{j=1}^l \text{Prob}(\vec{\alpha}_j | \vec{n}, \theta) \frac{n_i + \alpha_{j,i}}{|\vec{n}| + |\vec{\alpha}_j|},$$

where:

l is the number of components in the Mixture Dirichlet Prior.

$\text{Prob}(\vec{\alpha}_j | \vec{n}, \theta)$ is the probability of a component generating the observed alignment column.

The part that I want to focus on is

$$\frac{n_i + \alpha_{j,i}}{|\vec{n}| + |\vec{\alpha}_j|},$$

where:

n_i is the observed count of residue i in the alignment column.

$\alpha_{j,i}$ is the α parameter of residue i contributed by component j . This can be thought of as that component's pseudocount for residue i

$|\vec{n}|$ and $|\vec{\alpha}_j|$ are the total observed counts and total α parameters contributed by component j , respectively.

Therefore, the final emission probability of amino acid i , $P(i)$, is the weighted average of the above terms over all components in the mixture Dirichlet prior. An important point to note is that the number of counts added by the prior does not vary according to the number of observed counts in the column. Therefore, as the number of observed residue counts increases, the effect of the prior on the emission probability distribution is reduced and *vice versa*.

Sequence Weighting

An underlying assumption in the calculation of match state emission probabilities utilizing a Mixture Dirichlet Prior is that the training sequences are independent. Obviously, this is not the case. Homologous sequences share an evolutionary ancestry and thus are not independent of each other. A number of approaches have been devised to compensate for the dependency among homologous sequences. HMMs commonly use internal and external sequence weighting. Internal sequence weighting approaches are designed to correct for an over-representation of sequences from a particular family in a multiple sequence alignment [2, 25, 75, 90]. External weighting is used to estimate the total number of independent samples, or sequences, an alignment contains. For external weighting, HMMER uses a single linkage clustering at the $\geq 62\%$ level, similar to that used by Henikoff and Henikoff to build the BLOSUM series of substitution

matrices [33]. The number of clusters is then used as the number of effective sequences in the alignment.

3.1.1 Information Theory and Scoring Systems

In 1991, Altschul published an article analyzing local alignment and substitution matrices from an information theoretical perspective [1]. In it he points out that a given substitution matrix corresponds to a set of expected paired residue frequencies. For example, the PAM 40 matrix encapsulates the expected pair frequencies of proteins that are separated by an evolutionary distance of 40 mutations in 100 residues. In order to maximize the sequence information between two homologs, one needs to use a substitution matrix whose target frequencies match those of the homologs. Given aligned sequences of determined distance, Altschul calculated the average score per alignment position for a collection of substitution matrices of varied evolutionary distances (Table 3.1.1). From this information one can see that the maximum average score per alignment position is achieved when a PAM matrix with the appropriate target frequencies is used.

PAM Matrix employed	PAM Distance of Sequences							
	40	80	120	160	200	240	280	320
40	2.26	1.31	0.62	0.10	-0.30	-0.61	-0.86	-1.06
80	2.14	1.42	0.92	0.58	0.23	-0.02	-0.21	-1.37
120	1.93	1.39	0.98	0.67	0.42	0.22	0.06	-0.07
160	1.71	1.28	0.95	0.70	0.50	0.33	0.20	0.09
200	1.51	1.16	0.90	0.68	0.51	0.38	0.26	0.17
240	1.32	1.05	0.82	0.65	0.51	0.39	0.29	0.21
280	1.17	0.94	0.75	0.60	0.48	0.38	0.30	0.23
320	1.03	0.84	0.68	0.56	0.46	0.37	0.30	0.24

Table 3.1: Average score per alignment position using PAM matrices to score alignments between sequences of varying alignment distances.

Data taken from Altschul 1991. Bold values are the optimal mean score per alignment position for the sequences of the given distance.

3.2 Results

3.2.1 HMMER 2.3.2 Models Heavily Weight the Training Sequences

Based off of Altschul’s work, I hypothesized that HMMER’s local models were optimized for the detection of close homologs and not the more distant homologs present in the benchmarks. A useful measure that Altschul used in his study of substitution matrices was the relative entropy of the matrix. One could use the relative entropy of an HMM’s match state emissions as an estimate of that models’ optimal scoring distance. Therefore, I calculated the mean relative entropy, R_{mean} , of the individual match state emission probabilities of an HMM as;

$$R_{mean} = \frac{1}{M} \sum_{j=1}^M \sum_i^{20} q_i \log_2 \frac{q_{ij}}{p_i},$$

where:

M is the number of match states in the HMM.

q_{ij} is the probability of residue i given match state j .

p_i is the null model probability of residue i .

It is common for a multiple sequence alignment of a protein domain to have position-specific variability in its sequence conservation. Taking the mean relative entropy gives an overall sense of the evolutionary distance that the match state scores are optimized to detect (Figure 3.1).

It can be seen that the majority of the HMMER 2.3.2 models possess a mean match state relative entropy of approximately 1.5 bits. As a point of reference, the BLOSUM 100 substitution matrix has a relative entropy of 1.45 bits. This substitution matrix is ideal for identifying highly similar homologs. However, the benchmarking sequences are quite divergent. Thus, it seems that match state emission scores from default HMMER 2.3.2 local models were not being constructed to optimally search for more remote homologs. This also

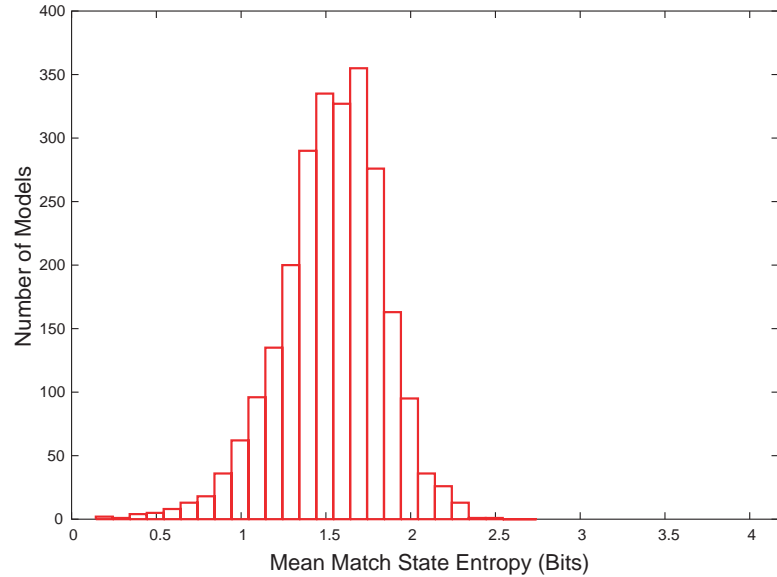


Figure 3.1: **Mean Relative Entropy of Match State Emissions**

The match state mean relative entropy was calculated for each of the HMMER 2.3.2 benchmark models.

provides an explanation as to why the glocal models were not as adversely affected. The alignment lengths of local models are dependent on information content, while glocal model alignments are required to align to the entire core of the HMM.

Due to these findings, I explored alternative external sequence weighting strategies. One alternative strategy, first implemented by Dr. Kevin Karplus, is to use to a model’s match state emission information content to determine external sequence weight [45]. I implemented my own approach of this strategy, which I describe later in this chapter. In order to discuss the details of this method, I will now briefly discuss two more concepts in information theory.

3.2.2 Entropy and Information Content

Dr. Claude Shannon defined entropy as a measure of “uncertainty” [74]. A simple scenario to explain this concept is the flipping of a coin. The two possible outcomes from flipping a coin are revealing a “heads” or a “tails”. Before flipping

a fair coin, since these two possibilities are equally likely, our “uncertainty” of the outcome is maximized. We have no idea how it will land. Now let’s say that there is a rigged coin that lands “tails” 99% of the time. Before flipping a rigged coin, our uncertainty about the outcome is reduced because we know the coin will probably land “tails”.

Mathematically, entropy can be represented as;

$$H(p) = - \sum_{i=1}^N p_i \log_2 p_i,$$

where:

N is the number of possible outcomes.

p_i is the probability of outcome i .

We can also use this same procedure to calculate the entropy of a match state emission probability distribution by summing over the emission probabilities of the individual residues. The information content of a sequence profile has been defined as the difference in entropy between the profile and a random distribution [73]. Therefore, the information content of a match state’s emission probability distribution is simply;

$$I = H_{Background} - H_{Match}$$

where:

$H_{Background}$ is the entropy of the null model residue probability distribution. This value is 4.07 bits.

H_{Match} is the entropy of the match state emission probabilities.

3.2.3 Information Content Based Sequence Weighting

The goal of this weighting scheme is to construct a model with a mean match state emission information content equal to or lower than a target value. This is accomplished by taking advantage of the fact that as the external

sequence weight is lowered, the effect of the contribution of the prior is increased and the mean information content of the match state emission distributions is decreased. It should be noted that my implementation will not increase the external sequence weight higher than the number of sequences in the multiple sequence alignment.

In my implementation, the initial external sequence weight is set to equal to the number of sequences in the alignment. The mean information content the model's match state emission probability distributions is calculated. If this mean is lower than the target value, the process is stopped and the external sequence weight is left as the number of sequences in the alignment. Otherwise, a binary search is performed to identify the external sequence weight needed to obtain within 0.01 bits of the desired mean information content [31]. The starting endpoints for this search is an external sequence weight equal to the number of sequences in the alignment and zero.

3.2.4 Optimization of Information Content Based Sequence Weighting

Given this weighting strategy, I identified the optimum information content for HMMER's models. It is highly probable there might be different optimal information contents for different protein domains. This would probably be due to the different evolutionary histories and distances present in these families. Since it isn't clear *a priori* what this value is for a particular domain, I use an information content that provides optimal performance over the 2,521 models in my benchmark. This value was identified by benchmarking the performance of models built with various information contents. Based on these results, I determined the optimal information content for local and glocal models to be 0.59 bits and 1.30 bits, respectively. It is interesting to note that the optimal information content identified for local models corresponds to a relative entropy that is equal to that of the commonly used BLOSUM62 substitution matrix.

3.2.5 Information Content-Based Sequence Weighting Improves HMMER Local Model Performance

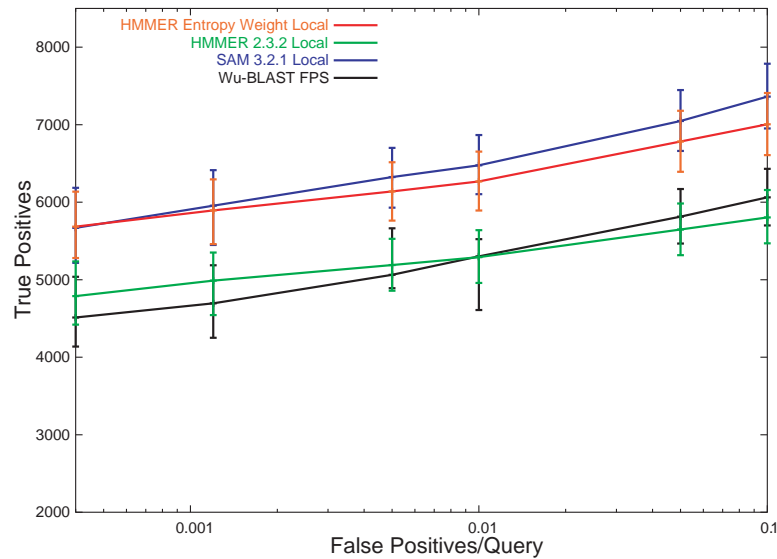


Figure 3.2: **Information Content Based Sequence Weighting. Local Model Performance**

Information content based sequence weighting significantly improves local model performance. Error bars represent the minimum and maximum true positives identified from bootstrapping.

To test whether this weighting strategy corrected HMMER’s local models, I compared the performance of local models built with information content sequence weighting to default HMMER 2.3.2 local models. The new sequence weighting approach appears to correct the problem with HMMER local models (Figure 3.2). The improvement in local model performance is statistically significant, using bootstrapping and a 95% confidence interval, over the complete range of false positives per query range.

