12-20-2009

Meta State Generalized Hidden Markov Model for Eukaryotic Gene Structure Identification

Carl Baribault
University of New Orleans

Meta State Generalized Hidden Markov Model for Eukaryotic Gene Structure Identification

A Dissertation

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Engineering and Applied Science
Computer Science/Bioinformatics

by
Carl Edward Baribault

B.S. Loyola University, New Orleans, LA, 1978
M.S. University of North Carolina at Chapel Hill, 1981
M.A. University of Texas at Austin, 1986

December, 2009

Dedication

I hereby dedicate this effort to the loving memory of my late father, Jules Baribault, who said “education is the key”, and his late brother and my godfather, Edward Baribault, for all their untold forbearances in providing for their greater family and for their admirable service to their country. I also dedicate to the fond memory of the late Dr. John F. Christman, an extraordinary educator, lecturer, and mentor and engaging conversationalist, for his acquainting me with the Socratic method in my undergraduate years at Loyola University, New Orleans, and his guidance there and thereafter in formulating my varied career paths. Ad maiorem Dei gloriam.

Acknowledgment

Funding for this effort was derived from grants from NIH, NSF, LA Board of Regents, and NASA.

To the members of my committee, thanks in general for their perseverance and critical input. In particular, thanks to Dr. Huimin Chen for his guidance in formulating the factorizations of the HOHMM in this effort. Thanks to my advisor, Dr. Steven Winters-Hilt for providing the initial versions of the theory and code for the HOHMM and for his support and guidance throughout this effort. In my humble opinion, Dr. Winters-Hilt with his outstanding background and cross-disciplinary knowledge is a credit to the institution of the University of New Orleans as well as the Louisiana State University system.

To my research colleagues, A. Murat Eren, Brian Roux, Zuliang Jiang, and Alex Lu, thanks for their collaboration and provision of supporting material for this thesis. Thanks also, to all other members of Dr. Winters-Hilt's bioinformatics research team for all their cooperation.

To members of the UNO Department of Computer Science thanks for their supporting roles. In particular, thanks to the chairman, Dr. Mahdi Abdelguerfi, for his generous support. Thanks to the secretary, Ms. Jeanne Boudreaux, for all of her efforts and cooperation in processing payroll related forms, etc. Thanks also to the system administrator and colleague, Mr. Fareed Qaddoura, for his advice and help in keeping the various department servers in working order.

Finally, thanks to the members of my extended family for their various supporting roles – as well as their prayerful requests for my progress. Thanks to my wife, Sandra, for all her efforts, prayers, and support. Thanks to my parents, Jules and Marguerite, for nurturing a culture of learning in our young family and throughout our development and to all my siblings in their respective setting of examples of achievement, cooperation, familial care and loving concern.

Table of Contents

List of Figures

Abstract

Using a generalized-clique hidden Markov model (HMM) as the starting point for a eukaryotic gene finder, the objective here is to strengthen the signal information at the transitions between coding and non-coding (c/nc) regions. This is done by enlarging the primitive hidden states associated with individual base labeling (as exon, intron, or junk) to substrings of primitive hidden states or *footprint* states. Moreover, the allowed footprint transitions are restricted to those that include either one c/nc transition or none at all. (This effectively imposes a minimum length on exons and the other regions.) These footprint states allow the c/nc transitions to be seen sooner and have their contributions to the gene-structure identification weighted more heavily – yet contributing as such with a natural weighting determined by the HMM model itself according to the training data – rather than via introducing an artificial gain-parameter tuning on major transitions. The selection of the generalized HMM model is interpolated to highest Markov order on emission probabilities, and to highest Markov order (subsequence length) on the footprint states. The former is accomplished via simple count cutoff rules, the latter via an identification of anomalous base statistics near the major transitions using Shannon entropy. Preliminary indications, from applications to the *C. elegans* genome, are that the sensitivity/specificity (SN/SP) result for both the individual state and full exon predictions are greatly enhanced using the generalized-clique HMM when compared to the standard HMM. Here the standard HMM is represented by the choice of the smallest size of footprint state in the generalized-clique HMM. Even with these improvements, we observe that both extremely long and short exon and intron segments would go undetected without an explicit model of the duration of state.

The key contributions of this effort are the full derivation and experimental confirmation of a rudimentary, yet powerful and competitive gene finding method based on a higher order hidden Markov model. With suitable extensions, this method is expected to provide superior gene finding capability – not only in the context of pre-conditioned data sets as in the evaluations cited but also in the wider context of less preconditioned and/or raw genomic data.

Keywords: hidden Markov model, HMM, GHMM, gene finding, gene prediction

1 Introduction

1.1 Background

This work is largely a study in the extension of the traditional, 1st order hidden Markov model (HMM) to be used in the analysis of genomic signals, but some mention is also made of significant supporting roles played by Support Vector Machines (SVM's). The complementary capabilities of HMM feature extraction and SVM classification can be combined to produce highly effective approaches to signal processing in both genomic and cheminformatic settings as well.

The use of 1st order HMM in genomic signal processing is expected to have limited results in light of observed fluctuations of not only 1st order relative entropy but also that for higher orders over extended regions surrounding coding/non-coding (c/nc) boundaries in DNA sequence data. In general, attempts to employ higher order HMM's (HOHMM) can result in an exponential explosion of states and hence run-time complexity, where each HOHMM state is a fixed-length, but otherwise less restricted (though not entirely arbitrary) concatenation of primitive exon (`e`), intron (`i`), and junk (`j`) states.

This thesis presents an HOHMM using a *minimum-length assumption*, where coding and non-coding segments are assumed to be bounded in length from below, with the reasonable penalty of an additional, post-processing step in order to handle the shorter segments excluded from the ensuing prediction as a result. The advantage of this minimum-length assumption is a scalable HOHMM with *linear* complexity in the length of the extended states or *footprint* states.

1.2 Related Work

Computational gene finding dates back to the mid 1980's if not further [**1**]. New gene finding programs arose up until about the year 2000. Afterward came a period of publication of summaries of gene finding programs until 2003. Since that time up until the present, the available literature is generally in the area of sensors for detecting specific sites or features, such c/nc transitions, gene starts, etc.

One approach to gene finding used in *homology*-based programs generally attempts to identify new genes by characterizing the sequence alignments among known genes, such as in [**2**]. Another more recent effort discussed in [**3**], combines homology information with other *ab initio* information or intrinsic, statistical properties of the DNA sequence alone. In general, the main drawback of homology-based approaches is that they appear incapable of finding new genes. See, for example, in [**4**], where a study is cited in [**5**], in which only 50% of the proteins encoded in human chromosome 22 were found to be similar to previously known proteins.

As mentioned above, much of the effort in recent years has been expended in the development of specific sensors, such as those listed in Table 1 below. In general, these programs require (and in some cases provide) an external interface for incorporation into a comprehensive, structural gene identification system, for example, see **5.2 Future Work**, page 80.

Table 1 Programs for genetic feature sensors among the related work

Program Name	**Targeted Feature**	**Description**	**Reference**
MetaTISA	Transcription initiation sites (TIS)	Classification via phylogenetic clustering of genomic sequence data	[**6**]
WordSpy	Transcription factor binding motifs (TFBM)	Dictionary and grammar for sequence motifs + EST similarity	[7]
MZEF	Internal exons in human genes	Classification of genomic sequence features via quadratic discriminant analysis (QDA)	[**8**]
SpliceMachine	splice sites	SVM classification of splice site and exon content features	[**9**]
FirstEF	promoters and first exons	Classification of features in promoter and first donor splice sites via QDA	[**10**]
DNAFSMiner	TIS in vertebrate DNA/mRNA/cDNA, polyadenylation in human DNA	SVM classification of entropic/Markov features of target sites	[**11**]
	Splice sites	SVM classification with weighted degree kernel	[**12**], [**13**]
PEAKS	Regulatory motifs in DNA	Frequency of motifs near target feature sites	[**14**]
	Splice sites, translation start sites (TSS's), TIS	Neural network classification of higher order Markov dependencies at target sites	[**15**]
	Gene 3´ ends (stops)	Classification via positional clustering of polyadenylation sites of known transcripts	[**16**]

1.3 Organization of This Thesis

In **2 Structural Gene Prediction via HOHMM**, as the main body of work in this effort, we develop a higher order HMM (HOHMM) in order to predict the genetic structure of lower order eukaryotic genomes. We observe that the scalability of the computation is realized by allowing only those hidden states in a relatively small but reasonably representative subset of the Cartesian product, Λ^{F+1}, where Λ is the alphabet of primitive hidden states, and where the subset grows *linearly* in F, the footprint state size in consecutive (but overlapping by one primitive) dimers. In other words, in choosing a model where we fix the allowed minimum length of exons, introns, and junk strands, this in turn has the effect of limiting not only the size of the hidden state space but also the number of allowed state transitions to linear growth in state-word size F as measured in consecutive, overlapping dimers. As a reasonable cost for attaining such controlled growth and scalability, we would then simply defer the detection of both the very short and very long exons and introns (if not junk as well) until an additional post-processing step.

In **3 HOHMM Implementation and Results**, page 44, first there is an overview of the implementation including the organization at the top level of the client-server architecture common to both the training and testing facilities. Following that are the results of testing with 3 data sets, entire_1_contig of *C. elegans*, the benchmark data set ALLSEQ assembled by Burset & Guigo in [**17**], and data-reduced versions of Chromosomes I-V of *C. elegans* [**18**]. In the first two cases of the entire_1_contig and ALLSEQ data sets, we use the same data set for both training and testing in order to exhibit a simple, brute force search for optimum values of the HOHMM parameters. The 3rd data set has been data-reduced in order to remove alternative splicings. Using that data set we show the results of testing with 5-fold cross-validation. We also include graphs of relative entropy and a discussion of its usefulness for the various types of c/nc transitions identified in the annotation data for this last data set, where the entropy values have been computed in a window centered on the respective transition sites.

In **4 Other HMM Applications**, page 68, first we mention several other applications based on iterative HOHMM processing. Then we include a treatment taken largely from [**19**] on a form of duration modeling to the HMM (HMMwD) where the form is exact in the 2-state case in that there is no ambiguity of the destination state upon exit of a given state duration. This work has served to motivate the development of an exact, implicit HMMwD [**20**]. Either the approximate or the exact formulation can be used to extend the HOHMM to include duration submodels for introns and exons in order to improve splice site recognition. Here the future intent would be enhanced recognition of both unusually short as well as unusually long instances of both exons and introns. We also include the results of successful, preliminary trials of the HMMwD on synthetic data [**19**].

Also in **4.3 Controlled Acquisition via SVM**, page 77, and **4.4 Pattern Recognition Informed Feedback via LabWindows Automation,** page 78, we briefly describe ongoing efforts in measurements involving real time molecular capture and acquisition and control of pico-amp (pA) ion current signals induced by that capture at the site of a highly narrow protein channel called a *nanopore*. Also, with material taken largely from [**21**] we present the first functional prototype of a nanopore signal acquisition system using SVM-based identification and control of the molecular species captured at the nanopore channel site. Continuation of this work discussed in [**22**].

In **5 Discussion and Future Work**, page 79, we discuss the results of the HOHMM referring also to the results in the two evaluations cited. We also discuss some issues associated with the testing methodology in the evaluations cited and the impact on the programs evaluated. Finally, we provide perspective on the HOHMM and its greater usefulness in a wider context of gene finding.

6 is **Bibliography**, page 83, and in **7 Appendix**, page 88, there are detailed results of testing in the form of extended sequences of (combined contour and 3D surface) graphs of accuracy of the HOHMM for all values tested of the 4 main parameters in the model, M, F, L, and R. These extended sequences of graphs provide a compelling demonstration of the stability of the HOHMM with respect to the choice of those 4 parameter values for all such values tested. There is also in **7** an extended, fully connected, 7-page flowchart depicting the function and collaboration of key modules in both the training and testing phases of the HOHMM. The flowchart is then followed by a list of those same key modules, including descriptions of the collaborating interfaces. Both training and testing implementations share the same, high-level, client/server infrastructure. All training code specifically as well as all code in general for this client-server infrastructure is implemented in Perl. The C-based portion of the testing code, the Viterbi engine, is contained entirely within the server side.

1.4 Contributions

It should be noted that the original HOHMM derivation and code implementation (in Perl and C) were done in their initial form by my Ph.D. thesis advisor, Dr. Winters-Hilt [**23**]. When first presented to me, however, he indicated that there appeared to be an insidious run-time bug in the code implementation and that, although the implementation was 95% complete and operated with some predictive power (~70% accuracy), it was hopelessly flawed. My task, thus, began with learning how to do run-time debugging of a complex higher order HMM. I was warned that HMM implementation errors that only show at run-time are very complicated to isolate, and this was indeed the case, as born out by my thorough review and rederivation of the theory outlined in [**24**]. As a direct result of my review, the bug was identified and found to be due to a minor, but critically significant approximation error in the theoretical description itself and that this error had previously been faithfully represented in the code. Upon repairing the code with the correct underlying theory, the predictive power on the test data improved to 99% accuracy on the base-by-base prediction of coding state (exon) versus non-coding state (intron or junk) using the diagnostic data set (or *training contig*), entire_1_contig of *C. elegans*. (See **3.4.2 Summary of Results for Entire_1_contig of *C. elegans***, page 50.)

Moreover, the correct factorization of total probability for the HOHMM is presented in two different forms, each with accompanying sample calculation. As suggested above, the first form of factorization had already been outlined by my advisor, Dr. Winters-Hilt in [**24**]. The final, corrected form is presented in **2.5 Factorization 1 of Total Probability**, page 27, and clearly indicates the proper formulation for future extension via SVM-based discriminator as demonstrated to some extent in [**25**]. The 2nd form of factorization is also a contribution in this effort, and also admits formal substitution of an SVM-based discriminator. Though no testing was performed with this alternate form in this effort, this form was introduced to also link most closely with standard autoregressive HMM descriptions. (See, for example, [**26**].)