3.2.6 Information Content Based Sequence Weighting Improves HMMER Glocal Model Performance

I also tested whether this weighting method would improve glocal model performance. While there did not seem to be a problem using the previous

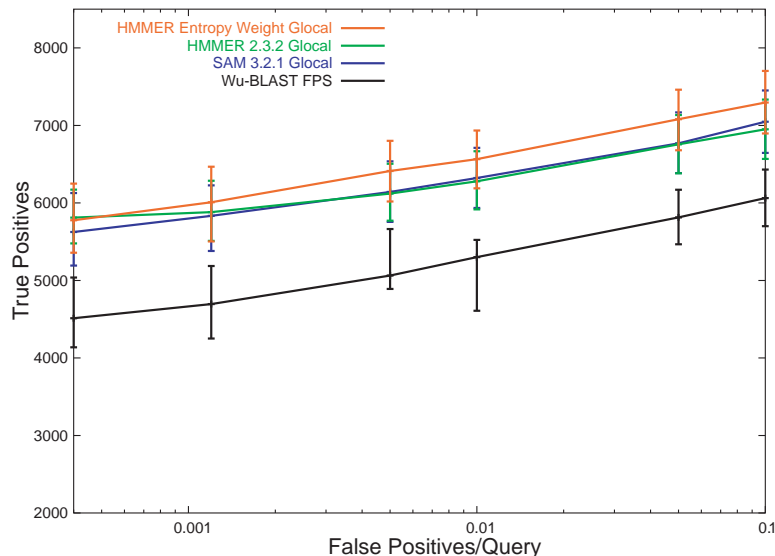


Figure 3.3: **Information Content Based Sequence Weighting. Glocal Model Performance**

Information content based sequence weighting modestly improves glocal model performance. Error bars represent the minimum and maximum true positives identified from bootstrapping.

sequence weighting method on glocal models, the new weighting also seems to give these models a slight boost in performance (Figure 3.3). Information content weighted glocal models detect a significantly greater number of trues than the HMMER 2.3.2 glocal models at false positives per query levels > 0.001 .

3.2.7 SAM 3.5 Demonstrates Improved Performance Over HMMER

During the course of this research, SAM released a newer version of their software. I performed an additional benchmark using the newest version of SAM, version 3.5. While my sequence weighting method results in a definite improvement over HMMER 2.3.2, HMMER is still lagging behind the latest version of SAM (Figure 3.4A, 3.4B). SAM 3.5 detects more trues than HMMER for both types of models. For glocal models, this is statistically significant at all

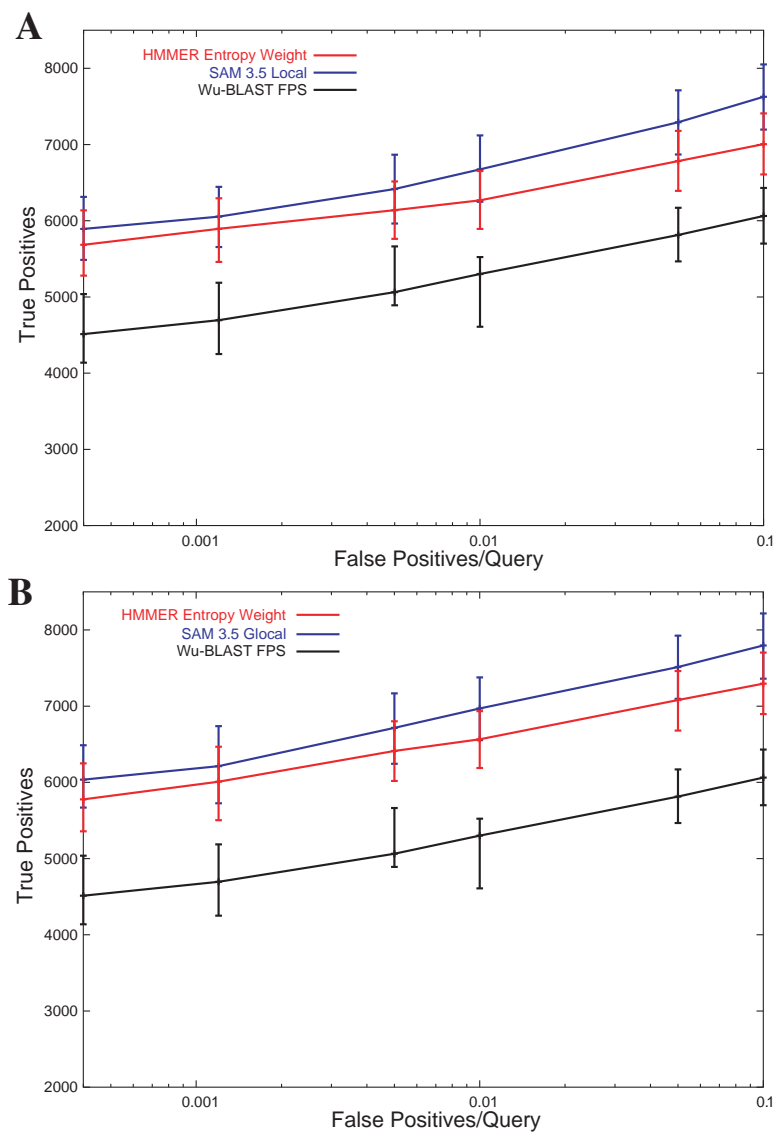


Figure 3.4: **SAM 3.5 Performance**

The latest version of SAM shows improved performance over HMMER even with the new sequence weighting scheme. Error bars represent the minimum and maximum true positives identified from bootstrapping. **A)** Local Model Performance **B)** Glocal Model Performance.

error rates, For local models, it is only statistically significant for local models at false positives per query of ≥ 0.01 .

3.3 Concluding Remarks

Measuring the mean relative entropy of HMMER 2.3.2 models revealed that the overall emission probabilities were not optimal for identifying local alignments between distant homologs. The apparent over-weighting most likely had more of a negative impact on local models because their alignment lengths are dependent on information content. However, glocal model alignments are required to align to the entire core of the HMM. Given this problem, information content seemed an appropriate measure to determine the external sequence weight of HMMER HMMs. In my implementation of information content based sequence weighting method, I raise/lower the sequence weight until the average information content of the match state emissions reaches a desired target.

This sequence weighting method improves HMMER local model performance relative to both HMMER glocal models and SAM. Additionally, there is a modest improvement of HMMER glocal models. It should be noted that the optimal information content will undoubtedly vary according to protein domain family, but since it is not currently possible to know this value *a priori*, I determined a value that is optimal across many different domains.

Internal and external sequence weighting strategies are used to try and correct for the fact that biological sequences violate independence assumptions. While information content sequence weighting improves HMMER performance, there is no probabilistic justification for this approach. An alternative strategy would be to utilize a probabilistic phylogenetic approach in building the HMMs [41, 76]. These approaches have difficulty in modeling insertions and deletions, however this is a currently active area of research [57, 68].

3.4 Methods

3.4.1 Optimization of Information Content

Local and glocal models were built and benchmarked using target mean information contents of 0.1 to 2.0 bits at 0.1 bit intervals. The 0.1 interval that provided the optimal performance was further explored using models built at a 0.01 bit interval.

3.4.2 Optimization of SAM 3.5

In SAM 3.5 they eliminated many of the model building scripts that were present in the previous version. I benchmarked models built with each of the scripts in SAM 3.5, w0.5, w0.7, and w1.0. In SAM 3.5, models built with w0.5 still perform the best on my benchmark. In addition, SAM 3.5 now utilizes calibrated E-values. Calibration was thus performed for these models. The SAM software can be found at:

<http://www.cse.ucsc.edu/compbio/sam.html>

3.4.3 Availability

My implementation of information content based sequence weighting, hmmer-2.4dev1, can be found combined with the benchmarking scripts, alignments, and test database used in this chapter at:

<http://selab.wustl.edu/people/steve/thesis/bm.tar.gz>

Chapter 4

Forward Algorithm

4.1 Background

When using HMMs for remote protein homology what one wants to know is $P(Seq|HMM)$, or the likelihood that the model generated the sequence. This can be calculated using the Forward algorithm. Despite this, the Viterbi algorithm is routinely used to calculate the score of a target sequence given an HMM. The Viterbi algorithm is a dynamic programming algorithm that calculates the $P(Seq, \hat{\pi} | HMM, \beta)$, where $\hat{\pi}$ is the optimal path over all paths π , and β is the parameterization of the HMM [67]. There are several reasons why Viterbi is used in HMMER. One reason is that, due to HMMER's use of log odds scores, the Forward implementation is considerably slower than Viterbi. In order to avoid possible underflow errors caused from the multiplication of small probabilities, HMMER uses log odds scores. Thus, the summation that occurs in the Forward algorithm requires log conversions at each step, which considerably slows down the algorithm. The Viterbi algorithm is the functional equivalent to the Smith-Waterman and Needleman-Wunch algorithms. Using the Viterbi algorithm allows for the identification of the most likely state path or alignment between the model and target sequence. A common assumption is that the most likely path is the dominate factor in the $P(Seq|HMM)$ and can be used as an accurate approximation of this value. Additionally, the distribution of the

Forward null scores is currently unknown. Thus, there is no foundation for the calculation of P-values for Forward scores.

Forward Score Statistics

In 1990, Karlin and Altschul published a study regarding the distribution of optimal scores from ungapped sub-sequence alignments [13, 14, 43]. This work showed that given certain assumptions, local alignment score distributions approach a Gumbel distribution. Their work provided a solid mathematical foundation for the calculation of accurate P-values for local alignment scores. It is important to note that there were several assumptions made in their work. One that I would like to point out was that their work was restricted to ungapped local alignments.

Many commonly used algorithms, such as Smith-Waterman, BLAST, and Viterbi, are used in identifying high scoring gapped local alignments. Even though these methods employ the use of gaps, Karlin-Altschul statistics are still commonly used to assign P-values to the local alignment scores from these methods. One justification is the assumption that, due to the selection of highest local alignment scores, gapped alignment score distributions should approach a Gumbel. In fact, empirical studies have shown that the scores from optimal gapped local alignments do appear to follow the Gumbel distribution [3].

Forward scores are calculated in a completely different fashion than these algorithms. Instead of identifying the highest scoring local alignment, the Forward algorithm calculates the summed probability of the sequence given all possible alignments. Thus, there is no selection of a maximum score in this algorithm. While one can imagine that the Forward algorithm might be more sensitive than the Viterbi algorithm, there is a serious concern that Forward scores do not follow a Gumbel distribution. In fact, this has been confirmed in the work of Yu and Hwa [95]. This severely limits this method's usefulness in remote homology detection.

In 2001, Yu and Hwa developed a 'semi-probabilistic' alignment approach. Their method is a hybrid of the Smith-Waterman and the Forward algorithms. In essence, a Forward matrix is calculated and then one takes the log of the

maximum score of the matrix. They go on to show that the maximum scores of Forward matrices can be approximated by the Gumbel distribution and thus provide a basis for calculating P-values. They demonstrate that the performance of this algorithm is comparable to the Smith-Waterman algorithm [94]. This avenue of research was explored by other members of the Eddy lab. However, a better understanding of local Forward null scores was still desired.

In 2005, Karplus *et al.* published an extensive description of the Forward null score distribution using the SAM HMM package [46]. They fit their Forward scores to a variety of different known numerical distributions. They claim there is an adequate fit between their Forward null scores and an *ad hoc* two parameter sigmoidal distribution. However, there are several differences between SAM and HMMER HMMs. In particular, SAM utilizes a reversed sequence null model. It is unclear exactly what the effects of this null model have on the Forward score distribution.

4.1.1 Example of Forward Algorithm Sensitivity

My initial interest in the Forward algorithm came from analyzing the results from a handful of models. I will discuss the results from one of these models to explain why I thought the Forward algorithm might be worth exploring.

d1bqca_ is one of 33 members of the glycosyltransferase Superfamily. The top 5 scoring sequences returned from a Viterbi search of the test database using an information content weighted model of d1bqca_ are from the glycosyltransferase Superfamily with scores ranging from 373 bits to 12.5 bits. The 6th and 7th score, 3.7 and 3.6 bits, come from a sequence from a different SCOP Fold and a shuffled sequence, respectively. When the Forward algorithm was used to score the same model against the test database, the same top 5 homologs were identified. However, the 6th top score was now from another member of the glycosyltransferase Superfamily, d1cz1a_. The Viterbi score for this sequence is 0.7 bits and the Forward score is 17.5 bits. It should be noted that there are obvious difficulties in comparing Viterbi and Forward given the differences in the algorithms. However, for those homologs that are very similar

to the HMM, the Viterbi path is a dominant factor in the Forward score. I thought this score increase given the jump in rank for this sequence was intriguing.

The results from d1cz1a_ led me to examine this sequence more closely. I suspected that there may be additional regions of homology between this HMM and target sequence that may contribute to the increase in the Forward score relative to Viterbi. To test this, I utilized code written by Dr. Ian Holmes to determine the posterior probability matrices of this HMM and target sequence. From this I obtained the posterior probabilities that a particular match state of the HMM emitted a particular residue of the sequence, given all possible paths of the sequence through the model. In effect, this matrix gives me a map of the high probability alignment segments between the HMM match states and the sequence.

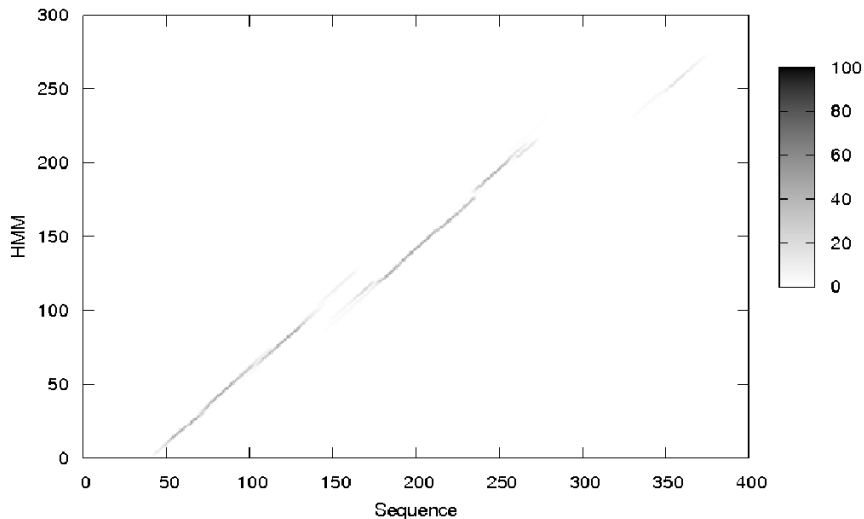


Figure 4.1: Match State Posterior Probability Matrix of d1bqca_ HMM and the d1cz1a_ Sequence

The match state posterior probability matrix for d1bqca_ and d1cz1a_ reveals several highly probable alignment segments.

Figure 4.1 is the match state posterior probability matrix between the d1bqca_ HMM and the target sequence d1cz1a_. The Viterbi path for this sequence is an alignment between the d1bqca_ HMM and d1cz1a_ at position 53 to 130 in d1cz1a_. This region is clearly detectable in the match state posterior probability matrix. There is also an additional area of high posterior probability located at positions 175 to 275 and a minor region centered around position 350 in the target sequence. It should be noted that this is but one example, I'm sure there are many alternative scenarios that could lead to Forward being a more sensitive algorithm.

4.2 Results

4.2.1 Fitting Local Forward Null Score Distributions

At this junction, the work on the Forward null score distribution was the combined efforts of several individuals in the Eddy lab. In particular, Dr. Eddy wrote an extensive package of C code, EASEL, that was capable of fitting known numerical distributions to a distribution of observed scores by finding the maximum likelihood distribution parameters using conjugate gradient descent. I used this code to investigate the fit between the observed Forward null score distributions of several models to known numerical distributions.

The basic design of these fitting experiments were to generate a set of Forward scores from comparing 5,000 random sequences to an HMM. This null score distribution was then used to find the maximum likely set of parameters for the distribution being examined. Then, a larger set, 500,000, of local Forward null scores are compared to the distribution with the previously determined parameters.

4.2.2 1% Tail of Forward Null Score Distributions Fit an Exponential Distribution

I focused my work on finding a known numerical distribution that approximated the right-most tail of the Forward null score distribution. This region of the score distribution corresponds to those scores with lower probability and are of the most interest. One distribution that gave promising results was the exponential distribution. An example of one of these fits is shown in Figure 4.2. It should be noted that our fitting results confirm the findings of Yu and Hwa [95]. The Gumbel distribution does not provide an adequate approximation of HMMER's Forward scores (Figure 4.2).

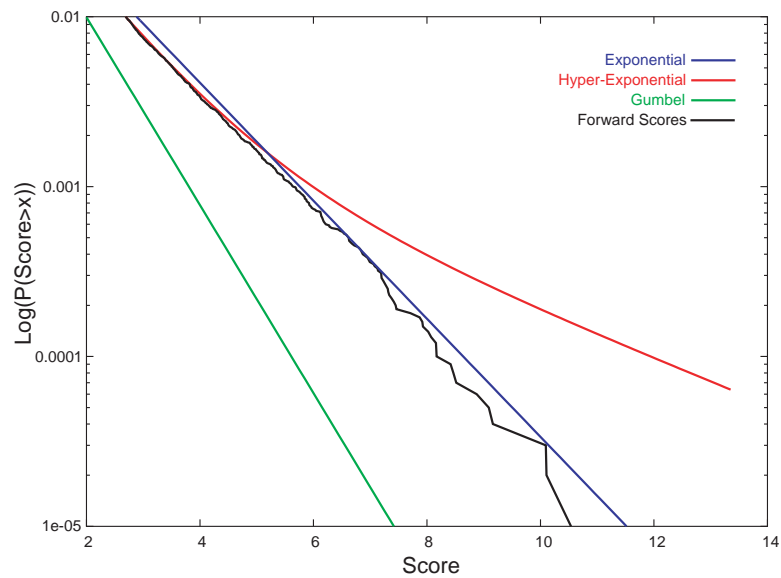


Figure 4.2: **Fits to the 1% Tail of the d1b72a_ Forward Null Score Distribution**

The 1% tail of the d1b72a_ Forward null score distribution fitted to the Gumbel, hyper-exponential, and exponential distributions.

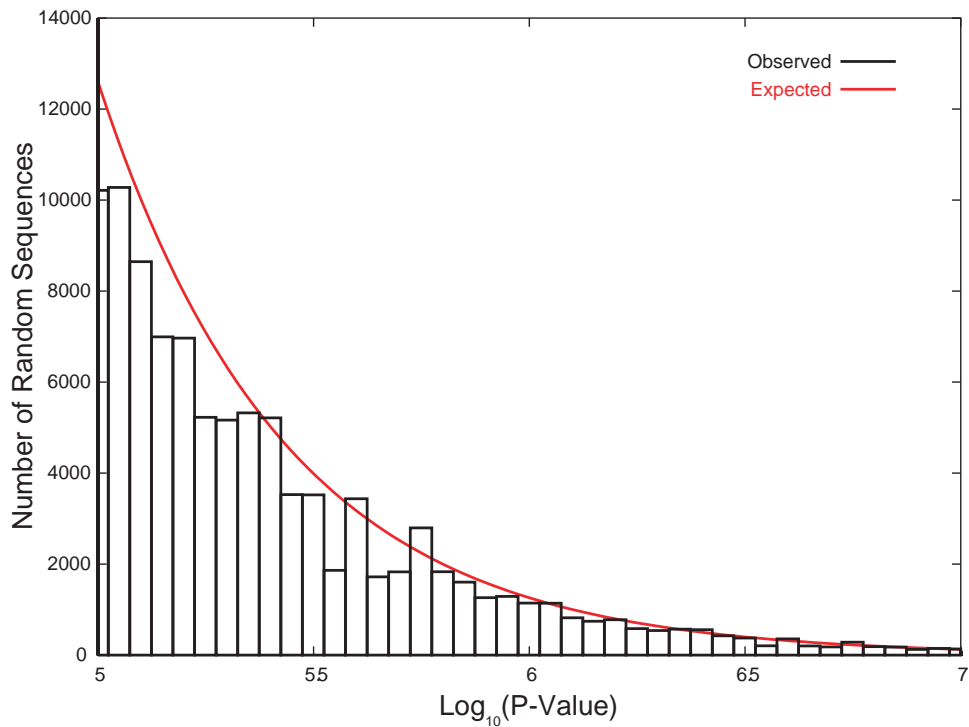


Figure 4.3: **Robustness of Exponential Based P-Values**
 Expected versus observed number of sequences with the given P-value.

4.2.3 The Exponential Distribution Provides Robust P-Values

Given the exponential fitting results, I tested whether the exponential distribution could provide robust P-values for HMMER’s Forward local scores. I implemented a version of HMMER 2.5 that utilized the exponential distribution for the calculation of the P-values of Forward scores. I modified HMMER’s `hmmcalibrate` to identify the exponential μ parameter for the 1% tail of 5000 random Forward local scores. λ is set to 0.693 based off a proof by Dr. Eddy.

Given these parameters, the P-value for a Forward score, $P(x)$, is calculated as;

$$P(x) = (e^{-\lambda(x-\mu)}) * t,$$

where:

t is the tail fraction of the score distribution used to estimate the exponential parameters. e.g. 0.01

λ and μ are the exponential distribution parameters.

I used this implementation to score the 2,521 benchmarking local models against a database of 500,000 random sequences. Figure 4.3 is a histogram of the observed versus expected number of sequences with a given P-value. It does appear that the 1% tails of our benchmarking models' Forward null score distributions can be adequately approximated by the exponential distribution. It should be noted that a chi-square goodness-of-fit test does fail a 0.05 significance test. However, I am not particularly troubled by this. First, P-values are routinely calculated for optimal local alignments that violate one or more of Karlin and Altschul's constraints. The most common being the use of gaps. The null score distributions of commonly used programs, such as NCBI's BLASTP and Psi-BLAST, are known to empirically approximate a Gumbel distribution and provide accurate statistics. However, these programs fail 0.05 tests of goodness-of-fit [3]. As an example, I ran Psi-BLAST on the same database described above and assessed its goodness-of-fit with chi-square. It also failed a 0.05 test of significance. Our main focus is to identify a known numerical distribution that approximates our Forward null score distribution in a robust and sensitive way. The exponential distribution accomplishes this goal.

4.2.4 HMMER Local Forward Searches with Exponential P-Values are an improvement over Local Viterbi

Now that it had been shown that the exponential distribution could be used to provide reliable P-values, I tested whether the Forward algorithm was more sensitive than the Viterbi algorithm by measuring the performance of local Forward searches using exponential based P-values and local Viterbi searches on my benchmark (Figure 4.4). Utilizing the Forward algorithm with exponential based P-values gives HMMER a significant performance boost over local Viterbi.

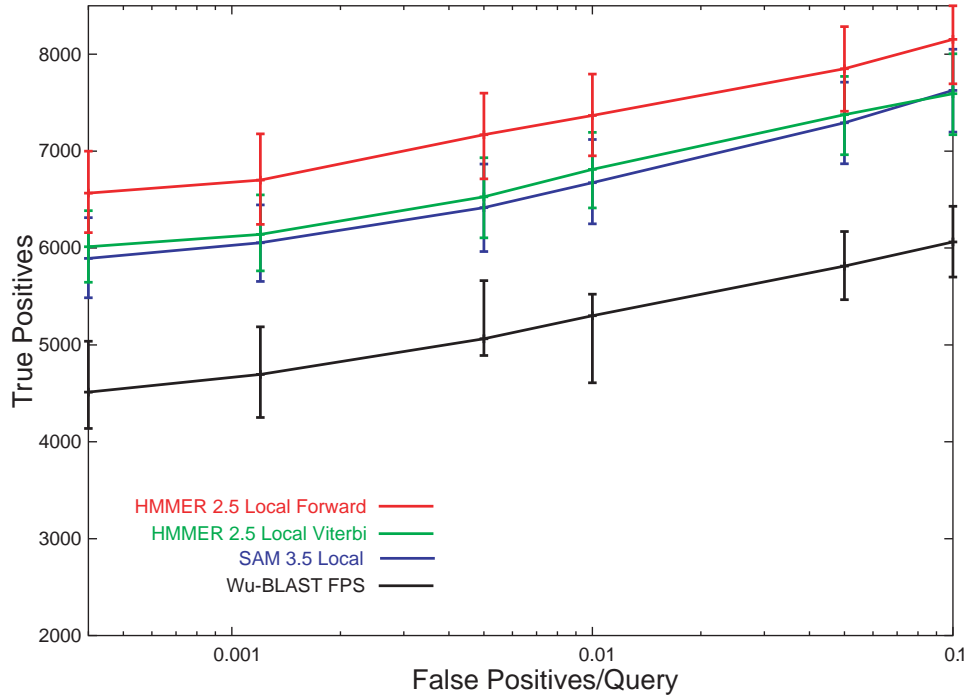


Figure 4.4: **Performance Benchmark of Local Forward Utilizing Exponential Based P-Values**

HMMER 2.5 local Forward outperforms HMMER 2.5 local Viterbi. Error bars represent the minimum and maximum true positives identified from bootstrapping.

There is an overall average increase of approximately 500 trues and is statistically significant across all the false positive per query levels.

4.3 Concluding Remarks

After several observations of the Forward algorithm’s increased sensitivity relative to the Viterbi algorithm, I worked in collaboration with Drs. Sean Eddy and Alex Coventry on the Forward null score distribution. I measured the ability of known numerical distributions to approximate the 1% tail of the Forward null score distributions of several models. I observed a good approximation by the exponential distribution to this region of the Forward null score distribution.