Another contribution in this effort is included in **2.10 Alternate Proof of Linear Upper Bound on Footprint States and Transitions**, page 39. This provides a more compact and concise argument for demonstrating the linear limit of growth of the HOHMM's complexity with respect to footprint state size – without the need for the full enumeration and precise counting of footprint states as shown in **2.4 Linear Growth in State/Transition Space**, page 14.

Last but not least in terms of man-hours among the contributions in this effort is the additional software architectural features (design) and coding effort (implementation) for the training and testing phases of the HOHMM, including among others:

1) a Perl-syntax based, hence fully programmable, user configuration input file for defining:
    a. units of work or *tasks* for both training and testing of the HOHMM, and
    b. full specification of the distribution and multi-threaded execution (via Perl threads) of those tasks among both task-specific and/or generic sets of processors available via public key authentication
2) a Perl based, reusable, client/server, multi-task, distributed processing architecture – reused for performing both training and testing tasks
3) a well-defined, hence extendable interface for task execution in multiple job scheduling contexts, including the currently implemented interfaces for:
    a. native OS job scheduling as implemented for the Ubuntu/Linux OS [**27**]

b. LoadLeveler job scheduling as implemented for the IBM/AIX OS on the IBM Power5 cluster [**28**].

(See **3.1.2 HOHMM Application Features**, page 44.)

2 Structural Gene Prediction via HOHMM

2.1 (Eukaryotic) Genomic Data

Once it is fully annotated, genomic data can be unambiguously represented by strings formed from the 4 letters a, c, g, and t denoting the DNA nucleotide bases adenine, cytosine, guanine, and thymine, respectively. Genes are units of genomic data which encode specific sequences of *amino acids* to form *proteins*. The process of annotating consists of designating specific segments of the data as coding and non-coding (c/nc) segments. (The uppercase letters A, C, G, and T are often used to designate coding segments as distinct from the lower-cased, non-coding segments in the data.) In eukaryotes, genes consist of coding segments or *exons* which are delimited internally by special, *intragenic*, non-coding segments or *introns*. The *intergenic*, non-coding regions of bases outside the genes are generically referred to as *junk*.

The process of removing the intermediate introns and reconnecting (possibly variable subsets of) the resulting exons end-to-end is referred to as *splicing*. Perhaps the most important role of introns is to provide a mechanism for the formation of alternative combinations and/or subsets of the exons contained in a given gene in order to form alternative proteins also used by the organism in question. These alternative combinations are referred to as *alternative splicings*.

2.1.1 Framing

Exons have a 3-base encoding as directly revealed in a mutual information analysis of gapped base interpolations as shown in [**23**]. The 3-base encoding elements are called *codons*, and the partitioning of the exons into 3-base subsequences is known as the codon *framing*. Thus in order to both precisely and concisely encode a protein, the gene's cumulative exon string length must be a multiple of 3 bases. The term *reading frame* is used to denote one of the 3 possible positions – 0, 1, or 2 by our convention – relative to the start of a codon. Moreover, in general introns may interrupt genes at any point along the coding extent. In other words, introns can split the codon framing either at a codon boundary or one of two positions internal to a codon.

Although there is no notion of framing per se among introns, for convenience we associate framing with the intron, as indicated in the example below, as a tracking device in order to ensure that the frame of the immediately following exon would be predicted correctly. Finally there is neither any notion nor any use for framing among strings of junk per se. Thus we denote the primitive states of the individual bases occurring in exons, introns, and junk by members of the sets

Exon states = { e_0, e_1, e_2 },
Intron states = { i_0, i_1, i_2 }, and
Junk state = {j}.

For example, we have 3 possible intron framings indicated in the following state strings.

$jj\ldots je_0e_1e_2\ldots e_0\mathbf{i_0i_0}\ldots\mathbf{i_0}e_1\ldots e_0e_1e_2jj\ldots j$ (intron frame 0)
$jj\ldots je_0e_1e_2\ldots e_1\mathbf{i_1i_1}\ldots\mathbf{i_1}e_2\ldots e_0e_1e_2jj\ldots j$ (intron frame 1)
$jj\ldots je_0e_1e_2\ldots e_2\mathbf{i_2i_2}\ldots\mathbf{i_2}e_0\ldots e_0e_1e_2jj\ldots j$ (intron frame 2)

2.1.2 Forward and Reverse Encoding

Encodings for proteins can be found in both directions along the DNA strand such that the two directions rarely interfere with each other. In other words, a coding segment in one encoding direction does not interfere with or overlap a coding segment in the other encoding direction. Furthermore, when comparing exons, introns, and junk segments, the estimated probabilities of occurrence of specific base sequences are substantially different, yet when comparing forward and reverse encodings the differences of such probability estimates are negligible among sequences of a given segment type – exon, intron, or junk. Hence, it would be counterproductive to simply ignore reverse encodings by relegating them to a second pass of our prediction algorithm, since that would require the interpretation of (reverse) coding segments, or exons, as junk – contradictory to that indicated by any reasonable set of estimates of the probabilities of nucleotide occurrences.

Thus we incorporate *shadow* states indicating reversely encoded exons and introns into the state model of our HOHMM, denoted by the primitives by $\hat{e}$ and $\hat{\imath}$, respectively. For example, we illustrate the 3 possible intron framings for the reverse encoding as follows:

$\mathtt{jj...j}\hat{e}_2\hat{e}_1\hat{e}_0\ldots\hat{e}_1\boldsymbol{\hat{\imath}_0\hat{\imath}_0}\ldots\boldsymbol{\hat{\imath}_0}\hat{e}_0\ldots\hat{e}_2\hat{e}_1\hat{e}_0\mathtt{jj}\ldots\mathtt{j}$ (intron frame 0)
$\mathtt{jj...j}\hat{e}_2\hat{e}_1\hat{e}_0\ldots\hat{e}_2\boldsymbol{\hat{\imath}_1\hat{\imath}_1}\ldots\boldsymbol{\hat{\imath}_1}\hat{e}_1\ldots\hat{e}_2\hat{e}_1\hat{e}_0\mathtt{jj}\ldots\mathtt{j}$ (intron frame 1)
$\mathtt{jj...j}\hat{e}_2\hat{e}_1\hat{e}_0\ldots\hat{e}_0\boldsymbol{\hat{\imath}_2\hat{\imath}_2}\ldots\boldsymbol{\hat{\imath}_2}\hat{e}_2\ldots\hat{e}_2\hat{e}_1\hat{e}_0\mathtt{jj}\ldots\mathtt{j}$ (intron frame 2)

2.2 Limited Success with 1st Order HMM

We define the 1st order HMM as consisting of the following:

An observable alphabet, B
A hidden state alphabet, Λ
("Prior") Probability $P(\lambda)$ for all $\lambda \in \Lambda$
("Transition") Probability $P(\lambda_2|\lambda_1)$ for all $\lambda_1\ \lambda_2 \in \Lambda$
("Emission") Probability $P(b|\lambda)$ for all $\lambda \in \Lambda\ b \in B$,

Given the above, there are three classes of problems [**29**] which the HMM can be used to solve:

1. Evaluation - Determine the probability of occurrence of the observed sequence.

2. Learning - Determine the most likely emissions and transitions for the observed sequence.

3. Decoding (Viterbi) - Determine the most probable sequence of states emitting the observed sequence.

Here we focus only on the 3rd problem, the Viterbi decoding problem. The probability of a sequence of observables $B=b_0\ b_1 \ldots\ b_{n-1}$ being emitted by the sequence of hidden states $\Lambda=\lambda_0\ \lambda_1 \ldots\ \lambda_{n-1}$ is given by

$P(B, \Lambda) = P(B|\Lambda)\ P(\Lambda)$ (chain rule) where the chain rule as been applied, resulting in the two factors known in the traditional HMM as the *observation model* and the *state model*, respectively. In the 1st order HMM, the state model has the 1st order Markov property and the observation model is such that the current observation, b_i, depends only on the current state, λ_i. According to this, we may further factor the above probability as in the following:

$P(B|\Lambda)\ P(\Lambda) =$ (chain rule)

$P(b_0|\Lambda)\ P(b_1|\Lambda)\ldots P(b_{n-1}|\Lambda)$

$P(\lambda_0)P(\lambda_1|\lambda_0)P(\lambda_2|\lambda_0, \lambda_1)\ldots P(\lambda_{n-1}|\lambda_0 \ldots \lambda_{n-2})$

$=$ (1st order HMM states)

$P(b_0|\Lambda)\ P(b_1|\Lambda)\ldots P(b_{n-1}|\Lambda)$

$P(\lambda_0)P(\lambda_1|\lambda_0)P(\lambda_2|\lambda_1)\ldots P(\lambda_{n-1}|\lambda_{n-2})$

$=$ (1st order HMM observations)

$P(b_0|\lambda_0)\ P(b_1|\lambda_1)\ldots P(b_{n-1}|\lambda_{n-1})$

$P(\lambda_0)P(\lambda_1|\lambda_0)P(\lambda_2|\lambda_0, \lambda_1)\ldots P(\lambda_{n-1}|\lambda_0 \ldots \lambda_{n-2})$

$=$ (regrouping)

$P(\lambda_0)\ P(b_0|\lambda_0)$

$P(\lambda_1|\lambda_0)\ P(b_1|\lambda_1)$

$\ldots$

$P(\lambda_{n-1}|\lambda_{n-2})\ P(b_{n-1}|\lambda_{n-1})$

According to [**30**], a *factor graph* is a tool frequently used in information theory to visually specify the *belief propagation* algorithm by which information is propagated throughout the model. Referring to the diagram in Figure 1, a factor graph consists of a bipartite graph containing two sets of nodes V and F for variables $s \in V$ and factor nodes $f \in F$ respectively, where variables (shown as circles) can be connected by edges to factor nodes (shown as squares) but not other variables and vice versa.

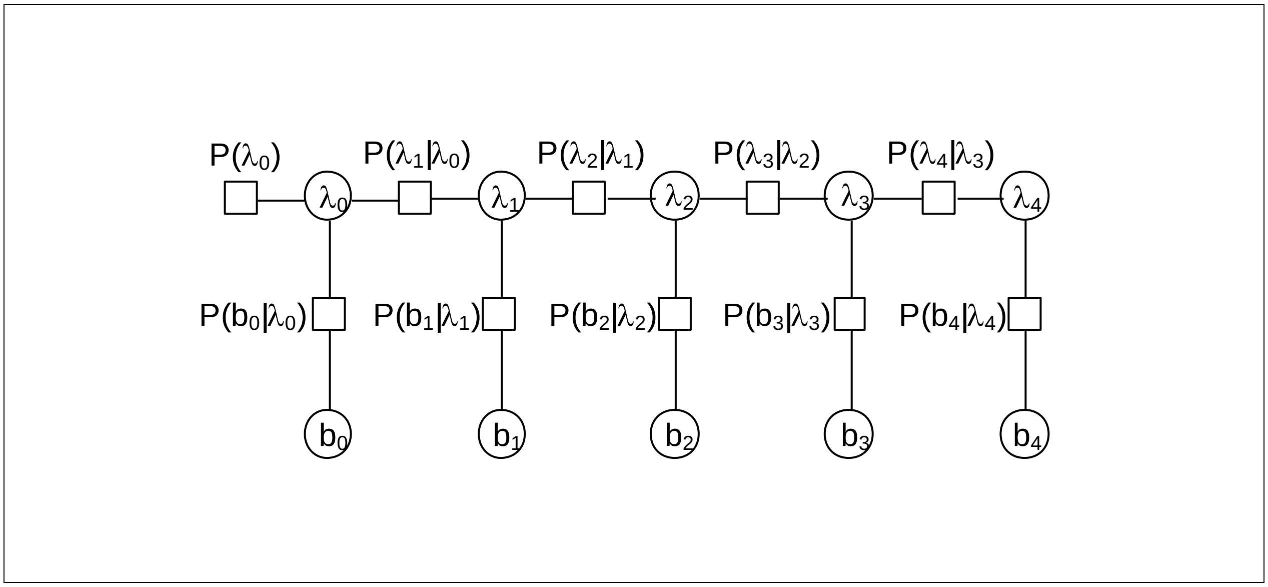


Figure 1 Factor graph for traditional, 1st order HMM.

The Viterbi algorithm is encoded in the factor graph using the max-product form of the belief propagation algorithm. In order to implement the Viterbi algorithm, each of the variable (square) factor nodes along the top row is to be interpreted as taking a maximum of the probability with respect to the previous state. This results in two recursion relations, one for each of two types of values, 1) the total probability at each observation, b_i and 2) the most probable state at each observation, b_i . In practice, these recursively defined quantities are kept in two tables, a *probability table* and a *back pointer table*, with one column in each of the two tables for each observation, b_i , and one cell or row in each column for each of the possible values of the state, λ_i, at observation, b_i .

The recursive expression for the total probability, $P(b_i, \lambda_i)$ at each observation, b_i , as follows:

$$P(b_i, \lambda_i) = P(b_i| \lambda_i) \max_j [P(\lambda_i| (\lambda_{i-1})^j) \; P(b_{i-1},(\lambda_{i-1})^j)]$$

where the maximum is taken over all possible values of the previous state,

$(\lambda_{i-1})^j$ for j = 1, … $|\Lambda|$, and

$| \Lambda |$ = the size of the state alphabet, Λ.

In order to avoid having to recompute probabilities, the most probable state at each observation is computed once in dynamic programming fashion as determined by the recursive expression:

$$(\lambda_{i-1})^* = \arg\max_j P(b_i, \lambda_i) = \arg\max_j [P(\lambda_i| (\lambda_{i-1})^j) \; P(b_{i-1},(\lambda_{i-1})^j)]$$

The recursion terminates with the last observation, b_i , and the most probable path of states is given by

$$\Lambda^* = \text{argmax}_{\Lambda} P(B, \Lambda)$$
$$= (\lambda_0^*, \ldots, \lambda_4^*)$$

where the decoding consists of retracing through the table of back pointers starting with

$$\lambda_4^* = \text{argmax}_j [P(b_4, (\lambda_4)^j)]$$

With genomic data, Viterbi decoding is used to identify the coding/non-coding transition sites, such as the exon-intron and intron-exon transitions, referred to as *donor* and *acceptor splice sites*, respectively. Thus the job of gene prediction is to predict the precise locations of such c/nc transitions within a string of genomic data. First order hidden Markov models (HMM's) have been used here and elsewhere for the prediction of donor and acceptor splice sites with limited success [**4**]. This can be attributed to the notion that using first order statistics for c/nc sites effectively under weights such transitions (relative to other homogeneous sites) when higher-order statistics are more appropriate in regions centered more or less on the c/nc transitions. (Similar though weaker arguments can be made for junk-exon and exon-junk transitions as well. See, for example, [**4**].) This is typically revealed, for example as in **3.7 Relative Entropy of C/NC Transitions** by the extent of anomalous base statistics in the neighborhood of the transition.