I implemented exponential based P-values for local Forward scores in HMMER 2.5 and tested their robustness by scoring the local benchmarking models using Forward against a random database. Exponential based P-values demonstrate a good agreement between expected and observed number of sequences with a given P-value.

I then tested the performance of the Forward algorithm relative to the Viterbi algorithm on my benchmark to determine whether the Forward algorithm was more sensitive. The Forward algorithm outperforms the Viterbi algorithm on local models by approximately 500 more trues across all false positive per query levels.

One area of future work would be to determine if a known numerical distribution can approximate the Forward glocal scores, and whether this is a more sensitive approach than Viterbi glocal. Additionally, HMMER Forward searches are considerably slower than Viterbi searches. The Viterbi algorithm could be used as an initial search. Those sequences that pass a liberal significance cutoff could then be passed on to the Forward algorithm. Additionally, myself and a collaborator have worked out a search heuristic that significantly improves HMMER's search speed. This is discussed in more detail in the next chapter.

4.4 Methods

4.4.1 Posterior Probability Matrix

The code that I used to generate the posterior probability matrices was `heatmap.c`. It heavily uses functions written by Dr. Ian Holmes to calculate the posterior probability matrices given the Forward and Backward matrices of scores between an HMM and a target sequence. Dr. Holmes code can be found in `postprob.c` in the HMMER 2.5 version.

4.4.2 Fitting of Forward Null Score Distributions

Forward Null Score Distribution Used to Fit a Distribution

The score distributions I used to determine the maximum likely parameters of the distribution I was studying consisted of 5,000 scores of random sequences of length 100. This was generated using Dr. Eddy's sre-vtest or a modified version of this original.

Forward Null Score Distribution Used to Assess the Fit of a Distribution

The score distribution that I used to test the fit of the distribution consisted of 500,000 scores of random sequences of length 100. These scores were generated by Dr. Eddy's sre-calibrate.

Assessment of the Distribution Fit

The survival function of the fitted known distribution was plotted against the 500k score distribution to visually reveal the quality of the fit. A chi-square statistic was also calculated for the fit between the two distributions. This data was also generated using Dr. Eddy's sre-vtest or a slight modification of this code.

4.4.3 HMMER 2.5

Improvement of HMMER 2.5 Local Viterbi Relative to HMMER 2.4

The HMMER version used in these experiments is HMMER 2.5. It should be noted that the performance of HMMER 2.5 local Viterbi in Figure 4.4 has improved relative to HMMER 2.4. This is primarily due to several bug fixes and the implementation of target length dependent transitions by Dr. Sean Eddy.

4.4.4 Availability

My implementation of HMMER 2.5 with exponential based P-values can be found combined with the benchmarking scripts, fitting programs, alignments, and test database used in this chapter at:

<http://selab.wustl.edu/people/steve/thesis/bm.tar.gz>

Chapter 5

HMM Search Heuristic: HMMERHEAD

5.1 Background

The Smith-Waterman algorithm is a modification of the Needleman-Wunsch algorithm that allowed for the identification of the optimal alignment between two sub-sequences [78]. Given a scoring scheme and two sequences, the algorithm calls for filling in a $L \times M$ matrix, where L and M are the lengths of the sequences, which encompasses all possible alignments of the two sequences. However, due to the fact that each sequence comparison required filling in this matrix, this type of search required several hours to perform using the hardware and databases of the time [36, 53]. Therefore, shortcuts that approximated the sensitivity of the Smith-Waterman algorithm were explored [4, 53, 92].

An important difference between these new methods and the Smith-Waterman algorithm is that they are heuristics. A heuristic is a method that commonly gives a correct answer but is not guaranteed to do so. One aspect that is shared among these different heuristics is the matching of “words”. Using different approaches, a list of “words”, of user-defined length, are generated from the query sequence. Exact matches to these words are then searched for in the

sequence database. Exact string matching algorithms are a well researched area of computer science and there are algorithms that permit extremely rapid scanning for these words [4, 31].

One of the first local similarity search heuristics was published in 1983 by Wilbur and Lipman [92]. Their method consisted of identifying identical words of user-defined length between the query and the target sequence. They then determined the offsets or diagonals between the two sequences that would produce a significant number of identical paired residues while minimizing the number of mismatches over a set window of sequence. For those windows whose scores were considered significant, their method then performed a banded Needleman-Wunsch alignment. Two major weaknesses of this method were that they focused on identical residues between two sequences and that all identical residue pairs were scored the same regardless of their prevalence in the sequence database. In other words, aligned pairs of rare residues such as tryptophan received the same score as more common pairs, like alanine.

In 1985, Lipman and Pearson published the FASTP heuristic which built off of the previous study with some important improvements [53]. This method also identifies high scoring offsets of words. One important difference was that it re-scored these regions with an amino acid substitution matrix, PAM250, instead of relying solely on identity. The score from the best sub-sequence alignment then represented the similarity between the two sequences. When FASTP was searched against a sequence database, it kept track of the scores and used the subsequent histogram to calculate z-scores as a measure of the statistical significance of the similarity scores. The sub-sequence regions that have a high rank compared to the other database sequences are then globally aligned using a modified Needleman-Wunsch algorithm.

The commonly used BLAST heuristic was first published by Altschul *et al.* in 1990 [4]. This approach, combined with its modifications, has proved to be the most widely accepted pairwise sequence comparison heuristic. This method also consists of identifying words, finding identical matches to these words in the target sequence, and identifying high scoring regions around these words. However, there are subtle important differences. In BLAST, words were identified that would score above some threshold when compared to the query

sequence using a given substitution matrix. Identical matches to these selected words were then scanned for in the sequence database. When a word match is identified, the alignment between the query and the target is extended from the word edges, in an ungapped fashion, until the alignment score drops below some threshold amount from the best score identified. An additional feature present in the BLAST algorithm was the incorporation of the work of Karlin and Altschul on the distribution of optimal sub-sequence alignment scores for random sequences [43]. While the usage of gaps violates one of the assumptions of their work, numerous studies have verified the empirical observation that gapped local alignments seem to follow the same distribution [3]. The BLAST family of algorithms have been modified through the years to incorporate gaps, use a two word hit filter, and handle low-complexity sequences [4, 5].

BLAT is a tool that was designed primarily for identifying homologous mRNA/DNA regions but can also be used in protein homology detection [48]. The most important difference in BLAT is that it preprocesses the database into an indexed set of non-overlapping words. Thus, when a search is performed, each of these words are searched for in the query. This is a much easier task than searching for a query word in a large database and is attributed to the algorithm's 50X speedup versus Wu-TBLASTX.

A relatively recent area of study has been in the use of non-consecutive word hits or "spaced" seeds. As described previously, algorithms like BLAST use words of k consecutive letters to identify potential regions of homology between two sequences. Ma *et al.* observed improved results if the conserved positions were spaced out [55]. In their terminology, a standard NCBI BLASTN 11-mer word is represented as "11111111111" where 1's represent positions in the word that must exactly match between query and target. They found that in aligning the *E. coli* and *H. influenza* genomes the "spaced" 11-mer seed, "111010010100110111", was able to obtain equal sensitivity to BLASTN at 20X faster speed and 0.1 the memory requirement. Several groups have been exploring this area and empirically identifying optimally spaced words. A potential limitation to this approach is that it appears the optimal spaced seed varies for searches at different levels of similarity. There is a growing interest in the *a priori* design of optimal words but this is still a new area [10, 86, 87].

An alternative approach is to increase search speed through the use of parallelizing the search process. The basic idea of parallelization is to take a problem made up of repetitive tasks and have each of the tasks be performed separately and at the same time. Most parallelization approaches either take advantage of specialized instruction sets or multiprocessor hardware. This type of parallelization has been attempted for both pairwise sequence algorithms and HMMs [39, 60, 69, 93]. The potential speedups for HMM Viterbi are considerable. Some authors claim 10-100 fold speed improvements, although there are serious concerns regarding many of these approaches. In addition to the lack of general availability of the hardware involved in some of these approaches, another drawback is that these methods involve altering the HMM structure in order to efficiently parallelize the search process. Removal of the HMMER J multi-hit state and the omission of calculating a Viterbi traceback are two areas of most concern. Most authors dismiss the traceback procedure as being a minor compute concern. However, the match and insert emissions used in the traceback, or Viterbi path, are averaged to obtain a target sequence composition specific null model. This is then used to correct any compositional bias in the Viterbi score. Most of these studies also lack benchmarks to assess any sensitivity and specificity changes their implementations may have caused.

5.1.1 HMM Scoring Algorithms

Currently, HMMER utilizes the Viterbi and Forward scoring algorithms to identify significant database sequences. The Viterbi algorithm used on a local model is the HMM functional equivalent of the Smith-Waterman algorithm. Like Smith-Waterman, the Viterbi and Forward algorithms require filling in a $L \times M$ matrix for each HMM/sequence comparison. Also like Smith-Waterman, given the hardware and database sizes of today, they are inconveniently slow. Scoring an HMM against Genbank's NR using the Forward algorithm takes approximately 5 hours. The goal for this research is to develop a heuristic HMM scoring algorithm to reduce the amount of time required to score an HMM against a large sequence database.

5.2 Results

5.2.1 HMMERHEAD Algorithm

My colleague Elon Portugaly and I developed a heuristic HMM scoring algorithm, HMMERHEAD. Our approach consists of applying numerous filtering steps to a perspective homologous sequence. If a sequence passes all of the filtering steps, it is then presented to a full HMM scoring algorithm. e.g. the Viterbi algorithm. We tested a variety of filtering steps and measured their speedup benefit over unfiltered Viterbi and Forward scoring for the models in the benchmark. We were also able to use the benchmark to measure the sensitivity loss due to the various filters and their different parameter settings. Based on the benchmark and timing results, we have designed a series of filtering steps and parameter settings that achieve a substantial speedup relative to the Viterbi and Forward scoring algorithms with a minimum of sensitivity loss.

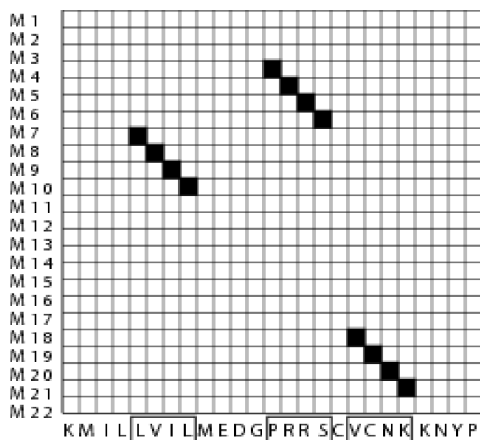


Figure 5.1: HMMERHEAD: Word Hit Identification

'Word' Generation and Identification

The first step consists of identifying four residue 'words' from an HMM that possess a score above some threshold, θ . These words come from the match state emission scores. All possible words in sliding four match state windows are generated from the HMM. Those words with scores above θ are then identified in

the database sequences using a deterministic finite automaton. Overlapping words are merged into a single word (Figure 5.1).

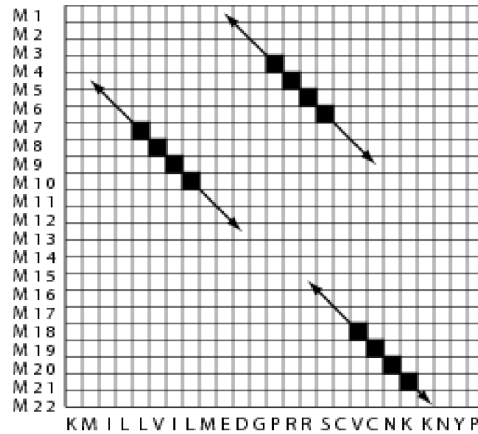


Figure 5.2: HMMERHEAD: Optimal Scoring Ungapped Word Extension

Ungapped Word Extension

Each word hit is extended in an ungapped fashion in each direction until its score has dropped δ below the optimum score. Specifically, a word hit is extended from one end until the optimum score has dropped $\frac{\delta}{2}$ below the optimum; then the other end is extended. Those ungapped extensions whose optimum score, at any point, is greater than μ are retained (Figure 5.2).

Two Word Hit Filter

The next filtering step involves identifying those database sequences that have at least two word hits that passed the μ threshold. If these two hits are within 5 diagonals and the closest ends of these extended words are within 25 residues along the target sequence, they pass this filtering step. If a merged word hit is at least 8 residues long, it will pass this step and its score will be compared to η . (See below.)

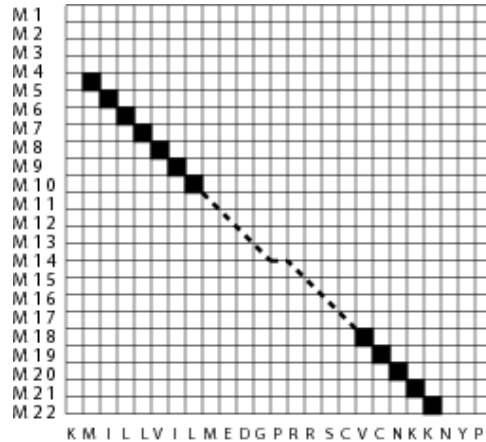


Figure 5.3: HMMERHEAD: Optimal Scoring Gapped Alignment

Gapped Alignment of Extended Words

The final step consists of performing gapped alignments between the remaining word hits (Figure 5.3). The two closest ends of the extended word hits are trimmed back to the original hits and gapped Viterbi is performed between these ends. If the sequence contains a gapped alignment between two extended word hits with score greater than η , the sequence is then passed on to the full scoring algorithm, which is either the Viterbi or Forward algorithm. For reference, when I refer to HMMERHEAD Viterbi, this indicates that Viterbi is the full scoring algorithm performed on those sequences that passed all of the filtering steps.

5.2.2 HMMERHEAD is Faster Than Default Scoring Algorithms

In order to test the speed and sensitivity of HMMERHEAD relative to the Viterbi and Forward algorithms, we needed to determine the optimal values for HMMERHEAD's different thresholds. We were able to take advantage of the benchmark to test several different values for each of the thresholds and evaluate their impact on these performance measures. We identified thresholds with a

HMNERHEAD Timing Results

Model	Algorithm	θ	δ	μ	η	Fold Speedup
Local	Forward	6000	2000	7000	20000	18,288s/767s = 24X
Local	Forward	7000	2000	8000	22000	18,288s/676s = 27X
Local	Viterbi	6000	2000	7000	20000	4,612s/724s = 6X
Local	Viterbi	7000	2000	8000	20000	4,612s/657s = 7X
Glocal	Viterbi	7000	2000	8000	22000	4,404s/725s = 6X
Glocal	Viterbi	8000	2000	9000	22000	4,404s/659s = 7X

Table 5.1: **HMNERHEAD** parameter settings and corresponding speed improvements for the different model and search types.

Fold speedup is measured as the mean default algorithm search time/mean HMNERHEAD search time. Bold parameter settings are those that achieve maximum speedup with a minimum loss in sensitivity relative to the default algorithm. (See section 5.2.3)

minimum loss in sensitivity from the default HMM scoring algorithms that still maximized the heuristic’s speedup (Table 5.2.2).

HMNERHEAD Forward is approximately 24X faster than default HMNER 2.5 Forward with a minimum loss of sensitivity. HMNERHEAD Viterbi is approximately 6X faster. The reason for the fold speedup difference between HMNERHEAD Forward and HMNERHEAD Viterbi is due to the fact that the Forward algorithm runs considerably slower than the Viterbi algorithm. Again, this is due to the fact that HMNER’s Forward algorithm requires log conversions at each summation step. Using a typical sized HMM and CPU, HMNERHEAD Viterbi and Forward takes approximately 12 minutes to search a database the size of Genbank’s NR. Default HMNER 2.5 Forward and Viterbi algorithms take approximately 5 and 1.2 hours, respectively. As demonstrated in Table 5.2.2, HMNERHEAD can run faster with more stringent threshold settings, although, this comes at the cost of decreased sensitivity, as seen in the performance benchmarks in section 5.2.3.

5.2.3 HMNERHEAD’s Loss in Sensitivity is Minimal

HMNERHEAD’s gains in search speed need to be balanced against any loss in sensitivity. Therefore, for each parameter setting, we measured

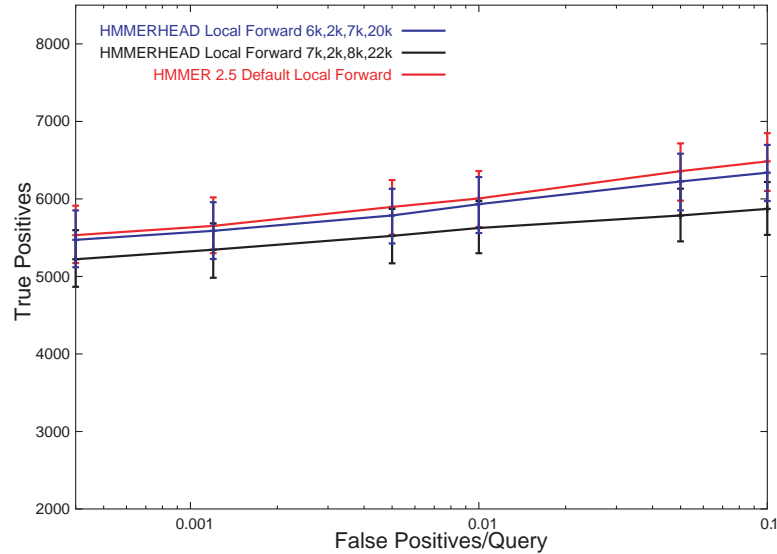


Figure 5.4: **HMMERHEAD Local Forward Performance**

HMMERHEAD local Forward 6k/2k/7k/20k has a minimal loss in sensitivity. While HMMERHEAD local Forward 7k/2k/8k/20k obtains a faster search speed, it displays a greater loss in sensitivity. Error bars represent the minimum and maximum true positives identified from bootstrapping.

HMMERHEAD’s performance on the benchmark relative to default Viterbi and Forward. HMMERHEAD demonstrates a minimum loss of sensitivity while still obtaining a considerable speedup relative to default HMMER 2.5 Forward and Viterbi scoring algorithms. The HMMERHEAD Forward parameter settings of 6000, 2000, 7000, and 20000 achieve a 24X speedup with an average loss of 50 true positives across the range of false positives per query. This represents a 0.1% loss in sensitivity (Figure 5.4). The HMMERHEAD Viterbi settings of 6000, 2000, 7000/8000, and 20000/22000 achieve an approximately 6X speedup with local and glocal models with effectively the same level of sensitivity as default Viterbi (Figures 5.5 and 5.6). The slight variations in optimal local versus glocal HMMERHEAD Viterbi settings are due to the fact that local and glocal models have different information content weighting optimums and thus different model scores. It can be seen in these experiments that the faster threshold settings demonstrate an increased loss in sensitivity (Figures 5.4, 5.5, and 5.6).

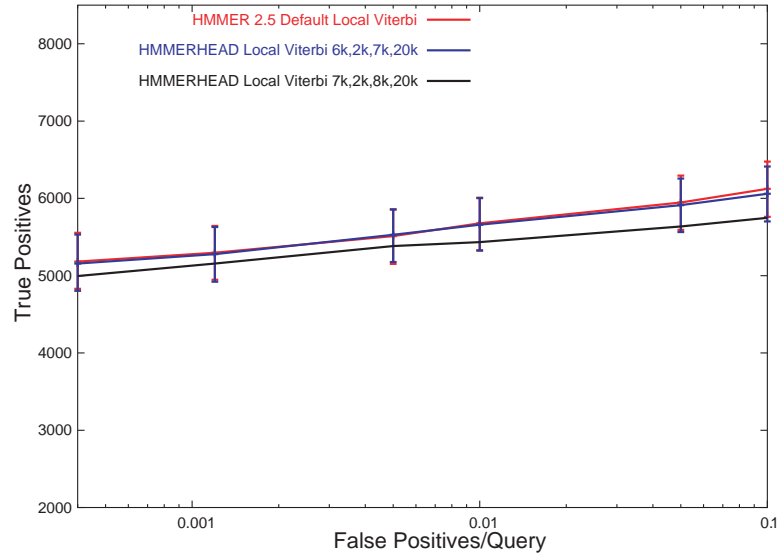


Figure 5.5: **HMMERHEAD Local Viterbi Performance**

HMMERHEAD local Viterbi 6k/2k/7k/20k identifies a comparable number of trues as the Viterbi algorithm. While HMMERHEAD local Viterbi 7k/2k/8k/20k obtains a faster search speed, it displays a greater loss in sensitivity. Error bars represent the minimum and maximum true positives identified from bootstrapping.

5.3 Concluding Remarks

The goal for this research was to design a heuristic scoring algorithm for HMMs that reduced the considerable amount of time required to do searches of large databases. My colleague Elon Portugaly and I combined several filters that had been applied in pairwise sequence comparison heuristics for use in HMM scoring. Our approach, HMMERHEAD, consists of identifying high scoring words in database sequences, performing ungapped extension of these words, applying a two word hit filter, and performing a gapped alignment between those word hits that passed the previous filtering steps. If a sequence passes all the filtering steps, the full Viterbi or Forward algorithm is then performed.

HMMERHEAD is able to obtain a 24X and 6X search speedup relative to default Forward and Viterbi, respectively. This is achieved while maintaining 99.9% of the sensitivity of the default algorithms as measured on my benchmark. Using the parameter settings that achieve this performance, HMMERHEAD

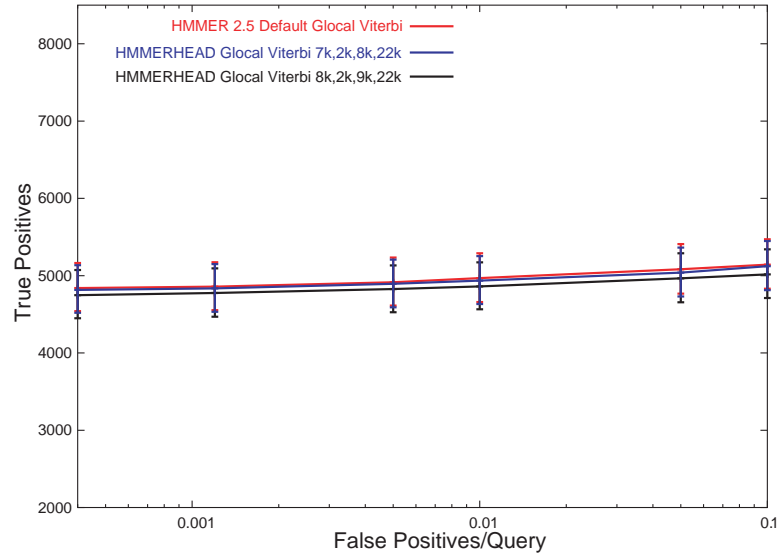


Figure 5.6: **HMMERHEAD Glocal Viterbi Performance**

HMMERHEAD glocal Viterbi 7k/2k/8k/22k identifies a comparable number of trues as the Viterbi algorithm. While HMMERHEAD glocal Viterbi 8k/2k/9k/22k obtains a faster search speed, it displays a greater loss in sensitivity. Error bars represent the minimum and maximum true positives identified from bootstrapping.

reduces the time needed to search Genbank’s NR database from hours to approximately 12 minutes on a single processor.

One concern with the HMMERHEAD algorithm is that it uses score dependent thresholds. The optimal HMMERHEAD thresholds were determined empirically using HMMER 2.5 default information content weighted models. These are HMMER’s currently best performing models according to my benchmark. However, if models were constructed using an alternative target mean information content, there is no guarantee that the default HMMERHEAD thresholds would be optimal. A potential solution to this would be to make HMMERHEAD’s thresholds P-value based and thus supposedly less vulnerable to model score changes. We explored this option early in HMMERHEAD’s development, but we could not identify thresholds that provided a minimum in sensitivity loss while still speeding up the search process.

5.4 Methods

5.4.1 HMMERHEAD and HMMER 2.5 Timings

Default `hmmsearch` and `HMMERHEAD` searches were timed on a randomly selected list of 200 HMMs. The linux “time” function was used to measure the runtime of these searches on the fast nodes of the Eddy lab cluster. The processors on these nodes are Intel Xeons 3.60 GHz.

5.4.2 Test Database Modifications

The development of this search heuristic means that HMMER is likely to be used on large databases. To better account for the possible compositional biases present in a larger sequence database, I decided to increase the number and compositional variation of my non-homologous sequences. I used SQUID’s shuffle program to create a shuffled version of each of the sequences in the non-redundant sequence database, NRDB95-06. (See Chapter 6 Methods.) These shuffled sequences were combined with my SCOP test sequences to create a new test database. This new test database consists of a total of 2,316,099 sequences.