2.3 Higher Order HMM (HOHMM) Specifications

As stated above, traditional 1st order HMM's assume that a 1st order Markov property holds among the states and that each observable depends only on the corresponding state and not any other observable. However, the current work entails a departure from these conventions in an attempt to leverage information from the entire region of anomalous statistics in the neighborhood of c/nc transitions (when compared to base statistics of coding or non-coding regions away from such transitions). The regions of anomalous statistics are often highly structured, having consensus sequences that strongly depart from the strong independence assumptions of the 1st order HMM. The existence of such consensus sequences suggests that we adopt an observation model that has a higher order Markov property with respect to the observations. Furthermore, since the consensus sequences vary by the type of transition, this observational Markov order should be allowed to vary depending on the state – with higher order for non-splice sites and lower order for splice sites (where interpolation is through highest order supported by the size of the training data).

Furthermore, in the Viterbi context, for a given state dimer transition or non-transition, such as e_0e_1 or e_0i_0, respectively, we can compound the contributions of the corresponding base emissions to the correct prediction of state by using extended states. For example when encountered sequentially in the Viterbi algorithm, the following sequence of extended or *footprint* states would conceivably score highly when computed in the given order at consecutive positions in the vicinity of a consensus sequence for a donor splice site.

```
eeeeeei,
 eeeeeii,
  eeeeiii,
   eeeiiii,
    eeiiiii, and
     eiiiiii
```

In other words we can expect a *natural boosting* effect for the correct prediction at such c/nc transitions.

In order to accommodate the above concerns, we consider the pairing of an extended observation centered on the primitive observation b_i with an extended state centered on the dimer state s_i (see below for notation), where the observation and state extents are parameterized independently according to the following.

1) non-negative integers L and R denoting left and right maximum extents of a substring, w_i, (with suitable truncation at the data boundaries, b_0 and b_{n-1}) associated with the primitive observation, b_i , in the following way,

   $w_i = b_{i-L+1}, \ldots, b_i, \ldots, b_{i+R}$
   $\hat{w}_i = b_{i-L+1}, \ldots, b_i, \ldots, b_{i+R-1}$

   for $i = -R, \ldots, n-R-1$, where

   $L \geq 0$, (left extent for bases in w_i)
   $R \geq 0$, (right extent for bases in w_i)
   $L+R > 0$. (total extent for bases in w_i)

2) non-negative integers ℓ and r denoting the left and right extents the extended (footprint) states, f, expressed usually in practice in terms of its component, *overlapping* dimer states, s. Here, we show the relationships among the primitive states λ, dimer states s, footprint states f, and footprint transitions σ.

   $s_i = \lambda_i\lambda_{i+1}$ (dimer state, length in λ's =2)
   $f_i = s_{i-\ell+1}, \ldots, s_{i+r} \cong \lambda_{i-\ell+1}, \ldots, \lambda_i, \ldots, \lambda_{i+r+1}$ (footprint state, length in s's= ℓ+r)
   $\sigma_i = f_{i-1}, f_i$ (length in s's= ℓ+r+1, length in λ's= ℓ+r+2)
   $\cong s_{i-\ell}, \ldots, s_{i+r} \cong \lambda_{i-\ell}, \ldots, \lambda_i, \ldots, \lambda_{i+r+1}$

   where

   $\ell \geq 0$, (left extent of f or $s_{i-\ell+1}, \ldots, s_i$)
   $r \geq 0$, (right extent of f or $s_{i+1}, \ldots, s_{i+r}$)
   $\ell+r > 0$. (total extent of f in s's)

3) a non-negative integer function $M(f_{i-1}, f_i, j)$ denoting a variable Markov order among the

bases in the string w_i with

$M(f_{i-1}, f, j) \geq 0$, where

f_{i-1}, f_i = sequential footprint states,
j = the index of any member of w_i in the range $i-L+1 \leq j \leq i+R$

We note that there is no other restriction on the relative sizes of the four parameters L, R, l, and r. As in the 1st order HMM case, we assume that the ith base observation b_i is aligned with the ith hidden state λ_i. However we note in the following – as a departure from the 1st order case – that we modify the condition that b_i is dependent exclusively on the hidden state λ_i. We illustrate this pairing of specific observation strings and hidden state strings at either end and at an interior position in the processing via the diagram in Figure 2 below.

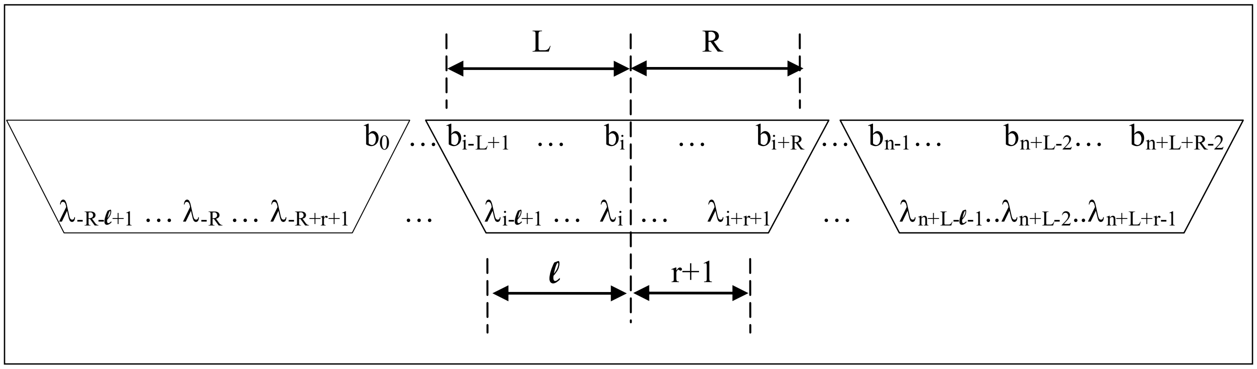

L
R
b_0
...
b_{i-L+1} ... b_i ... b_{i+R}
...
b_{n-1}... b_{n+L-2}... $b_{n+L+R-2}$
$\lambda_{-R-\ell+1}$ ... λ_{-R} ... λ_{-R+r+1}
...
$\lambda_{i-\ell+1}$ ... λ_i ... λ_{i+r+1}
...
$\lambda_{n+L-\ell-1}$..λ_{n+L-2}..$\lambda_{n+L+r-1}$
ℓ
r+1

Figure 2 Stepwise association (or clique) of observations and hidden states in HOHMM

Referring to the diagram above, assuming a left-to-right traversal of the data, many choices of first and last clique are possible depending upon the desired point of onset and strength of influence from the emission probabilities in the prediction process of the Viterbi algorithm. In the case of protein product data, such choices could be critical since there is typically little, if any, buffer extent at either ends of the data for the Viterbi algorithm to recover from poorly estimated state prior probabilities. (See also **2.8 Estimates of Probabilities from Counts**, page 37.)

For convenience, we have chosen the first such that the first observations, b_0, is included at the leading edge. This allows the emission probabilities to exert the earliest influence on the prediction process, though the influence is weakest, since only one base, b_0, is contributing at the outset. However, for the last clique, we recognize the need to predict no further than the last observation, b_{n-1}, but we choose the trailing edge to include the last observation for convenience at the cost of making predictions for states beyond the last observation.

With this choice of first and last clique, we observe that in order to satisfy the conditions at the left and right boundaries, we have introduced some additional sets of state and observation primitives (with associated, unit-valued transition and emission probabilities) for suitable values of L, R, ℓ, and r. These additional primitives are shown in Table 2 below.

Table 2 Additional primitives for completion of boundary cliques

Additional Primitives	**Type of Primitive**	**Boundary**
$\lambda_{-R-\ell+1}, \ldots, \lambda_{-1}$	States	Left
$b_{n}, \ldots, b_{n+L+R-2}$	Observations	Right
$\lambda_{n}, \ldots, \lambda_{n+L+r+1}$	States	Right

In **2.5**, page 27, and **2.6**, page 34, we will investigate two different factorizations of the total probability for states and observations

$$P(B, \Lambda) = P(b_0, \ldots, b_{n+L+R-2} ; \lambda_{-R-\ell+1}, \ldots, \lambda_{n+L+r-1})$$

noting that both factorizations represent departures in their own varying degrees from the factorization of the traditional HMM. The first factorization departs immediately from the traditional separation between observation and state models, but the second does at least initially separate the observation and state models.

We will observe that both factorizations allow for an alternate representation such that the internal scalar-based state discriminant can be replaced with a vector-based discriminant. This would allow the substitution of a discriminant based on a Support Vector Machine (SVM) as demonstrated for splice sites in [**25**]. Also, we note that these alternate representations would not introduce any significant increase in computational complexity, since the SVM-based discriminant, having been trained offline, would require the computation of a simple vector dot product.

2.3.1 Assumptions

We demonstrate in **2.4 Linear Growth in State/Transition Space** below the fact that the number of allowed transitions among footprint states is restricted to the linear growth in the size of the footprint states. First, we first list the assumptions used as well as some mostly notational conventions in **2.3.2 Conventions** below.

1) The length of introns and exons are bounded below, such that the following hold:
    a. footprint states can contain at most one coding/non-coding (c/nc) transition
    b. footprint state transitions can involve at most one c/nc transition among both the source and destination footprint states

   (This is referred to as the *minimum length assumption*. See parameter F defined below in **Conventions**.)
2) The length distributions of introns and exons are taken to be geometric by default. (See [**20**] and [**4**].)
3) We introduce support in the model for the notions of reading frame (0, 1, 2) and encoding direction (forward, reverse), thus replacing the basic exon e and intron i state primitives with the set of state primitives $e_0,e_1,e_2,i_0,i_1,i_2,\hat{e}_0,\hat{e}_1,\hat{e}_2,\hat{i}_0,\hat{i}_1,\hat{i}_2$. Without such explicit support in the model for the reverse encoding, we incur undue risk of indeterminate results [**24**]. For example, consider the case of a small, reverse encoded gene contained entirely within a forward encoded intron. As the prediction process traverses forward through the data, the encounter with the reverse encoded gene can result in sufficient cumulative reduction of the computed state path probabilities, such that the ensuing end of the forward encoded intron would go undetected.

4) We make no attempt to encode the notions of reading frame or encoding direction in the case of junk, hence designating a single primitive junk state, j.
5) The base Markov order, with maximum value specified by parameter M of the HOHMM, is allowed to vary below the maximum M as a function of both footprint state as well as the position of the observed base relative to the c/nc transition site within the footprint state.
6) We make no attempt thus far to vary the base Markov emission extent with respect to the encoding direction. For example, we use the value R to designate the right base emission extent for both the 3'-side in the context of forward transitions as well as the 5'-side in the context of reverse (shadow) transitions. Whereas, otherwise, we would anticipate some overall improvement in prediction accuracy of the HOHMM.

2.3.2 Conventions

1) According to Figure 2 above, a footprint state is an extended state consisting of ℓ+r+1 primitive states. In practice, however, we use a symmetric footprint state, where ℓ = r, and we denote the size of the footprint states for a given instance of an HOHMM by a single parameter F, where F is measured in units of consecutive, overlapping dimers. Thus, we write

$$F = 2\ell = 2r = \text{length of footprint state in dimers.}$$

2) In the following we will use the symbols ê, î in order to denote exon and intron primitive states for the reverse encoding direction. Yet when we refer to the corresponding footprint states by name the primitives will be listed in 5'-to-3', encoding order, such as the dimer $\hat{e}_0\hat{i}_0$, and the dimer transition $\hat{e}_0\hat{i}_0$ to $\hat{i}_0\hat{i}_0$. Thus, we have the following complete set of state primitives.

Set of state primitives = { $e_0, e_1, e_2, i_0, i_1, i_2, \hat{e}_0, \hat{e}_1, \hat{e}_2, \hat{i}_0, \hat{i}_1, \hat{i}_2, j$ }

3) The reading frame varies cyclically among the primitives within an exon, such as $e_0e_1e_2$, whereas the reading frame of an intron is fixed over the entire segment of intron primitives, such as $i_0i_0i_0$.
4) By convention, the reading frame at the exon-intron interface does not change, such as e_0i_0, whereas the reading frame at the intron-exon interface does, such as i_0e_1.
5) We observe that introns are artificially split into the primitives i_0, i_1, i_2 and $\hat{i}_0, \hat{i}_1, \hat{i}_2$ for the forward and reverse encoding directions, respectively, but are not expected to have notably different statistical profiles. Such is the case for example in *C. elegans* upon inspection, and thus they are pooled as one "i" statistic in the training procedure.
5) Six different footprint states will be used to capture the three possible stop codons, TAA, TAG, and TGA – 3 each for the forward and reverse encoding directions.

2.4 Linear Growth in State/Transition Space

In order to control the exponential explosion of the set of footprint states – as well as the set of allowed transitions – we use the observation in nature of a minimum length for exons, introns and junk segments. This observation translates to a rule in the implementation that for any footprint state f_i containing a primitive transition the following f_{i+1} cannot contain any new, additional primitive transition as in the following:

If the minimum length = 6 (as in footprint size 6 in units of s states) and

$f_i = \texttt{eeeeeei}$

then we must have

$f_{i+1} = \texttt{eeeeeii}$

and not

$f_{i+1} = \texttt{eeeeeie}$. (intron is too short – not allowed).

The following subsections were taken from [**24**], where it was shown that this restriction on minimum length results in limiting the growth of the possible set of states to linear in the state string length, as is discussed briefly in [**23**]. For an alternate calculation of this linear bound without employing the full enumeration, see **2.10 Alternate Proof of Linear Upper Bound on Footprint States and Transitions**, page 37.

2.4.1 Enumeration of the 33 Dimer States

A dimer state is a footprint state consisting of two state primitives. There are two types of dimer states, those that include a c/nc transition, and those that do not, which we refer to as the eij-type dimers and xx-dimers, such as e_0i_0 and e_0e_1, respectively.

As a direct consequence of the assumptions 2, 3, and 4 above, we have a total of 33 possible dimer states, which can be enumerated as follows:

1) 13 xx-type (homogeneous) dimers
   a. 6 Intron-intron – i_0i_0, i_1i_1, i_2i_2, $\hat{\imath}_0\hat{\imath}_0$, $\hat{\imath}_1\hat{\imath}_1$, $\hat{\imath}_2\hat{\imath}_2$
   b. 6 Exon-exon – e_0e_1, e_1e_2, e_2e_0, $\hat{e}_0\hat{e}_1$, $\hat{e}_1\hat{e}_2$, $\hat{e}_2\hat{e}_0$
   c. 1 Junk-junk – jj
2) 20 eij-type (heterogeneous) dimers
   a. 6 Exon-intron – e_0i_0, e_1i_1, e_2i_2, $\hat{e}_0\hat{\imath}_0$, $\hat{e}_1\hat{\imath}_1$, $\hat{e}_2\hat{\imath}_2$
   b. 6 Intron-exon – i_0e_1, i_1e_2, i_2e_0, $\hat{\imath}_0\hat{e}_1$, $\hat{\imath}_1\hat{e}_2$, $\hat{\imath}_2\hat{e}_0$
   c. 6 Exon-junk – $(e_2j)_{TAA}$, $(e_2j)_{TAG}$, $(e_2j)_{TGA}$, $(\hat{e}_2j)_{TAA}$, $(\hat{e}_2j)_{TAG}$, $(\hat{e}_2j)_{TGA}$
   d. 2 Junk-exon – (je_0), $(j\hat{e}_0)$

2.4.2 Enumeration of the Footprint States

As a consequence of assumption 1) above, footprint states fall into the same two categories or types, xx-type and eij-type, as the dimers above. Regardless of footprint state type, each footprint state can be considered to be generated by the xx-type dimer that it contains. For xx-types, it is sufficient to specify the generating dimer only, such as i_0i_0 for the xx-type footprint state $i_0i_0\ldots i_0$. For eij-types, a position must also be specified for the location of the generating dimer within the generated footprint state.

By definition, the number of xx-type footprint states is identical to the number of xx-type dimers, as enumerated in Table 3 below.

Table 3 All 13 xx-type footprint states generated by the xx-type dimers

Dimer Index	Xx- type Generating Dimer	Xx- type Generated Footprint State
0	i_0i_0	$i_0i_0\ldots i_0$
1	i_1i_1	$i_1i_1\ldots i_1$
2	i_2i_2	$i_2i_2\ldots i_2$
3	$\hat{i}_0\hat{i}_0$	$\hat{i}_0\hat{i}_0\ldots \hat{i}_0$
4	$\hat{i}_1\hat{i}_1$	$\hat{i}_1\hat{i}_1\ldots \hat{i}_1$
5	$\hat{i}_2\hat{i}_2$	$\hat{i}_2\hat{i}_2\ldots \hat{i}_2$
6	e_0e_1	$e_0e_1\ldots e_{(F)\mathrm{mod}3}$
7	e_1e_2	$e_1e_2\ldots e_{(F+1)\mathrm{mod}3}$
8	e_2e_0	$e_2e_0\ldots e_{(F-1)\mathrm{mod}3}$
9	$\hat{e}_0\hat{e}_1$	$\hat{e}_0\hat{e}_1\ldots \hat{e}_{(F)\mathrm{mod}3}$
10	$\hat{e}_1\hat{e}_2$	$\hat{e}_1\hat{e}_2\ldots \hat{e}_{(F+1)\mathrm{mod}3}$
11	$\hat{e}_2\hat{e}_0$	$\hat{e}_2\hat{e}_0\ldots \hat{e}_{(F-1)\mathrm{mod}3}$
12	jj	$jj\ldots j$

As for the eij-type footprint states, each is generated by the non-homogeneous dimer that it contains but is further characterized by the position of the generating dimer within the footprint string, such as e_0i_0 in the right-most position of the eij-type footprint state $e_{(F-2)\mathrm{mod}3}e_{(F-1)\mathrm{mod}3}\ldots e_0e_0e_0i_0$.

As a consequence of assumption 1 above, there are F eij-type footprint states for each corresponding eij-type dimer. This can be elaborated as follows. First, observe that assumption 1 above implies that a footprint state can contain at most one, if any, eij-type dimer, such as the eij-type dimer e_2i_0 in the eij-type footprint $e_0e_1e_2i_0i_0i_0$. Hence, given an eij-type footprint state of length F in dimers, there are precisely F possible positions for the implied eij-type dimer to occur within the footprint state's string of primitives. These dimer positions are labeled 0, …, F-1 and taken in the order of encoding (forward or reverse) in

Table 4 below. Thus we have the relation

# eij-type footprint states
= 20 (F)
= (# eij-type dimer states) (F)

Table 4 All 20(F) eij-type footprint states generated by the eij-type dimers

Dimer Index	Eij-type Generating Dimer	Eij-type Generated Footprint State For Generating Dimer Positions 0, …, F-1		
		0	**…**	**F-1**
13	e_0i_0	$e_0i_0\ldots i_0$	…	$e_{(1-F)mod3}e_{(2-F)mod3}\ldots e_0i_0$
14	e_1i_1	$e_1i_1\ldots i_1$	…	$e_{(2-F)mod3}e_{(-F)mod3}\ldots e_1i_1$
15	e_2i_2	$e_2i_2\ldots i_2$	…	$e_{(-F)mod3}e_{(1-F)mod3}\ldots e_2i_2$
16	$\hat{e}_0\hat{\imath}_0$	$\hat{e}_{(1-F)mod3}\hat{e}_{(2-F)mod3}\ldots\hat{e}_0\hat{\imath}_0$	…	$\hat{e}_0\hat{\imath}_0\ldots\hat{\imath}_0$
17	$\hat{e}_1\hat{\imath}_1$	$\hat{e}_{(2-F)mod3}\hat{e}_{(-F)mod3}\ldots\hat{e}_1\hat{\imath}_1$	…	$\hat{e}_1\hat{\imath}_1\ldots\hat{\imath}_1$
18	$\hat{e}_2\hat{\imath}_2$	$\hat{e}_{(-F)mod3}\hat{e}_{(1-F)mod3}\ldots\hat{e}_2\hat{\imath}_2$	…	$\hat{e}_2\hat{\imath}_2\ldots\hat{\imath}_2$
19	i_0e_1	$i_0e_1e_2\ldots e_{(F)mod3}$	…	$i_0\ldots i_0e_1$
20	i_1e_2	$i_1e_2e_0\ldots e_{(F+1)mod3}$	…	$i_1\ldots i_1e_2$
21	i_2e_0	$i_2e_0e_1\ldots e_{(F-1)mod3}$	…	$i_2\ldots i_2e_0$
22	$\hat{\imath}_0\hat{e}_1$	$\hat{\imath}_0\ldots\hat{\imath}_0\hat{e}_1$	…	$\hat{\imath}_0\hat{e}_1\hat{e}_2\ldots\hat{e}_{(F)mod3}$
23	$\hat{\imath}_1\hat{e}_2$	$\hat{\imath}_1\ldots\hat{\imath}_1\hat{e}_2$	…	$\hat{\imath}_1\hat{e}_2\hat{e}_0\ldots\hat{e}_{(F+1)mod3}$
24	$\hat{\imath}_2\hat{e}_0$	$\hat{\imath}_2\ldots\hat{\imath}_2\hat{e}_0$	…	$\hat{\imath}_2\hat{e}_0\hat{e}_1\ldots\hat{e}_{(F-1)mod3}$
25	$(e_2j)_{TAA}$	$(e_2j)_{TAA}jj\ldots j$	…	$e_{(-F)mod3}e_{(1-F)mod3}\ldots(e_2j)_{TAA}$
26	$(e_2j)_{TAG}$	(Similar to above)	…	(Similar to above)
27	$(e_2j)_{TGA}$	“ “	…	“ “
28	$(\hat{e}_2j)_{TAA}$	$\hat{e}_{(-F)mod3}\hat{e}_{(1-F)mod3}\ldots(\hat{e}_2j)_{TAA}$	…	$(\hat{e}_2j)_{TAA}j\ldots j$
29	$(\hat{e}_2j)_{TAG}$	(Similar to above)	…	(Similar to above)
30	$(\hat{e}_2j)_{TGA}$	“ “	…	“ “
31	je_0	$je_0e_1\ldots e_{(F-1)mod3}$	…	$jj\ldots je_0$
32	$j\hat{e}_0$	$jj\ldots j\hat{e}_0$	…	$j\hat{e}_0\hat{e}_1\ldots\hat{e}_{(F-1)mod3}$

2.4.3 Enumeration of the Allowed Footprint State Transitions

As an additional consequence of assumption 1 above, there are two different types of transitions among footprint states, depending on whether they introduce a new c/nc transition in the second footprint state of the footprint transition.

The transitions of the first type share the property that they *do not* introduce a new c/nc transition in the second footprint state. These transitions comprise a group of size (13+20(F)) and are thus in 1-to-1 correspondence with the entire set of footprint states.

As listed below in Table 5, the first 13 of these transitions are xx-type to xx-type, such as $i_0\ldots i_0$ to itself, and in Table 6 the other 20(F) transitions are either eij-type to eij-type or eij-type to xx-type.

Table 5 All 13 (xx-type) to (xx-type) footprint transitions

Dimer Index	Xx-type Generating Dimer	Xx-type Footprint Transition	
		Generated Source Footprint State	Destination Footprint State
0	i_0i_0	$i_0i_0\ldots i_0$	$i_0i_0\ldots i_0$
1	i_1i_1	$i_1i_1\ldots i_1$	$i_1i_1\ldots i_1$
2	i_2i_2	$i_2i_2\ldots i_2$	$i_2i_2\ldots i_2$
3	$\hat{i}_0\hat{i}_0$	$\hat{i}_0\hat{i}_0\ldots \hat{i}_0$	$\hat{i}_0\hat{i}_0\ldots \hat{i}_0$
4	$\hat{i}_1\hat{i}_1$	$\hat{i}_1\hat{i}_1\ldots \hat{i}_1$	$\hat{i}_1\hat{i}_1\ldots \hat{i}_1$
5	$\hat{i}_2\hat{i}_2$	$\hat{i}_2\hat{i}_2\ldots \hat{i}_2$	$\hat{i}_2\hat{i}_2\ldots \hat{i}_2$
6	e_0e_1	$e_0e_1\ldots e_{(F)\mathrm{mod}3}$	$e_1e_2\ldots e_{(F+1)\mathrm{mod}3}$
7	e_1e_2	$e_1e_2\ldots e_{(F+1)\mathrm{mod}3}$	$e_2e_0\ldots e_{(F-1)\mathrm{mod}3}$
8	e_2e_0	$e_2e_0\ldots e_{(F-1)\mathrm{mod}3}$	$e_0e_1\ldots e_{(F)\mathrm{mod}3}$
9	$\hat{e}_0\hat{e}_1$	$\hat{e}_0\hat{e}_1\ldots \hat{e}_{(F)\mathrm{mod}3}$	$\hat{e}_2\hat{e}_0\ldots \hat{e}_{(F-1)\mathrm{mod}3}$
10	$\hat{e}_1\hat{e}_2$	$\hat{e}_1\hat{e}_2\ldots \hat{e}_{(F+1)\mathrm{mod}3}$	$\hat{e}_0\hat{e}_1\ldots \hat{e}_{(F)\mathrm{mod}3}$
11	$\hat{e}_2\hat{e}_0$	$\hat{e}_2\hat{e}_0\ldots \hat{e}_{(F-1)\mathrm{mod}3}$	$\hat{e}_1\hat{e}_2\ldots \hat{e}_{(F+1)\mathrm{mod}3}$
12	jj	$jj\ldots j$	$jj\ldots j$