5.4.3 Availability

HMMERHEAD incorporated into HMMER 2.5 can be found combined with the benchmarking scripts, alignments, and the shuffled version of NRDB95-06 at: <http://selab.wustl.edu/people/steve/thesis/bm-nrdb95-06.tar.gz>

Chapter 6

JackHMMER

6.1 Background

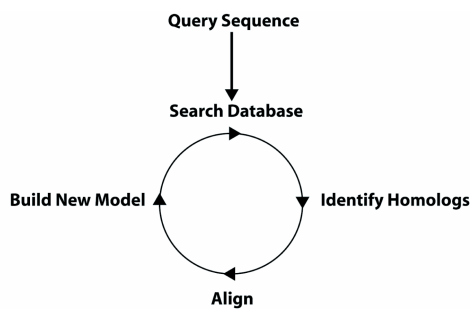


Figure 6.1: Overview of an Iterative Search Strategy

The basic logic behind an iterative search method is outlined in Figure 6.1. The idea is that one first searches a database and identifies homologous sequences. These newly identified sequences are then added to the model of the query. This revised model is then searched against the database again. This process continues until no new homologs are identified or some maximum number of iterations are reached. A variety of sequence comparison methods have been used in an iterative approach. I will briefly cover work using pairwise sequence comparison methods, Position Specific Scoring Matrices/Profiles, and the recent use of Hidden Markov Models in this fashion [5, 44, 88].

6.1.1 Position Specific Scoring Matrices and Profile

In iterative Position Specific Scoring Matrix (PSSM) an profile methods, the goal is to increase the sensitivity of the model being used by incorporating as many diverse homologs as possible. Thus, one takes the homologs from a search and folds them into the model, hopefully, making it more sensitive.

In 1987, Dodd and Egan published a study identifying novel potential λ Cro-like DNA binding regions in several proteins by using a Position Specific Scoring Matrix in an iterative fashion [15]. Like HMMs, PSSMs are built from alignments of the domain of interest. Protein PSSMs can be thought of as a matrix with 20 rows, one for each type of amino acid, and L columns, where L is the length of the alignment. Each column in the PSSM represents a column in the alignment and a row element corresponds roughly to the frequency of a particular amino acid in that column of the alignment. Their initial PSSM was built from 3 sequences, and they iteratively searched this PSSM over a database of DNA binding proteins. Using this approach, they identified 37 potential Cro-like DNA binding regions. There have been many strategies on how to calculate the element values in a PSSM. One weaknesses in Dodd and Egan's work came from not rigorously handling unobserved residues in an alignment column.

In the same year, Gribskov *et al.* introduced the concept of profiles, which are similar in many regards to PSSMs, but allow for the use of gaps. Gribskov's method used a substitution matrix in the determination of residue scores and the use of gapped Smith-Waterman to identify target sequence matches [28].

Profiles proved to be a powerful tool for sequence analysis and several additional improvements were made to their construction and scoring. Internal sequence weighting, to compensate for highly similar sequences in the alignment, were shown to improve profile performance [72, 89]. Also, pseudocount methods were employed to better account for unobserved residues [72].

In 1997, these improvements were combined in NCBI's Psi-BLAST [5]. Psi-BLAST was one of the first packages that handled profile construction, searching, and iteration in one user-friendly program. In addition, Psi-BLAST utilized the BLAST search heuristics to speedup profile scoring. Later versions of Psi-BLAST used the BLAST heuristic but then performed a full

Smith-Waterman alignment of those sequences that passed the heuristic [72]. In Psi-BLAST, the alignment sequences are weighted by a modified version of Henikoff and Henikoff [34]. The profile scores are of the form $\log(Q_i/P_i)$, where P_i is the background probability of residue i and Q_i is the estimated probability of seeing residue i in that column of the profile. Q_i is calculated from the observed residues and the pseudocount method of Tatusov *et al.* [88].

6.1.2 Pairwise Sequence Comparisons

In 1997, Park *et al.* proposed utilizing intermediate sequences to improve pairwise sequence comparison methods. In their approach, they compared a set of queries, SCOP, to a non-redundant protein database using the pairwise sequence comparison program FASTA [64]. They identified those cases where two different SCOP sequences matched the same region in a sequence from the non-redundant database with an E-value < 5 over a length of 30 residues or more. They found that 50% of the proteins linked by a common non-redundant database sequence were homologous according to SCOP, but had insignificant scores when compared against each other using FASTA. One potential limitation of this work is that the E-value and region length cutoff for calling proteins linked were empirically determined. It is unclear whether these are optimal cutoffs for other sequences. Additionally, these same authors performed a later benchmarking study that showed that this method is outperformed by both Psi-BLAST and an iterated HMM approach [63].

6.1.3 Iterative HMM Searches

Recently, HMMs have been used in an iterative fashion. As with iterative profile methods, the reasoning is that by incorporating more diverse homologs into an HMM, one can increase its sensitivity. SAM's t-02 program uses HMMs to iteratively search a database [45]. The main limitation of their implementation is due to the slowness of HMM search algorithms. Since searching large databases with HMMs has previously been extremely time consuming, it was not practical to perform an iterative HMM approach over the larger databases. Thus,

SAM's t-02, creates a sub-database of weakly similar proteins from the original database using a pairwise search method like Wu-BLAST.

The advent of HMMERHEAD allowed for the removal of this sub-database constraint and iteration over the full database. In the next section I describe an iterative HMM search program that I have developed called JackHMMER. I test whether the usage of a sub-database limits the effectiveness of my iterative HMM approach and test JackHMMER's performance relative to profile and HMM iterative search programs.

6.2 Results

6.2.1 JackHMMER Algorithm

In the JackHMMER algorithm, a query sequence is searched against a sequence database using the pairwise sequence programs Wu-BLASTP or NCBI's BLASTP. Initial homologs with E-values below a threshold from this search are identified. The sequence regions from those homologs that aligned to the HMM are then extracted from the database and aligned using ClustalW. This alignment is used to build a local HMM, which is calibrated for a Viterbi search. This model is then searched against the database using HMMERHEAD Viterbi. Sequence regions identified by this search are then extracted from the database and aligned to the model using HMMER's hmalign. A new local model is then built and calibrated. This model is again searched against the database. This continues until no unique homologs are identified or a maximum number of iterations is reached. The output consists of the final alignment of homologs identified and a HMMER local model built from this alignment.

6.2.2 General Iterative Method Concern

A concern regarding any iterative method is what is referred to as "model poisoning". Iterative methods are continuously adding novel sequences to their models. If a non-homologous sequence is allowed into the model, this may

increase the chance that other sequences related to this non-homologous sequence will then be accepted into the model. Thus, the model will no longer accurately represent the desired protein domain. To test whether this was occurring in JackHMMER and the other methods, shuffled sequences were added to the iteration database used for all methods. The addition of labeled shuffled sequences in the iteration database allows for the detection of model poisoning. Inspection of the alignment files shows that HMMER and SAM's models did not incorporate shuffled sequences after the iteration process. Psi-BLAST's checkpoint file is not in ASCII format, so it was not clear whether shuffled sequences had been incorporated into its profile. Additionally, the use of the same shuffled sequences in the iteration and the test database means that if a shuffled sequence did poison a model, there is a high probability that the same shuffled sequence in the test database will match the model and be revealed by a high scoring false positive in the benchmark.

6.2.3 Sub-Databases Limit the Performance of Iterative HMM Searches

SAM's t-02 utilizes a sub-database, created from the results of a pairwise sequence comparison method, for iteration. The main limitation of this type of approach is when remote homologs are not included in the sub-database due to the limited sensitivity of the pairwise search method. Regardless of the sensitivity of HMMs, homologs that are not included in the sub-database can never be identified and incorporated into the model. I tested whether the use of a sub-database limits the effectiveness of iterative HMM searches by benchmarking the performance of a modified version of JackHMMER that utilized a sub-database for iteration. Following the example of SAM's t-02, I built a sub-database from all the initial Wu-BLASTP sequences with E-values ≤ 600 . This sub-database was then iterated over. I compared the performance of this version to the JackHMMER algorithm. The version of JackHMMER that utilizes a sub-database identifies an average of 1000 less trues at every false positives per query level (Figure 6.2). It does appear that the use of a sub-database can limit the sensitivity of an iterative HMM approach.

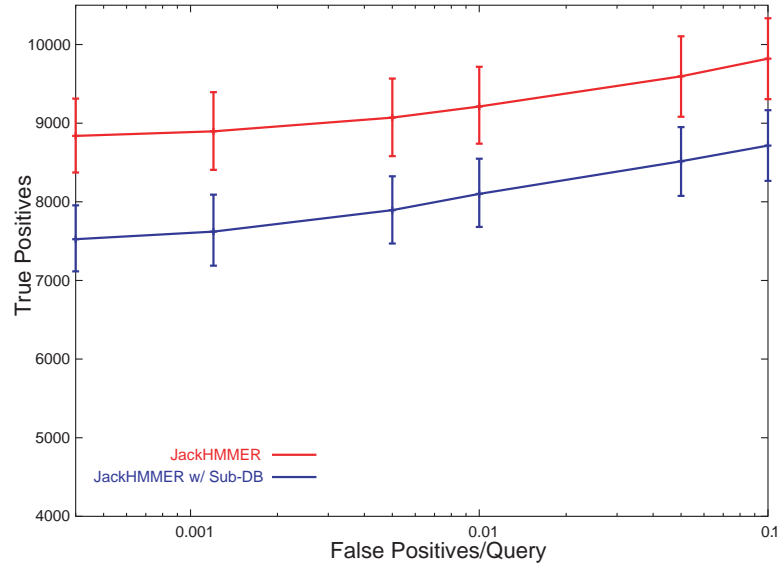


Figure 6.2: **Effect of Sub-Database Use on JackHMMER Searches**
 The utilization of a sub-database significantly impairs the performance of the JackHMMER iterative approach. Error bars represent the minimum and maximum true positives identified from bootstrapping.

6.2.4 JackHMMER Outperforms NCBI’s Psi-BLAST and SAM’s t-02

To test JackHMMER’s performance relative to other iterative search approaches, I benchmarked JackHMMER, Psi-BLAST, and SAM’s t-02. JackHMMER detects an average of 1,500 more trues than its closest competitor, Psi-BLAST (Figure 6.3). This is statistically significant at all false positive per query levels. I was surprised to find that Psi-BLAST outperforms SAM’s t-02 on this benchmark. Given the results of the previous experiment, I strongly suspect this is due to SAM’s reliance on a sub-database. It should be noted that JackHMMER’s performance has slightly increased relative to that shown in Figure 6.2. Prior to this experiment, I performed optimization experiments on the different methods and determined that increasing the maximum number of iterations from five to seven improved the performance of all the methods.

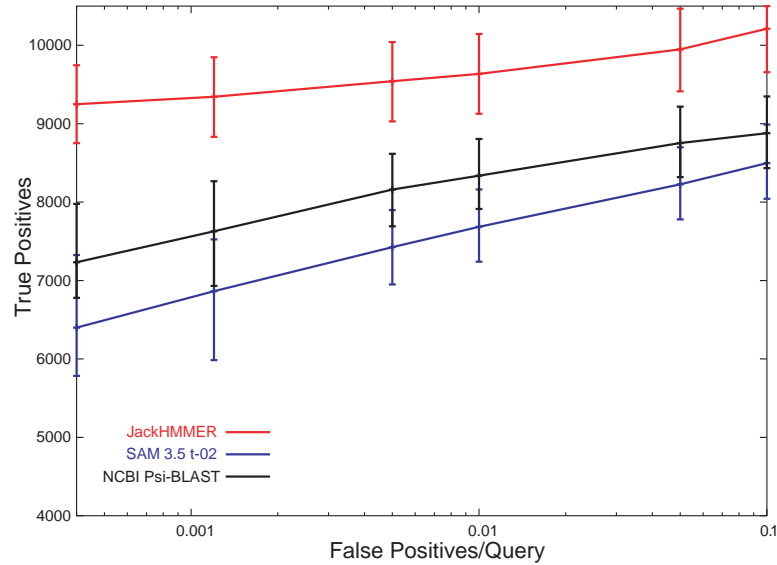


Figure 6.3: **JackHMMER Performance**

JackHMMER outperforms Psi-BLAST and SAM’s t-02. Error bars represent the minimum and maximum true positives identified from bootstrapping.

6.3 Concluding Remarks

In this research, I implemented an iterative approach utilizing HMMER HMMs. The current iterative HMM approach, SAM’s t-02, performs its iterative searches over a sub-database created from the use of a pairwise sequence comparison method. This is done primarily due to the slowness of the current HMM scoring algorithms. I implemented an iterative HMM approach, JackHMMER, using the search heuristic HMMERHEAD, that allows for iteration over a full database.

I suspected the use of a sub-database for iteration would limit the performance of an iterative HMM approach. I tested this by creating a modified version of JackHMMER that utilized a sub-database for iteration using the same criteria as SAM t-02 and compared its performance on my benchmark relative to the original JackHMMER. The use of a sub-database decreases JackHMMER’s sensitivity by an average of 1000 trues at every false positives per query level. Compared to Psi-BLAST and SAM’s t-02, JackHMMER detects an average of 1,500 more trues than its closest competitor, Psi-BLAST. This represents an

increase in sensitivity of 4% and is statistically significant at all false positive per query levels.

One concern regarding JackHMMER, as well as all iterative methods, is the possibility of model poisoning. This is where non-homologous sequences are incorporated into a model and then cause further degradation of the sequence signal being modeled. I have determined that no shuffled sequences have made it into the JackHMMER or SAM models benchmarked. However, shuffled sequences do not possess the same regional compositional bias and short-period repeats present in biological sequences. Thus, model poisoning by biological sequences may be more likely to occur.

A future area of work on JackHMMER would be the development of single sequence model searches that are comparable or better in performance than the currently employed BLAST step. It would be interesting to determine whether single sequence models, built with BLOSUM-based pseudocounts, appropriate insert/delete transitions, and scored using the HMMERHEAD Forward algorithm, would be more sensitive than BLAST. This would also allow JackHMMER to rely solely on HMMER HMMs for its database searches.

6.4 Methods

6.4.1 Iteration Database

An iteration database had to be generated to allow JackHMMER, SAM's t-02, and Psi-BLAST to generate their iterative models on a control database. The iteration database I used consisted of the non-redundant database, NRDB95-06, plus a shuffled version of NRDB95-06 generated by SQUID's shuffle program.

NRDB95-06 was created from Genbank's NR, 1/12/06, using Holm and Sander's method to remove sequences that are $\geq 95\%$ identical [38]. The script used is available at:

<http://www.embl-ebi.ac.uk/~holm/nrdb90>

6.4.2 Testset Modification

The SAM 3.5 t-02 program failed to build models for 2 of the 2,521 test sequences. Code specific errors were generated when these two specific sequences were used and since the SAM software package is closed source, I was unable to resolve the source of the problem. Thus, these two sequences were removed from all iterative model building analyses.

6.4.3 Benchmarking Procedure

All methods were run for a maximum of 6 iterations over the iteration database. The respective models were then searched over the test database and followed the standard benchmarking procedure.

6.4.4 Comparison Programs

JackHMMER

JackHMMER's models were default information content weighted local models. The iteration models were scored with Viterbi. The final benchmark searches were with the forward algorithm. While JackHMMER could use any version of HMMER, these experiments were done using HMMER 2.5.

SAM 3.5

Since the input alignments had changed from my previous SAM optimization experiments, I wanted to verify that SAM's w0.5 model building script was still optimal on my benchmark. I evaluated the performance of SAM's w0.5, w0.7, and w1.0 building scripts with local and glocal model architectures of SAM 3.5 on a subset of 250 iteration built models on the current test database. I confirmed that w0.5 built local models still performed optimally on my benchmark. Thus, SAM t-02 final models were built with w0.5 and forward local searches were performed against the test database.

Psi-BLAST

NCBI's Psi-BLAST version 2.2.13 was used for these experiments. Psi-BLAST checkpoint models were saved after the iteration process and these models were then used to search the test database using default settings. The NCBI BLAST software can be found at:

<http://www.ncbi.nlm.nih.gov/BLAST/download.shtml>

6.4.5 Availability

JackHMMER is part of HMMER 2.5 and can be found combined with the benchmarking scripts, alignments, and the shuffled version of NRDB95-06 at:

<http://selab.wustl.edu/people/steve/thesis/bm-nrdb95-06.tar.gz>

Appendix A

Major Scripts/Programs

A.1 Overview

In this section, I will list the major accessory scripts/programs I used in my research with a description of the options. An example of these scripts used in a benchmark given in appendix B.

A.2 Accessory Scripts/Programs

1) Search Output Parsing

mhmmercrunch.pl/hmmercrunch.pl <listfile of search output>

msamcrunch.pl/samcrunch.pl <listfile of search output>

mblastercrunch.pl/blastercrunch.pl <listfile of search output>

I used this family of scripts to parse the results from the different search output formats. There was a common theme between all these programs. As an example, mhmmercrunch.pl takes a list of hmmsearch files, created by 'ls -1 *.hmmsearch > hmmsearch.listfile', which runs hmmercrunch.pl on each. hmmercrunch.pl parses each of the individual files to a common output to be used by a scoring program described below. For hmmercrunch.pl, the output files

were *.hmmercrunch. These programs output a running count of how many input files, e.g. hmmsearch files, it has processed. This can be turned off by using the option “silent” command.

2) Scoring Scripts

```
score_hmmer.pl <listfile> <E-Val Thresh> <mg or random> <scop or cath>  
score_sam.pl <listfile> <E-Val Thresh> <mg or random> <scop or cath>  
score_blast.pl <listfile> <E-Val Thresh> <mg or random> <scop or cath>
```

These programs take in a list of parsed output files, e.g. created by 'ls -l *.hmmercrunch > crunch.listfile'. The output, <stdout>, for these scripts is a compiled list of all the query/target hits above the E-value threshold. This list is ranked in ascending E-value order and scored by either M/G or random, my benchmark, rules. Since at one point I doublechecked my SCOP results with those from the CATH database, there is also an option to let the program know if CATH sequences are being used.

is

3) Plotting Scoring Script Output

```
epq.pl <Score file>
```

This script takes a score file and finds the number of true positives at 1, 3, 12, 25, 125, and 250 false positives. It then outputs, <stdout>, these values preceded by the false positive per query values for my benchmark. i.e. 0.0004, 0.0012, 0.005, 0.01, 0.05, 0.1

4) Bootstrapping Scripts

```
bootstrap_hmmer.pl <replicate listfile> <parsed dir> <E-Val Thresh> <mg or  
random> <scop or cath>
```

```
bootstrap_sam.pl <replicate listfile> <parsed dir> <E-Val Thresh> <mg or  
random> <scop or cath>
```

```
bootstrap_blast.pl <replicate listfile> <parsed dir> <E-Val Thresh> <mg or  
random> <scop or cath>
```

These programs use a pre-arranged list of sampled parsed output files, e.g. hmmercrunch files. These lists come from sampling, with replacement, the list of 2,251 original output files. These lists were generated using bootstrap-replicant.pl [list of parsed files] [2521] [1000] and are titled rep_‘number’. e.g. rep_0 In addition to the list of these different sampled lists, they take in a directory where all the parsed output files can be found. The rest of the arguments fulfill the same purpose as the above scoring scripts. These scripts will then output to a file named ‘replicate’.scored‘E-Value’. e.g. rep_0.scored10

```
bootstrap-minmax.pl <scored replicates listfile> <False Positive Thresh>
```

This program will take in a list of scored replicate files, created by ‘ls rep_* > rep.listfile’, and go through each scored replicate and determine the number of trues identified before the false positive threshold. This script then outputs, <stdout>, the mean number of trues found plus the range.

```
bootstrap-stats.pl <scored rep listfile> <scored rep listfile> <FP>
```

This program is the one used to determine if the bootstrapping results from two different programs are statistically significant. This program takes in two lists of scored replicate files, created as above. The program identifies how many trues are found before the FP threshold for each line from both lists. It then determines the difference between the two true positive counts for each scored replicate. NOTE: REPLICATES MUST BE IDENTICAL AS WELL AS THE TWO LISTFILES NEED TO BE IN THE SAME ORDER. This program then outputs, <stdout>, the mean number of trues found by each program and the

middle 95% range of the distribution of differences across all the replicates. If this range overlaps zero, it fails the 0.05 significance test.

Appendix B

Benchmarking Pipelines

B.1 Overview of My Benchmark

This is an example list of commands I used to generate a single score file and bootstrapping results from a directory of search results using my scoring scheme, previously described in Chapter 2. The example I give is for hmmsearch output but the relevant parts can be changed, as described in Scripts/Programs, for the different search methods.

1) List output files

```
'ls -l *.hmmsearch > hmmsearch.listfile'
```

2) Generation of parsed hmmsearch files

```
'mhammercrunch.pl hmmsearch.listfile'
```

3) List Parsed Output Files

```
'ls -l *.hammercrunch > crunch.listfile'
```

4) Generation of Scorefile

```
'score_hmmer.pl crunch.listfile 10 random scop > SFscorefile'
```

5) Creation of True Positives versus False Positives Per Query Gnuplot file

```
'epq.pl SFscorefile > SFscorefile.epqtp'
```

6) NOTE: NOW WE ASSUME THERE IS A COMPLETE SET OF FILES PRODUCED BY bootstrap_hmmer.pl

7) List the scored replicates

```
'ls -l rep_* > rep.listfile'
```

8) Determine the mean and range of trues for each fp/query level.

```
bootstrap-minmax.pl rep.listfile 1 > minmax-1fps
```

```
bootstrap-minmax.pl rep.listfile 1 > minmax-3fps
```

```
bootstrap-minmax.pl rep.listfile 1 > minmax-12fps
```

```
bootstrap-minmax.pl rep.listfile 1 > minmax-25fps
```

```
bootstrap-minmax.pl rep.listfile 1 > minmax-125fps
```

```
bootstrap-minmax.pl rep.listfile 1 > minmax-250fps
```

9) Put the mean and range data(for plotting errorbars) into a gnuplot file.

```
grep "Mean" minmax-1fps minmax-3fps minmax-12fps minmax-25fps
```

```
minmax-125fps minmax-250fps | awk 'print $3;' > minmax.epqtp
```

```
grep "Mean" minmax-1fps minmax-3fps minmax-12fps minmax-25fps
```

```
minmax-125fps minmax-250fps | awk 'print $3' '$5' '$7;' > minmax.error.epqtp
```

10) Test to see if the two programs pass a 0.05 confidence interval for significance at the different false positives per query levels.

```
bootstrap-stats.pl rep.listfile1 rep.listfile2 1 > stats-1fps
bootstrap-stats.pl rep.listfile1 rep.listfile2 13 > stats-3fps
bootstrap-stats.pl rep.listfile1 rep.listfile2 12 > stats-12fps
bootstrap-stats.pl rep.listfile1 rep.listfile2 25 > stats-25fps
bootstrap-stats.pl rep.listfile1 rep.listfile2 125 > stats-125fps
bootstrap-stats.pl rep.listfile1 rep.listfile2 250 > stats-250fps
```

B.2 Madera and Gough Benchmark Overview

This process would be the same as above except at steps 2) and 4) one would use 'mg' instead of 'random' to score the output in a Madera and Gough fashion.

References

- [1] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219:555–565, 1991.
- [2] S. F. Altschul, R. J. Carroll, and D. J. Lipman. Weights for data related by a tree. *J. Mol. Biol.*, 207:647–653, 1989.
- [3] S. F. Altschul and W. Gish. Local alignment statistics. *Meth. Enzymol.*, 266:460–480, 1996.
- [4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.
- [5] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.*, 25:3389–3402, 1997.
- [6] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S. R. Eddy, S. Griffiths-Jones, K. L. Howe, M. Marshall, and E. L. Sonnhammer. The Pfam protein families database. *Nucl. Acids Res.*, 30:276–280, 2002.
- [7] A. Bateman, E. Birney, R. Durbin, S. R. Eddy, R. D. Finn, and E. L. Sonnhammer. Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucl. Acids Res.*, 27:260–262, 1999.
- [8] S. E. Brenner, P. Koehl, and M. Levitt. The ASTRAL compendium for protein structure and sequence analysis. *Nucl Acids Res*, 28:254–256, 2000.
- [9] J. M. Chandonia, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt, and S. E. Brenner. ASTRAL compendium enhancements. *Nucl Acids Res*, 30:260–263, 2002.