Table 6 All 20(F) eij-type footprint transitions

Dimer Index	**Eij-type Generating Dimer**	**Eij-type Generated Footprint Transitions For Generating Dimer Positions 0, …, F-1**				
		0		**…**	**F-1**	
		Generated Source Footprint State	**Destination Footprint State**		**Generated Source Footprint State**	**Destination Footprint State**
13	$e_0 i_0$	$e_0 i_0 \ldots i_0$	$i_0 i_0 \ldots i_0$	…	$e_{(1-F)\mathrm{mod}3} \ldots e_0 i_0$	$e_{(2-F)\mathrm{mod}3} \ldots e_0 i_0 i_0$
14	$e_1 i_1$	$e_1 i_1 \ldots i_1$	$i_1 i_1 \ldots i_1$	…	$e_{(2-F)\mathrm{mod}3} \ldots e_1 i_1$	$e_{(-F)\mathrm{mod}3} \ldots e_1 i_1 i_1$
15	$e_2 i_2$	$e_2 i_2 \ldots i_2$	$i_2 i_2 \ldots i_2$	…	$e_{(-F)\mathrm{mod}3} \ldots e_2 i_2$	$e_{(1-F)\mathrm{mod}3} \ldots e_2 i_2$
16	$\hat{e}_0 \hat{\imath}_0$	$\hat{e}_{(1-F)\mathrm{mod}3} \ldots \hat{e}_0 \hat{\imath}_0$	$\hat{e}_{(-F)\mathrm{mod}3} \ldots \hat{e}_2 \hat{e}_0$	…	$\hat{e}_0 \hat{\imath}_0 \ldots \hat{\imath}_0$	$\hat{e}_2 \hat{e}_0 \hat{\imath}_0 \ldots \hat{\imath}_0$
17	$\hat{e}_1 \hat{\imath}_1$	$\hat{e}_{(2-F)\mathrm{mod}3} \ldots \hat{e}_1 \hat{\imath}_1$	$\hat{e}_{(1-F)\mathrm{mod}3} \ldots \hat{e}_0 \hat{e}_1$	…	$\hat{e}_1 \hat{\imath}_1 \ldots \hat{\imath}_1$	$\hat{e}_0 \hat{e}_1 \hat{\imath}_1 \ldots \hat{\imath}_1$
18	$\hat{e}_2 \hat{\imath}_2$	$\hat{e}_{(-F)\mathrm{mod}3} \ldots \hat{e}_2 \hat{\imath}_2$	$\hat{e}_{(2-F)\mathrm{mod}3} \ldots \hat{e}_1 \hat{e}_2$	…	$\hat{e}_2 \hat{\imath}_2 \ldots \hat{\imath}_2$	$\hat{e}_1 \hat{e}_2 \hat{\imath}_2 \ldots \hat{\imath}_2$
19	$i_0 e_1$	$i_0 e_1 e_2 \ldots e_{(F)\mathrm{mod}3}$	$e_1 e_2 \ldots e_{(F+1)\mathrm{mod}3}$	…	$i_0 \ldots i_0 e_1$	$i_0 \ldots i_0 e_1 e_2$
20	$i_1 e_2$	$i_1 e_2 e_0 \ldots e_{(F+1)\mathrm{mod}3}$	$e_2 e_0 \ldots e_{(F-1)\mathrm{mod}3}$	…	$i_1 \ldots i_1 e_2$	$i_1 \ldots i_1 e_2 e_0$
21	$i_2 e_0$	$i_2 e_0 e_1 \ldots e_{(F-1)\mathrm{mod}3}$	$e_0 e_1 \ldots e_{(F)\mathrm{mod}3}$	…	$i_2 \ldots i_2 e_0$	$i_2 \ldots i_2 e_0 e_2$
22	$\hat{\imath}_0 \hat{e}_1$	$\hat{\imath}_0 \ldots \hat{\imath}_0 \hat{e}_1$	$\hat{\imath}_0 \ldots \hat{\imath}_0$	…	$\hat{\imath}_0 \hat{e}_1 \ldots \hat{e}_{(F)\mathrm{mod}3}$	$\hat{\imath}_0 \hat{\imath}_0 \hat{e}_1 \ldots \hat{e}_{(F-1)\mathrm{mod}3}$
23	$\hat{\imath}_1 \hat{e}_2$	$\hat{\imath}_1 \ldots \hat{\imath}_1 \hat{e}_2$	$\hat{\imath}_1 \ldots \hat{\imath}_1$	…	$\hat{\imath}_1 \hat{e}_2 \ldots \hat{e}_{(F+1)\mathrm{mod}3}$	$\hat{\imath}_1 \hat{\imath}_1 \hat{e}_2 \ldots \hat{e}_{(F)\mathrm{mod}3}$
24	$\hat{\imath}_2 \hat{e}_0$	$\hat{\imath}_2 \ldots \hat{\imath}_2 \hat{e}_0$	$\hat{\imath}_2 \ldots \hat{\imath}_2$	…	$\hat{\imath}_2 \hat{e}_0 \ldots \hat{e}_{(F-1)\mathrm{mod}3}$	$\hat{\imath}_2 \hat{\imath}_2 \hat{e}_0 \ldots \hat{e}_{(F+1)\mathrm{mod}3}$
25	$(e_2 j)_{\mathrm{TAA}}$	$(e_2 j)_{\mathrm{TAA}} j j \ldots j$	$j j \ldots j$	…	$e_{(-F)\mathrm{mod}3} \ldots (e_2 j)_{\mathrm{TAA}}$	$e_{(1-F)\mathrm{mod}3} \ldots (e_2 j)_{\mathrm{TAA}} j$
26	$(e_2 j)_{\mathrm{TAG}}$	(Similar to above)	(Same as above)	…	(Similar to above)	(Similar to above)
27	$(e_2 j)_{\mathrm{TGA}}$	“ “	“ “	…	“ “	“ “
28	$(\hat{e}_2 j)_{\mathrm{TAA}}$	$\hat{e}_{(-F)\mathrm{mod}3} \ldots (\hat{e}_2 j)_{\mathrm{TAA}}$	$\hat{e}_{(2-F)\mathrm{mod}3} \ldots \hat{e}_1 \hat{e}_2$	…	$(\hat{e}_2 j)_{\mathrm{TAA}} j \ldots j$	$\hat{e}_1 (\hat{e}_2 j)_{\mathrm{TAA}} j \ldots j$
29	$(\hat{e}_2 j)_{\mathrm{TAG}}$	(Similar to above)	(Same as above)	…	(Similar to above)	(Similar to above)
30	$(\hat{e}_2 j)_{\mathrm{TGA}}$	“ “	“ “	…	“ “	“ “
31	$j e_0$	$j e_0 e_1 \ldots e_{(F-1)\mathrm{mod}3}$	$e_0 e_1 \ldots e_{(F)\mathrm{mod}3}$	…	$j j \ldots j e_0$	$j j \ldots j e_0 e_1$
32	$j \hat{e}_0$	$j \ldots j \hat{e}_0$	$j j \ldots j$	…	$j \hat{e}_0 \ldots \hat{e}_{(F-1)\mathrm{mod}3}$	$j j \hat{e}_0 \ldots \hat{e}_{(F+1)\mathrm{mod}3}$

The transitions of the second type share the property that they *do* introduce a new c/nc transition in the second footprint state. These transitions comprise a group of size 20 and thus in each case involve a transition from an xx-type footprint to an eij-type footprint.

These transitions can be subdivided, in turn, depending on whether they involve junk, as listed in Table 7 and Table 8 below, respectively.

Table 7 All 12 footprint transitions introducing a new c/nc transition *and* not involving junk

Dimer Index	Xx-type Generating Dimer	c/nc-type Footprint Transition	
		Generated Source Footprint State	Destination Footprint State
0	i_0i_0	$i_0i_0 \ldots i_0$	$i_0 \ldots i_0e_1$
1	i_1i_1	$i_1i_1 \ldots i_1$	$i_1 \ldots i_1e_2$
2	i_2i_2	$i_2i_2 \ldots i_2$	$i_2 \ldots i_2e_0$
3	$\hat{i}_0\hat{i}_0$	$\hat{i}_0\hat{i}_0 \ldots \hat{i}_0$	$\hat{e}_0\hat{i}_0 \ldots \hat{i}_0$
4	$\hat{i}_1\hat{i}_1$	$\hat{i}_1\hat{i}_1 \ldots \hat{i}_1$	$\hat{e}_1\hat{i}_1 \ldots \hat{i}_1$
5	$\hat{i}_2\hat{i}_2$	$\hat{i}_2\hat{i}_2 \ldots \hat{i}_2$	$\hat{e}_2\hat{i}_2 \ldots \hat{i}_2$
6	e_0e_1	$e_0e_1 \ldots e_{(F)\text{mod}3}$	$e_1 \ldots e_{(F)\text{mod}3}i_{(F)\text{mod}3}$
7	e_1e_2	$e_1e_2 \ldots e_{(F+1)\text{mod}3}$	$e_2 \ldots e_{(F+1)\text{mod}3}i_{(F+1)\text{mod}3}$
8	e_2e_0	$e_2e_0 \ldots e_{(F-1)\text{mod}3}$	$e_0 \ldots e_{(F-1)\text{mod}3}i_{(F-1)\text{mod}3}$
9	$\hat{e}_0\hat{e}_1$	$\hat{e}_0\hat{e}_1 \ldots \hat{e}_{(F)\text{mod}3}$	$\hat{i}_2\hat{e}_0 \ldots \hat{e}_{(F-1)\text{mod}3}$
10	$\hat{e}_1\hat{e}_2$	$\hat{e}_1\hat{e}_2 \ldots \hat{e}_{(F+1)\text{mod}3}$	$\hat{i}_0\hat{e}_1 \ldots \hat{e}_{(F)\text{mod}3}$
11	$\hat{e}_2\hat{e}_0$	$\hat{e}_2\hat{e}_0 \ldots \hat{e}_{(F-1)\text{mod}3}$	$\hat{i}_1\hat{e}_2 \ldots \hat{e}_{(F+1)\text{mod}3}$

Table 8 All 8 footprint transitions introducing a new c/nc transition *and* involving junk

Dimer Index	Xx-type Generating Dimer	c/nc-type Footprint Transition	
		Generated Source Footprint State	Destination Footprint State
6 +(2-F) mod3	$e_{(2-F)\text{mod}3}e_{(-F)\text{mod}3}$	$e_{(2-F)\text{mod}3}e_{(-F)\text{mod}3} \ldots e_2$	$e_{(-F)\text{mod}3} \ldots (e_2j)_{TAA}$
6 +(2-F) mod3	$e_{(2-F)\text{mod}3}e_{(-F)\text{mod}3}$	$e_{(2-F)\text{mod}3}e_{(-F)\text{mod}3} \ldots e_2$	$e_{(-F)\text{mod}3} \ldots (e_2j)_{TAG}$
6 +(2-F) mod3	$e_{(2-F)\text{mod}3}e_{(-F)\text{mod}3}$	$e_{(2-F)\text{mod}3}e_{(-F)\text{mod}3} \ldots e_2$	$e_{(-F)\text{mod}3} \ldots (e_2j)_{TGA}$
9	$\hat{e}_0\hat{e}_1$	$\hat{e}_0\hat{e}_1 \ldots \hat{e}_{(F)\text{mod}3}$	$j\hat{e}_0 \ldots \hat{e}_{(F-1)\text{mod}3}$
12	jj	$jj \ldots j$	$j \ldots je_0$
12	jj	$jj \ldots j$	$(\hat{e}_2j)_{TAA}j \ldots j$
12	jj	$jj \ldots j$	$(\hat{e}_2j)_{TAG}j \ldots j$
12	jj	$jj \ldots j$	$(\hat{e}_2j)_{TGA}j \ldots j$

Thus, we have shown by enumeration that both the number of footprint states as well as the number of allowed transitions between footprint states are both limited to linear growth in the footprint size F. (See also **2.11.3 HOHMM Testing Complexity**, page 41.) More precisely, we have the following relations:

# footprint states = 13 + 20(F)
# footprint state transitions = 13 + 20(F+1)

In Table 9 and Table 10 below, we consolidate and repeat the enumeration, first of the footprint states and then the allowed footprint transitions, respectively, with the inclusion of some additional index columns. Each of the tables includes an additional column of indices used in the actual implementation in order to refer to the states and transitions, respectively.

Table 10 also includes yet another column with title “Dom. footprint” for *dominating footprint*, which is used in the actual implementation to denote the type of calculation performed for the corresponding transition probability.

There is one possible source of confusion to be pointed out. The values for dominating footprint shown here have only a *binary* interpretation, indicating whether or not there is more than one allowed footprint transition sharing the common starting footprint state of the given transition. Moreover, the dominating footprint value should not be confused with the *total* number of allowed footprint transitions sharing the common starting footprint, since there are cases where this total as such exceeds the value 2, as in the following:

1. 5 Allowed forward transitions with starting footprint $e_{(2-F)\text{mod}3}\ldots e_1e_2$
    a. $e_{(2-F)\text{mod}3}\ldots e_1e_2 \rightarrow e_{(-F)\text{mod}3}\ldots e_2e_0$
    b. $e_{(2-F)\text{mod}3}\ldots e_1e_2 \rightarrow e_{(-F)\text{mod}3}\ldots e_2i_2$
    c. $e_{(2-F)\text{mod}3}\ldots e_1e_2 \rightarrow e_{(-F)\text{mod}3}\ldots(e_2j)_{TAA}$
    d. $e_{(2-F)\text{mod}3}\ldots e_1e_2 \rightarrow e_{(-F)\text{mod}3}\ldots(e_2j)_{TAG}$
    e. $e_{(2-F)\text{mod}3}\ldots e_1e_2 \rightarrow e_{(-F)\text{mod}3}\ldots(e_2j)_{TGA}$
2. 3 Allowed reverse transitions with starting footprint $\hat{e}_0\hat{e}_1\ldots\hat{e}_{(F)\text{mod}3}$
    a. $\hat{e}_0\hat{e}_1\ldots\hat{e}_{(F)\text{mod}3} \rightarrow \hat{e}_2\hat{e}_0\ldots\hat{e}_{(F-1)\text{mod}3}$
    b. $\hat{e}_0\hat{e}_1\ldots\hat{e}_{(F)\text{mod}3} \rightarrow \hat{\imath}_2\hat{e}_0\ldots\hat{e}_{(F-1)\text{mod}3}$
    c. $\hat{e}_0\hat{e}_1\ldots\hat{e}_{(F)\text{mod}3} \rightarrow j\hat{e}_0\ldots\hat{e}_{(F-1)\text{mod}3}$

Table 9 Enumeration of footprint states

Enumeration of Footprint States				
Dimer Index i	Dimer	Footprint state(f)	Footprint index(es)	Notes
0	i_0i_0	$(i_0i_0)_0$	0	←xx transitions (Dominator index defaults to 1=origin)
1	i_1i_1	$(i_1i_1)_0$	1	
2	i_2i_2	$(i_2i_2)_0$	2	
3	$\hat{i}_0\hat{i}_0$	$(\hat{i}_0\hat{i}_0)_0$	3	
4	$\hat{i}_1\hat{i}_1$	$(\hat{i}_1\hat{i}_1)_0$	4	
5	$\hat{i}_2\hat{i}_2$	$(\hat{i}_2\hat{i}_2)_0$	5	
6	e_0e_1	$(e_0e_1)_0$	6	
7	e_1e_2	$(e_1e_2)_0$	7	
8	e_2e_0	$(e_2e_0)_0$	8	
9	$\hat{e}_0\hat{e}_1$	$(\hat{e}_0\hat{e}_1)_0$	9	
10	$\hat{e}_1\hat{e}_2$	$(\hat{e}_1\hat{e}_2)_0$	10	
11	$\hat{e}_2\hat{e}_0$	$(\hat{e}_2\hat{e}_0)_0$	11	
12	jj	$(jj)_0$	12	
13	e_0i_0	$(e_0i_0)_{[j=0,\ldots,F-1)]}$	13,33,...,13+20(F-1)	←eij transitions (total=20= index offset)
14	e_1i_1	$(e_1i_1)_{[j=0,\ldots,F-1)]}$	14,34,...,14+20(F-1)	
15	e_2i_2	$(e_2i_2)_{[j=0,\ldots,F-1)]}$	15,35,...,15+20(F-1)	
16	$\hat{e}_0\hat{i}_0$	$(\hat{e}_0\hat{i}_0)_{[j=0,\ldots,F-1)]}$	16,36,...,16+20(F-1)	
17	$\hat{e}_1\hat{i}_1$	$(\hat{e}_1\hat{i}_1)_{[j=0,\ldots,F-1)]}$	17,37,...,17+20(F-1)	
18	$\hat{e}_2\hat{i}_2$	$(\hat{e}_2\hat{i}_2)_{[j=0,\ldots,F-1)]}$	18,38,...,18+20(F-1)	
19	i_0e_1	$(i_0e_1)_{[j=0,\ldots,F-1)]}$	19,39,...,19+20(F-1)	
20	i_1e_2	$(i_1e_2)_{[j=0,\ldots,F-1)]}$	20,40,...,20+20(F-1)	
21	i_2e_0	$(i_2e_0)_{[j=0,\ldots,F-1)]}$	21,41,...,21+20(F-1)	
22	$\hat{i}_0\hat{e}_1$	$(\hat{i}_0\hat{e}_1)_{[j=0,\ldots,F-1)]}$	22,42,...,22+20(F-1)	
23	$\hat{i}_1\hat{e}_2$	$(\hat{i}_1\hat{e}_2)_{[j=0,\ldots,F-1)]}$	23,43,...,23+20(F-1)	
24	$\hat{i}_2\hat{e}_0$	$(\hat{i}_2\hat{e}_0)_{[j=0,\ldots,F-1)]}$	24,44,...,24+20(F-1)	
25	e_2j-TAA	$(e_2j\text{-TAA})_{[j=0,\ldots,F-1)]}$	25,45,...,25+20(F-1)	
26	e_2j-TAG	$(e_2j\text{-TAG})_{[j=0,\ldots,F-1)]}$	26,46,...,26+20(F-1)	
27	e_2j-TGA	$(e_2j\text{-TGA})_{[j=0,\ldots,F-1)]}$	27,47,...,27+20(F-1)	
28	$\hat{e}_2j$-TAA	$(\hat{e}_2j\text{-TAA})_{[j=0,\ldots,F-1)]}$	28,48,...,28+20(F-1)	
29	$\hat{e}_2j$-TAG	$(\hat{e}_2j\text{-TAG})_{[j=0,\ldots,F-1)]}$	29,49,...,29+20(F-1)	
30	$\hat{e}_2j$-TGA	$(\hat{e}_2j\text{-TGA})_{[j=0,\ldots,F-1)]}$	30,50,...,30+20(F-1)	
31	je_0	$(je_0)_{[j=0,\ldots,F-1)]}$	31,51,...,31+20(F-1)	
32	$j\hat{e}_0$	$(j\hat{e}_0)_{[j=0,\ldots,F-1)]}$	32,52,...,32+20(F-1)	

Table 10 Enumeration of footprint state transitions

Enumeration of Footprint State Transitions				
Dimer Index i	Footprint state(f)	Footprint transition (σ)	Dom. foot print {1, 2}	σ Index
0	$(i_0 i_0)_0$	$(i_0 i_0)_0 \rightarrow (i_0 i_0)_0$	2	0
1	$(i_1 i_1)_0$	$(i_1 i_1)_0 \rightarrow (i_1 i_1)_0$	2	1
2	$(i_2 i_2)_0$	$(i_2 i_2)_0 \rightarrow (i_2 i_2)_0$	2	2
3	$(\hat{\imath}_0 \hat{\imath}_0)_0$	$(\hat{\imath}_0 \hat{\imath}_0)_0 \rightarrow (\hat{\imath}_0 \hat{\imath}_0)_0$	2	3
4	$(\hat{\imath}_1 \hat{\imath}_1)_0$	$(\hat{\imath}_1 \hat{\imath}_1)_0 \rightarrow (\hat{\imath}_1 \hat{\imath}_1)_0$	2	4
5	$(\hat{\imath}_2 \hat{\imath}_2)_0$	$(\hat{\imath}_2 \hat{\imath}_2)_0 \rightarrow (\hat{\imath}_2 \hat{\imath}_2)_0$	2	5
6	$(e_0 e_1)_0$	$\{i \rightarrow [6+(i+1)\%3]\}$	2	6
7	$(e_1 e_2)_0$		2	7
8	$(e_2 e_0)_0$		2	8
9	$(\hat{e}_0 \hat{e}_1)_0$	$\{i \rightarrow [9+(i+2)\%3]\}$	2	9
10	$(\hat{e}_1 \hat{e}_2)_0$		2	10
11	$(\hat{e}_2 \hat{e}_0)_0$		2	11
12	$(j j)_0$	$(j j)_0$	2	12
13	$(e_0 i_0)_j$	$j=0$: $i \rightarrow [(i-1)\%3]\}$ $j\neq0$: $i \rightarrow i$, $j \rightarrow (j-1)$	1	13,33,...,13+20(F-1)
14	$(e_1 i_1)_j$		1	14,34,...,14+20(F-1)
15	$(e_2 i_2)_j$		1	15,35,...,15+20(F-1)
16	$(\hat{e}_0 \hat{\imath}_0)_j$	$j=0$: $i \rightarrow [9+(i+1)\%3]\}$ $j\neq0$: like above	1	16,36,...,16+20(F-1)
17	$(\hat{e}_1 \hat{\imath}_1)_j$		1	17,37,...,17+20(F-1)
18	$(\hat{e}_2 \hat{\imath}_2)_j$		1	18,38,...,18+20(F-1)
19	$(i_0 e_1)_j$	$j=0$: $i \rightarrow [6+i\%3]\}$ $j\neq0$: like above	1	19,39,...,19+20(F-1)
20	$(i_1 e_2)_j$		1	20,40,...,20+20(F-1)
21	$(i_2 e_0)_j$		1	21,41,...,21+20(F-1)
22	$(\hat{\imath}_0 \hat{e}_1)_j$	$j=0$: $i \rightarrow [3+(i-1)\%3]\}$ $j\neq0$: like above	1	22,42,...,22+20(F-1)
23	$(\hat{\imath}_1 \hat{e}_2)_j$		1	23,43,...,23+20(F-1)
24	$(\hat{\imath}_2 \hat{e}_0)_j$		1	24,44,...,24+20(F-1)
25	$(e_2 j\text{-TAA})_j$	$j=0$: $i \rightarrow 12$ $j\neq0$: like above	1	25,45,...,25+20(F-1)
26	$(e_2 j\text{-TAG})_j$		1	26,46,...,26+20(F-1)
27	$(e_2 j\text{-TGA})_j$		1	27,47,...,27+20(F-1)

Enumeration of Footprint State Transitions, cont.				
Dimer Index i	Footprint state(f)	Footprint transition (σ)	Dom. foot print {1, 2}	σ Index
28	$(\hat{e}_2 j\text{-TAA})_j$	j=0: i → 10 j≠0: like above	1	28,48,...,28+20(F-1)
29	$(\hat{e}_2 j\text{-TAG})_j$		1	29,49,...,29+20(F-1)
30	$(\hat{e}_2 j\text{-TGA})_j$		1	30,50,...,30+20(F-1)
31	$(j e_0)_j$	j=0: i → 6; j≠0: like above	1	31,51,...,31+20(F-1)
32	$(j \hat{e}_0)_j$	j=0: i → 12; j≠0: like above	1	32,52,...,32+20(F-1)
0		$(i_0 i_0)_0 \rightarrow (i_0 e_1)_{(F-1)}$ $i \rightarrow [19 + i\%3 + 20(F-1)]$	2	33+20(F-1) (=#f.p.)
1			2	33+20(F-1)+1
2			2	33+20(F-1)+2
3		$(\hat{i}_0 \hat{i}_0)_0 \rightarrow (\hat{e}_0 \hat{i}_0)_{(F-1)}$ $i \rightarrow [16 + i\%3 + 20(F-1)]$	2	33+20(F-1)+3
4			2	33+20(F-1)+4
5			2	33+20(F-1)+5
6		$(e_0 e_1)_0 \rightarrow (e_0 i_0)_{(F-1)}$, for F%3 = 0 $(e_1 i_1)_{(F-1)}$, for F%3 = 1 $(e_2 i_2)_{(F-1)}$, for F%3 = 2 $i \rightarrow [13 + (F+i)\%3 + 20(F-1)]$	2	33+20(F-1)+6
7			2	33+20(F-1)+7
8			2	33+20(F-1)+8
9		$(\hat{e}_0 \hat{e}_1)_0 \rightarrow (\hat{i}_0 \hat{e}_1)_{(F-1)}$, for F%3 = 0 $(\hat{i}_1 \hat{e}_2)_{(F-1)}$, for F%3 = 1 $(\hat{i}_2 \hat{e}_0)_{(F-1)}$, for F%3 = 2 $i \rightarrow [22 + (2F+i)\%3 + 20(F-1)]$	2	33+20(F-1)+9
10			2	33+20(F-1)+10
11			2	33+20(F-1)+11
8-(F%3)		$(e_{(-F-1)\%3} e_{-F\%3})_0 \rightarrow (e_2 j\text{-TAA})_{F-1}$	2	33+20(F-1)+12
8-(F%3)		$(e_{(-F-1)\%3} e_{-F\%3})_0 \rightarrow (e_2 j\text{-TAG})_{F-1}$	2	33+20(F-1)+13
8-(F%3)		$(e_{(-F-1)\%3} e_{-F\%3})_0 \rightarrow (e_2 j\text{-TGA})_{F-1}$	2	33+20(F-1)+14
11-(F%3)		$(\hat{e}_{(F-1)\%3} \hat{e}_{F\%3})_0 \rightarrow (j \hat{e}_0)_{[\,j=0,\ldots,F-1)]}$	2	33+20(F-1)+15
12		$(j j)_0 \rightarrow (j e_0)_{F-1}$	2	33+20(F-1)+16
12		$(j j)_0 \rightarrow (\hat{e}_2 j\text{-TAA})_{F-1}$	2	33+20(F-1)+17
12		$(j j)_0 \rightarrow (\hat{e}_2 j\text{-TAG})_{F-1}$	2	33+20(F-1)+18
12		$(j j)_0 \rightarrow (\hat{e}_2 j\text{-TGA})_{F-1}$	2	33+20(F-1)+19

2.4.4 Complexity of Extended States Without the Minimum Length Assumption

We consider the complexity of a higher order HMM without the minimum length assumption. In such a model, we still have the fundamental set of 33 dimers as explained in **2.4.1 Enumeration of the 33 Dimer States** above. Furthermore, we observe that for every primitive state, there are at least two dimer states for which the given primitive occurs as the first primitive of a dimer state. These two observations provide lower bounds for both the number of states and the number of transitions for the case of higher order HMM without the minimum length assumption as

# extended states without minimum length assumption $\geq 33 * 2^{F-1}$
# extended state transitions without minimum length assumption $\geq 33 * 2^{F}$

For purposes of theoretical comparison, we assume no additional overhead in the prediction testing process other than the computation of the transition probabilities for the extended states. The graphs in Figure 3 and Figure 4 show in linear and log scale, respectively, the comparison of the theoretical relative time complexity for the extended states and extended state transitions *without* the minimum length assumption to that of the HOHMM *with* the minimum length assumption. Figure 3 with its linear scale clearly shows the linear behavior of the HOHMM used here, whereas Figure 4 with its log scale is included to indicate the exponential behavior of complexity without the minimum length assumption. (See also **2.11 Algorithmic Complexity**, page 40.)