- [10] K. P. Choi, F. Zeng, and L. Zhang. Good spaced seeds for homology search. *Bioinformatics*, 20:1053–1059, 2004.
- [11] L. Lo Conte, B. Ailey, T. J. Hubbard, S. E. Brenner, A. G. Murzin, and C. Chothia. SCOP: a structural classification of proteins database. *Nucl Acids Res*, 28:257–259, 2000.
- [12] M. O. Dayhoff. *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Silver Spring, Md, 1966.
- [13] A. Dembo and S. Karlin. Strong limit theorems of empirical functionals for large exceedences of partial sums of iid variables. *Annals of Probability*, 19:1737–1755, 1991.
- [14] A. Dembo, S. Karlin, and O. Zeitouni. Limit distribution of maximal non-aligned two-sequence segmental score. *Annals of Probability*, 22:2022, 1994.
- [15] I. B. Dodd and J. B. Egan. Systematic method for the detection of potential lambda cro-like DNA-binding regions in proteins. *J Mol Biol*, 194:557–564, 1987.
- [16] R. F. Doolittle. Some reflections on the early days of sequence searching. *J Mol Med*, 75:239–241, 1997.
- [17] R. F. Doolittle. On the trail of protein sequences. *Bioinformatics*, 16:24–33, 2000.
- [18] R. F. Doolittle, M. W. Hunkapiller, L. E. Hood, S. G. Devare, K. C. Robbins, S. A. Aaronson, and H. N. Antoniades. Simian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor. *Science*, 221:275–277, 1983.
- [19] R. C. Edgar and K. Sjolander. COACH: profile-profile alignment of protein families using hidden Markov models. *Bioinformatics*, 20:1309–1318, 2004.
- [20] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, UK, 1993.

- [21] A. L. Vettore et al. Analysis and functional annotation of an expressed sequence tag collection for tropical crop sugarcane. *Genome Res*, 13:2725–2735, 2003.
- [22] C. R. Buell et al. Sequence, annotation, and analysis of synteny between rice chromosome 3 and diverged grass species. *Genome Res*, 15:1284–1291, 2005.
- [23] R. D. Finn, J. Mistry, B. Schuster-Bockler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. Sonnhammer, and A. Bateman. Pfam: Clans, web tools and services. *Nucl Acids Res*, 34:D247–D251, 2006.
- [24] W. M. Fitch. An improved method of testing for evolutionary homology. *J Mol Biol*, 16:9–16, 1966.
- [25] M. Gerstein, E. L. L. Sonnhammer, and C. Chothia. Volume changes in protein evolution. *J. Mol. Biol.*, 235:1067–1078, 1994.
- [26] W. Gish. <http://blast.wustl.edu>. 1996-2004.
- [27] R. E. Green and S. E. Brenner. ASTRAL compendium enhancements. *Nucl Acids Res*, 30:260–263, 2002.
- [28] M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. *Proc Natl Acad Sci USA*, 84:4355–4358, 1987.
- [29] N. V. Grishin. KH domain: one motif, two folds. *Nucl Acids Res*, 29:638–643, 2001.
- [30] W. N. Grundy. Family-based homology detection via pairwise sequence comparison. *RECOMB 98*, pages 94–99, 1998.
- [31] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Biology*. Cambridge University Press, Cambridge UK, 1997.
- [32] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucl. Acids Res.*, 19:6565–6572, 1991.
- [33] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

- [34] S. Henikoff and J. G. Henikoff. Position-based sequence weights. *J. Mol. Biol.*, 243:574–578, 1994.
- [35] E.E. Hill, V. Morea, and C. Chothia. Sequence conservation in families whose members have little or no sequence similarity: The four-helical cytokines and cytochromes. *J Mol Biol*, 322:205–233, 2002.
- [36] T. C. Hodgman. A historical perspective on gene/protein functional assignment. *Bioinformatics*, 16:10–15, 2000.
- [37] L. Holm and C. Sander. The FSSP database of structurally aligned protein fold families. *Nucl. Acids Res.*, 22:3600–3609, 1994.
- [38] L. Holm and C. Sander. Removing near-neighbour redundancy from large protein sequence collections. *Bioinformatics*, 14:423–429, 1998.
- [39] D.H. Horn, M. Houston, and P. Hanrahan. ClawHMMER: A streaming hmmer-search implementation. *Supercomputing 2005*, 2005.
- [40] F. Jeanmougin, J. D. Thompson, M. Gouy, D. G. Higgins, and T. J. Gibson. Multiple sequence alignment with Clustal X. *Trends Biochem Sci*, 23:403–405, 1998.
- [41] V. Jovic, N. Jovic, D. Geiger, A. Siepel, D. Haussler, and D. Heckerman. Efficient approximations for learning phylogenetic HMM models from data. *Bioinformatics*, 20:161–168, 2004.
- [42] D. T. Jones, W. R. Taylor, and J. M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci*, 8:275–282, 1992.
- [43] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci USA*, 87:2264–2268, 1990.
- [44] K. Karplus, C. Barrett, M. Cline, M. Diekhans, L. Grate, and R. Hughey. Predicting protein structure using only sequence information. *Proteins*, 3:121–5, 1999.

- [45] K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1998.
- [46] K. Karplus, R. Karchin, G. Shackelford, and R. Hughey. Calibrating E-values for hidden Markov models using reverse-sequence null models. *Bioinformatics*, 21:4107–4115, 2005.
- [47] H. Kawaaji, C. Schonbach, Y. Matsuo, J. Kawai, Y. Okazaki, Y. Hayashizaki, and H. Matsuda. Exploration of novel motifs derived from mouse cDNA sequences. *Genome Res*, 12:367–378, 2002.
- [48] W. J. Kent. BLAT—the BLAST-like alignment tool. *Genome Res*, 12:656–664, 2002.
- [49] S. S. Krishna and N. V. Grishin. Structural drift: a possible path to protein fold change. *Bioinformatics*, 21:1308–1310, 2005.
- [50] S.S. Krishna, R.I. Sadreyev, and N.V. Grishin. A tale of two ferredoxins: sequence similarity and structural differences. *BMC Struct Biol*, 6, 2006.
- [51] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *J Mol Biol*, 235:1501–1531, 1994.
- [52] S. Li, J. Liao, G. Cutler, T. Hoey, J. B. Hogenesch, M. P. Cooke, P. G. Schultz, and X. B. Ling. Comparative analysis of human genome assemblies reveals genome-level differences. *Genomics*, 80:138–139, 2002.
- [53] D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441, 1985.
- [54] A.N. Lupas, C.P. Ponting, and R.B. Russell. On the evolution of protein folds: Are similar motifs in different protein folds the result of convergence, insertion, or relics of an ancient peptide world. *J Struct Biol*, 134:191–203, 2001.
- [55] B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18:440–445, 2002.

- [56] M. Madera and J. Gough. A comparison of profile hidden Markov model procedures for remote homology detection. *Nucl Acids Res*, 30:4321–4328, 2002.
- [57] G. J. Mitchison. A probabilistic treatment of phylogeny and sequence alignment. *J Mol Evol*, 49:11–22, 1999.
- [58] D. Morton. Andrew viterbi, electrical engineer, an oral history. *IEEE History Center*, 1999.
- [59] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48:443–453, 1970.
- [60] T. Oliver, B. Schmidt, and D. Maskell. Hyper-customized processors for bio-sequence database scanning on FPGAs. *International Symposium of Field Programable Gate Arrays*, pages 229–237, 2005.
- [61] C. A. Orengo and J. M. Thornton. Protein families and their evolution-A structural perspective. *Annu Rev Biochem*, 7:867–900, 2005.
- [62] R. Page and E.C. Holmes. *Molecular Evolution: A Phylogenetic Approach*. Blackwell Science Ltd, Oxford UK, 1998.
- [63] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J Mol Biol*, 284:1201–1210, 1998.
- [64] J. Park, S. A. Teichmann, T. Hubbard, and C. Chothia. Intermediate sequences increase the detection of homology between sequences. *J Mol Biol*, 273:349–354, 1997.
- [65] W. R. Pearson and M. L. Sierk. The limits of protein sequence comparison? *Curr Opin Struct Biol*, 15:254–260, 2005.
- [66] S. Pietrokovski, J. G. Henikoff, and S. Henikoff. The Blocks database – a system for protein classification. *Nucl Acids Res*, 24:197–200, 1996.

- [67] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77:257–286, 1989.
- [68] E. Rivas. Evolutionary models for insertions and deletions in a probabilistic modeling framework. *BMC Bioinformatics.*, 6:63, 2005.
- [69] T. Rognes and E. Seeberg. Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16:699–706, 2000.
- [70] R. Sadreyev and N. Grishin. COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. *J Mol Biol*, 326:317–336, 2003.
- [71] R. I. Sadreyev, D. Baker, and N. V. Grishin. Profile-profile comparisons by COMPASS predict intricate homologies between protein families. *Protein Sci*, 12:2262–2272, 2003.
- [72] A. A. Schaffer, L. Aravind, T. L. Madden, S. Shavirin, J. L. Spouge, Y. I. Wolf, E. V. Koonin, and S. F. Altschul. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucl Acids Res*, 29:2994–3005, 2001.
- [73] T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht. Information content of binding sites on nucleotide sequences. *J Mol Biol*, 188:415–431, 1986.
- [74] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656, 1948.
- [75] P. R. Sibbald and P. Argos. Weighting aligned protein or nucleic acid sequences to correct for unequal representation. *J. Mol. Biol.*, 216:813–818, 1990.
- [76] A. Siepel and D. Haussler. Combining phylogenetic and hidden Markov models in biosequence analysis. *J Comput Biol*, 11:413–428, 2004.
- [77] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: A method for improving detection of weak

- but significant protein sequence homology. *Comput. Applic. Biosci.*, 12:327–345, 1996.
- [78] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147:195–197, 1981.
- [79] T. F. Smith, M. S. Waterman, and C. Burks. The statistical distribution of nucleic acid similarities. *Nucl Acids Res*, 13:645–656, 1985.
- [80] J. Soding. Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21:951–960, 2005.
- [81] E. L. L. Sonnhammer, S. R. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: Multiple sequence alignments and HMM-profiles of protein domains. *Nucl. Acids Res.*, 26:320–322, 1998.
- [82] E. L. L. Sonnhammer, S. R. Eddy, and R. Durbin. Pfam: A comprehensive database of protein families based on seed alignments. *Proteins*, 28:405–420, 1997.
- [83] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucl Acids Res*, 12:505–519, 1984.
- [84] G. D. Stormo. Consensus patterns in DNA. *Methods Enzymol*, 18:211–221, 1990.
- [85] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht. Use of the ‘perceptron’ algorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids Res*, 10:2997–3011, 1982.
- [86] Y. Sun and J. Buhler. Designing multiple simultaneous seeds for DNA similarity search. *J Comput Biol*, 12:847–861, 2005.
- [87] Y. Sun and J. Buhler. Choosing the best heuristic for seed alignment of DNA sequences. *Bioinformatics*, 7:133, 2006.
- [88] R. L. Tatusov, S. F. Altschul, and E. V. Koonin. Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *Proc Natl Acad Sci USA*, 91:12091–12095, 1994.

- [89] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties, and weight matrix choice. *Nucl Acids Res*, 22:4673–4680, 1994.
- [90] M. Vingron and P. Argos. A fast and sensitive multiple sequence alignment algorithm. *Comput Appl Biosci*, 5:115–121, 1989.
- [91] H. C. Watson and J. C. Kendrew. The amino-acid sequence of sperm whale myoglobin. Comparison between the amino-acid sequences of sperm whale myoglobin and of human hemoglobin. *Nature*, 19:670–672, 1961.
- [92] W. J. Wilbur and D. J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proc Natl Acad Sci U S A*, 80:726–730, 1983.
- [93] B. Wun, J. Buhler, and P. Crowley. Exploiting coarse-grained parallelism to accelerate protein motif finding with a network processor. *Proceedings of the Fourteenth International Conference on Parallel Architectures and Compilation Techniques*, 2005.
- [94] Y. K. Yu, R. Bundschuh, and T. Hwa. Hybrid alignment: High-performance with universal statistics. *Bioinformatics*, 18:864–872, 2002.
- [95] Y. K. Yu and T. Hwa. Statistical significance of probabilistic sequence alignment and related local hidden Markov models. *J. Comput. Biol.*, 8:249–282, 2001.

Vita

Steven Johnson

- Date of Birth** December 2, 1969
- Place of Birth** Waukegan, Illinois
- Degrees** B.S. Biology, May 1992
Ph.D. Molecular Genetics, Expected September 2006
- Professional Societies** International Society for Computational Biology
- Publications** N.G. Starostina, S. Marshburn, S. Johnson, S.R. Eddy,
R.M. Terns, and M.P. Terns. Circular box C/D RNAs
in *Pyrococcus furiosus*. *Proc Natl Acad Sci USA*,
101:14097–15101, 2004.
- R.H. Waterston, *et al.* Initial sequencing and comparative
analysis of the mouse genome. *Nature*, 420:520–562,
2002.
- E.S. Lander, *et al.* Initial sequencing and analysis of the
human genome. *Nature*, 409:860–921, 2001.
- T. Cornish, J. Chi, S. Johnson, Y. Lu, and J.T.
Campanelli. Globular domains of agrin are functional
units that collaborate to induce acetylcholine receptor
clustering. *J Cell Sci*, 112:1213–1223, 1999.

September 2006