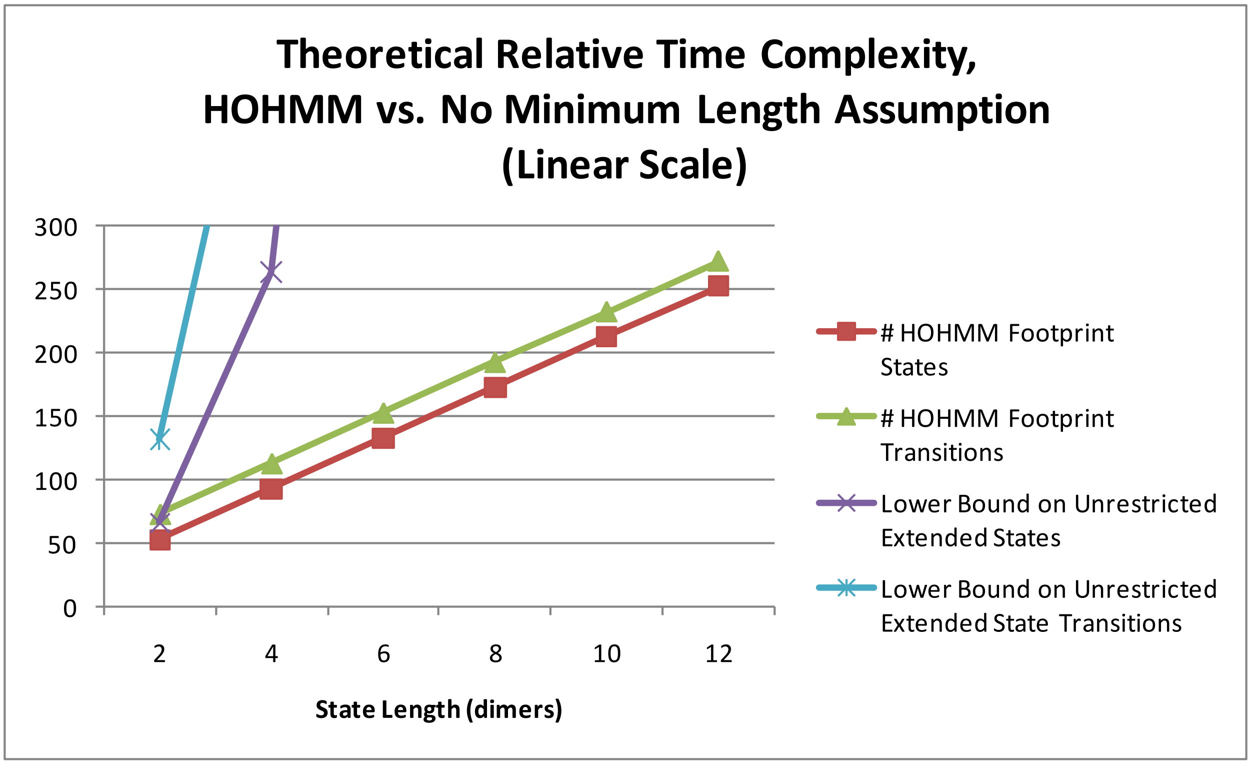


Figure 3 Comparison of theoretical relative time complexity – linear scale

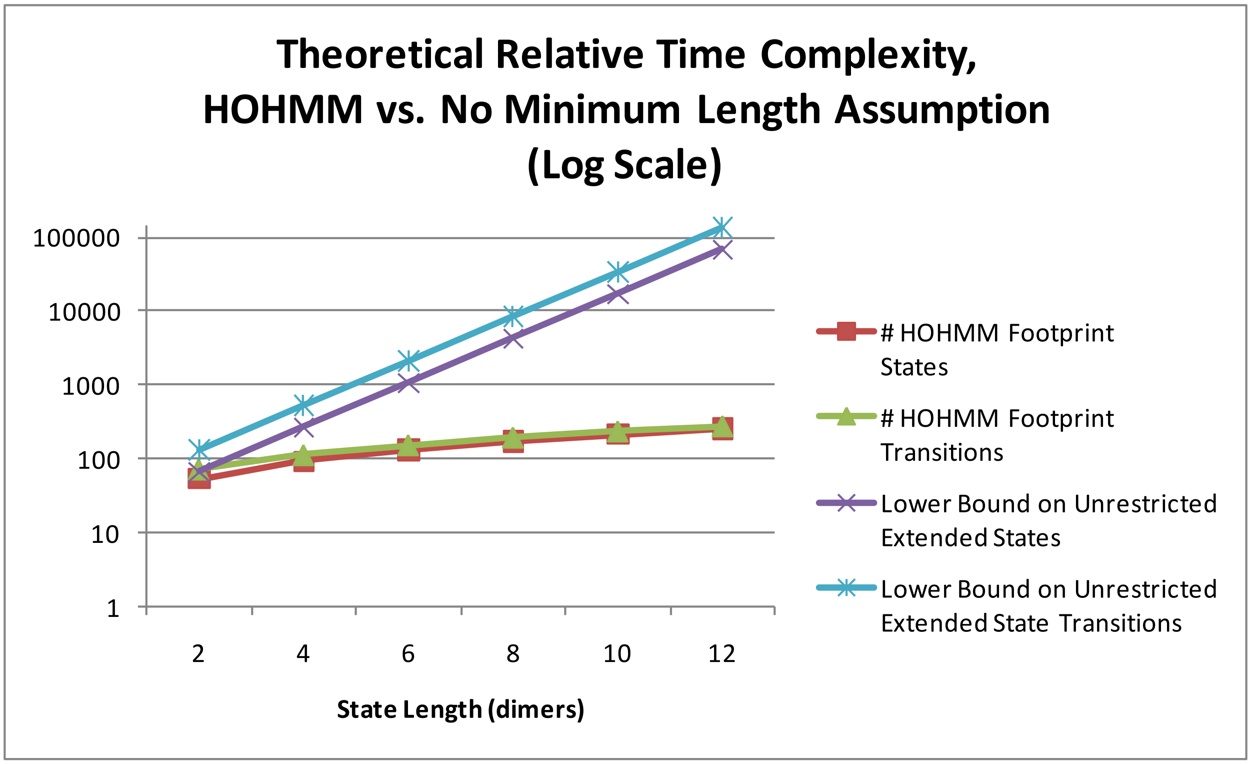


Figure 4 Comparison of theoretical relative time complexity – log scale

2.5 Factorization 1 of Total Probability

Starting from the expression for total probability for the observations and states introduced in **2.3 Higher Order HMM (HOHMM) Specifications**, page 10, we now factor the total probability using the chain rule such that the observation model and the state model remain coupled – contrary to the traditional HMM factorization – as follows:

$$P(B, \Lambda) = P(b_0 , \ldots, b_{n+L+R-2} ; \lambda_{-R -l+1}, \ldots, \lambda_{n+L+r-1}) = \qquad \text{(chain rule)}$$

$$P(b_0 ; \lambda_{-R- \ell+1}, \ldots, \lambda_{-R+r+1})$$
$$P(b_1, \lambda_{-R+r+2} \mid b_0 ; \lambda_{-R- \ell+1}, \ldots, \lambda_{-R+r+1})$$
$$P(b_2, \lambda_{-R+r+3} \mid b_0\, b_1 ; \lambda_{-R- \ell+1}, \ldots, \lambda_{-R+r+2})$$
$$\ldots$$
$$P(b_{i+R}, \lambda_{i+r+1} \mid b_0, \ldots, b_{i+R-1} ; \lambda_{-R- \ell+1}, \ldots, \lambda_{i+r})$$
$$\ldots$$
$$P(b_{n+L+R-2}, \lambda_{n+L+r-1} \mid b_0, \ldots, b_{n+L+R-3} ; \lambda_{-R- \ell+1}, \ldots, \lambda_{n+L+r-2}),$$

Referring back to the illustration of extended observations and states in Figure 2, page 12, we then introduce the assumption that the conditional dependencies on both the observations and the states for each of the above factors are limited accordingly.

$$P(B, \Lambda) = \qquad \text{(assumption of limited dependency of both b's and } \lambda\text{'s on previous b's and } \lambda\text{'s)}$$

$P(b_0 ; \lambda_{-R-\ell+1}, \ldots, \lambda_{-R+r+1})$

$P(b_1, \lambda_{-R+r+2} \mid b_0 ; \lambda_{-R-\ell+1}, \ldots, \lambda_{-R+r+1})$

$P(b_2, \lambda_{-R+r+3} \mid b_0, b_1 ; \lambda_{-R-\ell+2}, \ldots, \lambda_{-R+r+2})$

$\ldots$

$P(b_{i+R}, \lambda_{i+r+1} \mid b_{i-L+1} , \ldots, b_{i+R-1} ; \lambda_{i-\ell}, \ldots, \lambda_{i+r})$

$\ldots$

$P(b_{n+L+R-2}, \lambda_{n+L+r-1} \mid b_{n-1}, \ldots, b_{n+L+R-3} ; \lambda_{n+L-\ell-1}, \ldots, \lambda_{n+L+r-2})$

$=$ (substituting f's for λ's)

$P(b_0 ; f_{-R})$

$P(b_1, f_{-R+1} \mid b_0 ; f_{-R})$

$P(b_2, f_{-R+2} \mid b_0, b_1 ; f_{-R+1})$

$\ldots$

$P(b_{i+R}, f_i \mid b_{i-L+1} , \ldots, b_{i+R-1} ; f_{i-1})$

$\ldots$

$P(b_{n+L+R-2}, f_{n+L-2} \mid b_{n-1}, \ldots, b_{n+L+R-3} ; f_{n+L-3})$

$=$ (substituting $\hat{w}$'s – suitably truncated at the boundaries – for conditional b's)

$P(b_0, f_{-R})$

$P(b_1, f_{-R+1} \mid \hat{w}_{-R+1} ; f_{-R})$

$P(b_2, f_{-R+2} \mid \hat{w}_{-R+2} ; f_{-R+1})$

$\ldots$

$P(b_{i+R}, f_i \mid \hat{w}_i, f_{i-1})$

$\ldots$

$P(b_{n+L+R-2}, f_{n+L-2} \mid \hat{w}_{n+L-2} ; f_{n+L-3})$

$=$ (using $\mathbf{\Pi}$ to denote iterated product)

$P(b_0, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(b_{i+R}, f_i \mid \hat{w}_i, f_{i-1})] \}$

$=$ (definition of conditional probability)

$P(b_0, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(b_{i+R}, f_{i-1}, f_i) / P(\hat{w}_i, f_{i-1})] \}$

$=$ (chain rule)

$P(b_0, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(b_{i+R}|f_{i-1}, f_i) \; P(f_{i-1}, f_i) / P(\hat{w}_i, f_{i-1})] \}$

We observe at this point that the numerator can be expressed as a product of observation and state probability estimates tabulated from training data. As for the denominator, we demonstrate how the denominator can also be expressed in terms of those same tabulated estimates in the next section.

The work to date involves the additional step of replacing b's in the numerator with w's, since the additional b's are all incorporated in the symbol $\hat{w}$ appearing in the conditionals.

$P(b_{i+R}, f_i|\hat{w}_i, f_{i-1}) = P(w_i, f_i|\hat{w}_i, f_{i-1})$ $\qquad$ $(w_i = \hat{w}_i , b_{i+R} = b_{i-L+1}, ..., b_{i+R})$

With the above substitution in the numerator, the previous total probability becomes

$P(b_0, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(b_{i+R}, f_i \mid \hat{w}_i, f_{i-1})] \} =$ (substituting w – suitably truncated at the boundaries – for b)

$P(w_{-R}, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(w_i, f_i \mid \hat{w}_i, f_{i-1})] \}$

= (definition of conditional probability)

$P(w_{-R}, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(w_i, f_{i-1}, f_i) / P(\hat{w}_i, f_{i-1})] \}$

We now apply the state dependent Markov order function, $M(f_{i-1}, f_i, j)$, defined above. We proceed to factor the numerator in the i^{th} term of the above product and simply note that the denominator may be handled similarly. Hence for the numerator we write

$P(w_i, f_{i-1}, f_i) =$
(intermediate result from above & for values of i such that $i \geq L - 1$)

$P(b_{i-L+1}, \ldots, b_{i+R}, f_{i-1}, f_i)$
= (chain rule & for j, $-L+2 \leq j \leq R$)
$P(b_{i-L+1}, f_{i-1}, f_i)$
...
$P(b_{i+j} | b_{i-L+1}, \ldots, b_{i+j-2}, b_{i+j-1}, f_{i-1}, f_i)$
...
$P(b_{i+R} | b_{i-L+1}, \ldots, b_{i+R-1}, f_{i-1}, f_i)$
= (Markov order $M(f_{i-1}, f_i, j)$ defined above)
$P(b_{i-L+1}, f_{i-1}, f_i)$
...
$P(b_{i+j} | b_{i+j-M(f_{i-1}, f_i)}, \ldots, b_{i+j-2},, b_{i+j-1}, f_{i-1}, f_i)$
...
$P(b_{i+R} | b_{i+R-M(f_{i-1}, f_i)}, \ldots, b_{i+R-1}, f_{i-1}, f_i)$

2.5.1.1 Calculations for Factorization 1

Recall the iterated product for the total probability

$P(b_0, \ldots, b_{n-1}, \lambda_{-R-\ell+1}, \ldots, \lambda_{n-R+r}) =$ (substituting f and w from above)

$P(w_{-R}, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(w_i, f_{i-1}, \lambda_{i+r+1}) / P(\hat{w}_i, f_{i-1})] \}$

In order to evaluate the iterated factor above, we now consider the two different cases – as determined by the two possible types of values, eij-type and xx-type, of the footprint state f_{i-1}. We observe in the following that for both cases, the iterated factor can be expressed in terms of probabilities that can be tabulated and estimated from training data.

We may express the numerator of the iterated factor above in terms of an indicator function, D (an indicator function referred to as the "dominator" function [**24**]), indicating the footprint type for f_{i-1}

$P(w_i, f_{i-1}, \lambda_{i+r+1}) =$ (substituting for λ_{i+r+1})

$P(w_i, f_{i-1}, f_i)$

$=$ (splitting f_{i-1} into xx- and eij-types)

$P(w_i, f_{i-1}\text{=eij-type}, f_i)D(f_{i-1}) + P(w_i, f_{i-1}\text{=xx-type}, f_i)(1-D(f_{i-1}))$

where

$$D(f_{i-1}) = \begin{cases} 1, & \text{for } f_{i-1}\text{=eij-type} \\ 0, & \text{otherwise, or } f_{i-1}\text{=xx-type} \end{cases}$$

2.5.1.1.1 Case 1 for Factorization 1: f_{i-1} Footprint State Type = eij

An eij-type footprint state by definition contains a heterogeneous or eij-type dimer, such as the eij-type footprint state $e_0i_0\ldots i_0$ containing the eij-type dimer e_0i_0. In such cases when the footprint state f_{i-1} is an eij-type footprint state, it follows from the minimum length assumption that there is only one allowed target footprint state. For example, the transition

$e_0i_0\ldots i_0 \rightarrow i_0i_0\ldots i_0$

is allowed but not

$e_0i_0\ldots i_0 \rightarrow i_0\ldots i_0e_1$.

Otherwise this would violate the assumption of minimum length for introns. Hence, the subsequent primitive state, λ_{i+r+1}, is completely determined with certainty. Thus we write

$$P(\lambda_{i+r+1}|\ldots, f_{i-1}\text{=eij-type}) = P(\lambda_{i+r+1}|, \ldots, (\lambda_{i-\ell}, \ldots, \lambda_{i+r})\text{=eij-type}) = 1.$$

It follows then that for such cases, the factor in the iterated product above can be simplified.

$P(w_i, f_{i-1}\text{=eij-type}, \lambda_{i+r+1}) / P(\hat{w}_i, f_{i-1}\text{=eij-type}) =$

(definition of conditional probability)

$P(b_{i+R}, \lambda_{i+r+1}|b_{i-L+1}\ldots b_{i+R-1}, f_{i-1}\text{=eij-type})$

$=$ (chain rule)

$P(\lambda_{i+r+1}|b_{i-L+1}\ldots b_{i+R}, f_{i-1}\text{=eij-type})\ P(b_{i+R}|b_{i-L+1}\ldots b_{i+R-1}, f_{i-1}\text{=eij-type})$

$=$ (since $P(\lambda_{i+r+1}|, \ldots, f_{i-1}\text{=eij-type}) = 1$)

$P(b_{i+R}|b_{i-L+1}\ldots b_{i+R-1}, f_{i-1}\text{=eij-type})$

$=$ (substituting f and $\hat{w}$ from above)

$P(b_{i+R}|\hat{w}_i, f_{i-1}\text{=eij-type})$

We observe that such values can be estimated from training data and stored in tables for use during the Viterbi algorithm.

It is worth noting here that whereas this simplification represents a substantial computational savings, it cannot be used in any implementation where a vector-based discriminant, such as an SVM, is desired as there is only one factor, hence only one component, to compute. However, this particular state context is intended to model a coding/non-coding (c/nc) transition that has *already* been encountered recently in the Viterbi algorithm and hence, by the minimum length assumption, there is no competition among the allowed transitions for the occurrence of yet another c/nc transition.

2.5.1.1.2 Case 2 for Factorization 1: f_{i-1} Footprint State Type = xx

Recall that xx-type footprint states are those that do not contain any of the eij-type dimers, such as $\mathtt{e}_0\mathtt{i}_0$. Hence, each xx-type footprint state can be generated by a single primitive state, such as $\mathtt{i}_0\mathtt{i}_0 \ldots \mathtt{i}_0$. In such cases, we do not attempt any simplification as was done in Case 1. Nonetheless, we would like to express the iterated factor appearing in the iterated product above in terms of probabilities that can be tabulated and estimated from training data.

We pause here in order to show by contradiction that no further reduction is possible for evaluating the *joint probability* appearing in the numerator of the iterated product above:

$$P(w_i, f_{i-1}, f_i) \qquad \text{for values of } f_{i-1} \text{ such that } f_{i-1} = \text{xx-type footprint.}$$

Recalling the splitting of probability into the two terms – one for each of eij-type and xx-type – from above, the claim can be stated as

$$P(w_i, f_{i-1}\text{=xx-type}, f_i) = \qquad \text{(definition of function } D(f_{i-1}\text{=xx-type})\text{=0)}$$

$$P(w_i, f_{i-1}\text{=xx-type})D(f_{i-1}\text{=xx-type}) + P(w_i, f_i)(1\text{-}D(f_{i-1}\text{=xx-type}))$$

$$\neq \qquad \text{(claim)}$$

$$P(w_i, f_i)$$

We assume the contrary and arrive at a contradiction by counterexample in the following. We first apply the chain rule to factor the original expression in the numerator.

$$P(w_i, f_{i-1}\text{=xx-type}, f_i) = \qquad \text{(chain rule)}$$

$$P(f_{i-1}\text{=xx-type} \mid w_i, f_i)\, P(w_i, f_i)$$

We can then infer from the above that the first factor would be identically equal to unity.

$$P(f_{i-1}\text{=xx-type} \mid w_i, f_i) = 1.$$

However, we can now demonstrate that the above identity does not hold in general due to the following counterexample. We consider the case for the specific footprint state

$$f_i = \mathtt{i}_0\mathtt{i}_0 \ldots \mathtt{i}_0$$

It follows from the definition of the allowed transitions that this footprint state can only be the result of precisely two possibilities for the preceding footprint state f_{i-1}, namely

$$f_{i-1}\text{=}\ \mathtt{i}_0\mathtt{i}_0 \ldots \mathtt{i}_0 \ \text{ or } f_{i-1}\text{=}\ \mathtt{e}_0\mathtt{i}_0 \ldots \mathtt{i}_0$$

Hence, it follows that

$$P(f_{i-1}=i_0i_0...i_0|w_i,\ f_i=i_0i_0...i_0) = p \qquad \text{and}$$
$$P(f_{i-1}=e_0i_0...i_0|w_i,\ f_i=i_0i_0...i_0)= 1\text{-}p \qquad \text{for some p, } 0 \leq p \leq 1.$$

Now, since the training data are assumed to be eukaryotic and thus to contain instances of observations, w_i, for which both values of the footprint state $f_{i-1} = i_0i_0...i_0$ and $e_0i_0...i_0$ are attained, it then follows that p must be between – *but substantially different from either one of* – 0 and 1.

$$0 << p << 1$$

However, this contradicts the identity that

$$P(f_{i-1}=\text{xx-type} \mid w_i,\ f_i) = 1$$

as inferred from above and hence proves the original claim of irreducibility for the case of $f_{i-1} =$ xx-type. Rather we generally have

$$P(w_i,\ f_{i-1}=\text{xx-type},\ f_i)(1\text{-}D(f_{i-1}=\text{xx-type})) = \qquad (\text{chain rule, and } D(f_{i-1}=\text{xx-type})=0)$$
$$P(w_i|f_{i-1}=\text{xx-type},\ f_i)\ P(f_{i-1}=\text{xx-type} \mid f_i)\ P(f_i)$$
$$= \qquad (P(w_i|f_{i-1}=\text{xx-type},\ f_i)= P(w_i|f_i))$$
$$P(w_i|f_i)\ P(f_{i-1}=\text{xx-type} \mid f_i)\ P(f_i)$$
$$= \qquad (\text{chain rule alternative })$$
$$P(w_i|f_{i-1}=\text{xx-type},\ f_i)\ P(f_i|f_{i-1}=\text{xx-type})\ P(f_{i-1}=\text{xx-type})$$

Note that the first chain rule option above indicates that the conditional probability is essentially determined by f_i and not f_{i-1} .

It is worth noting here that we have observed from earlier trials of HOHMM/Viterbi predictions that the inclusion of the state transition probability factor in the above result is essential for prediction performance and that it's omission can have substantial negative impact on the accuracy of the resulting Viterbi predictions.

We now proceed further with the calculation in the case f_{i-1}=xx-type.

$$P(w_i,\ f_{i-1},\ \lambda_{i+r+1}) / P(\hat{w}_i,\ f_{i-1}) = \qquad (\text{definition of marginal probability})$$

$$P(w_i,\ f_{i-1},\ \lambda_{i+r+1}) / \boldsymbol{\Sigma}_{(\lambda'\text{ allowed})}\, P(\hat{w}_i,\ f_{i-1},\ \lambda_{i+r+1}=\lambda')$$
$$= \qquad (\text{chain rule})$$
$$P(w_i|f_{i-1},\ \lambda_{i+r+1})\ P(\lambda_{i+r+1}|f_{i-1})\ P(f_{i-1}) /$$
$$\boldsymbol{\Sigma}_{(\lambda'\text{ allowed})}\,[P(\hat{w}_i,|f_{i-1},\ \lambda_{i+r+1}=\lambda')\ P(\lambda_{i+r+1}=\lambda'|f_{i-1})\ P(f_{i-1})]$$

$$= \qquad (\text{simplifying by canceling } P(f_{i-1}))$$

$$P(w_i|f_{i-1},\ \lambda_{i+r+1})\ P(\lambda_{i+r+1}|f_{i-1}) / \boldsymbol{\Sigma}_{(\lambda'\text{ allowed})}\,[P(\hat{w}_i,|f_{i-1},\ \lambda_{i+r+1}=\lambda')\ P(\lambda_{i+r+1}=\lambda'|f_{i-1})]$$

$$= \qquad (\text{substituting } P(f_i|f_{i-1}) \text{ for } P(\lambda_{i+r+1}|f_{i-1})$$

and $P(w_i|f_i)$ for $P(w_i|f_{i-1}, \lambda_{i+r+1})$, etc.)

$$P(w_i|f_i)\ P(f_i|f_{i-1})\ /\ \mathbf{\Sigma}_{(f'\ \text{allowed})}\ [P(\hat{w}_i,|f_i{=}f')\ P(f_i{=}f'|f_{i-1})]$$

We observe the similarity between the expression in the numerator and any of the terms of the sum in the denominator. Also, we note that all of the emission and transition factors

$$P(\lambda_{i+r+1}|f_{i-1}),\ P(\lambda_{i+r+1}{=}\lambda'|f_{i-1})$$

occurring in both the numerator and denominator can be tabulated and estimated from training data.

We now factor the conditional probability for the observation w_i in the numerator above over the observations and simply note that a similar factorization can be applied to the various conditional probabilities for the observation $\hat{w}_i$ in the terms in the denominator.

$P(w_i|f_{i-1}, \lambda_{i+r+1}) =$ (substituting for w)

$$P(b_{\max(0,i-L+1)}, \ldots, b_{i+R}|f_{i-1}, \lambda_{i+r+1})$$

$=$ (chain rule)

$$P(b_{i+R}|b_{\max(0,i-L+1)}, \ldots, b_{i+R-1}, f_{i-1}, \lambda_{i+r+1})$$
$$P(b_{i+R-1}|b_{\max(0,i-L+1)}, \ldots, b_{i+R-2}, f_{i-1}, \lambda_{i+r+1})$$
$$\ldots$$
$$P(b_{\max(0,i-L+1)}|f_{i-1}, \lambda_{i+r+1})$$

$=$ (using $\mathbf{\Pi}$ to denote iterated product)

$$\mathbf{\Pi}_{j=\max(0,i-L+1)}^{\quad i+R}\ [P(b_j|b_{\max(0,i-L+1)}, \ldots, b_{j-1}, f_{i-1}, \lambda_{i+r+1})$$

$=$ (for $M = M(f_{i-1}, \lambda_{i+r+1}, j)$ = Markov order)

$$\mathbf{\Pi}_{j=\max(0,i-L+1)}^{\quad i+R}\ [P(b_j|b_{\max(0,\,j-M)}, \ldots, b_{j-1}, f_{i-1}, \lambda_{i+r+1})$$

Here in the last step of the above we have departed further from the traditional HMM by assuming that the observations in this context have a Markov property whose order is dependent on the footprint state transition as well as the position, j, of the observation primitive relative to that transition. Hence, the length of the string of conditional bases in the factors above will vary depending upon the specific footprint state and target state f_{i-1}, λ_{i+r+1}.

For a sample calculation, we let n=4, R=1, L=0, ℓ=r=2, M=1, and we compute the total sequence and state probability as

$P(B, \Lambda) =$

$$P(w_{-R}, f_{-R})\ \{\ \mathbf{\Pi}_{i=-R+1}^{\quad n+L-2}\ [P(w_i, f_{i-1}, f_i)\ /\ P(\hat{w}_i, f_{i-1})]\ \}$$

$=$ ($n = 4$, $R = 1$, $L=0$, ℓ=r=2, $M=1$)

$$P(w_{-1}, f_{-1})\ \{\ \mathbf{\Pi}_{i=0}^{\ 2}\ [P(w_i, f_{i-1}, \lambda_{i+3})\ /\ P(\hat{w}_i, f_{i-1})]\ \}$$

$=$ (expanding $\mathbf{\Pi}$)

$$P(w_{-1}, f_{-1})$$

$[P(w_0, f_0, \lambda_3) / P(\hat{w}_{-1}, f_{-1})]$
$[P(w_1, f_1, \lambda_4) / P(\hat{w}_0, f_0)]$
$[P(w_2, f_2, \lambda_5) / P(\hat{w}_1, f_1)]$
$=$ (substituting b and λ for w and f)
$P(b_0, \lambda_{-2}, \ldots, \lambda_2)$
$[P(b_1, \lambda_{-2}, \ldots, \lambda_3) / P(\lambda_{-2}, \ldots, \lambda_2)]$
$[P(b_2, \lambda_{-1}, \ldots, \lambda_4) / P(\lambda_{-1}, \ldots, \lambda_3)]$
$[P(b_3, \lambda_0, \ldots, \lambda_5) / P(\lambda_0, \ldots, \lambda_4)]$

Also, the result for the trace back in the Viterbi algorithm can be expressed as follows

$$\Lambda^* = \operatorname{argmax}_{\Lambda} P(B, \Lambda) = (\lambda_{-2}^*, \ldots, \lambda_4^*)$$

where

$$(\lambda_{-2}^*, \ldots, \lambda_2^*) = \operatorname{argmax}_{(\alpha,\beta,\chi,\delta,\varepsilon)} [P(b_0, \lambda_{-2}=\alpha, \lambda_{-1}=\beta, \lambda_0=\chi, \lambda_1=\delta, \lambda_2=\varepsilon)]$$

and

$$\lambda_i^* = \operatorname{argmax}_{\lambda'} \{P(b_i, \lambda_{i-3}, \ldots, \lambda_{i+2} = \lambda') / P(\lambda_{i-2}, \ldots, \lambda_{i+1})\} \qquad \text{for } i= 3, 4, 5.$$

2.6 Factorization 2 of Total Probability (Alternate Factorization)

Using the chain rule we now factor the total probability of the observations and states – initially according to the traditional HMM – into the observation model and the state model as follows:

$P(B, \Lambda) =$ (chain rule)
$P(B \mid \Lambda) P(\Lambda)$
$=$ (chain rule on $P(B|\Lambda)$)
$P(b_0 \mid \Lambda)$
$P(b_1 \mid b_0 ; \Lambda)$
$\ldots$
$P(b_{i+R} \mid b_0, \ldots, b_{i+R-1} ; \Lambda)$
$\ldots$
$P(b_{n+L+R-2} \mid b_0, \ldots, b_{n+L+R-3} ; \Lambda) P(\Lambda)$

We recognize in the 2^{nd} step above the departure from the conventional HMM by conditioning the observation model on previous observations. This type of model is not without precedent and has been referred to in the literature as an *autoregressive* HMM [**26**].

First we consider further factorization of the observation model, $P(B|\Lambda)$. Referring back to the illustration of extended observations and states in Figure 2, page 12, we again introduce the assumption that the conditional dependencies on both the observations and the states for each of the above factors are limited accordingly.

$P(B \mid \Lambda) =$ (assumption of limited dependency)
$P(b_0 \mid \lambda_{-R-\ell+1}, \ldots, \lambda_{-R+r+1})$
$P(b_1 \mid b_0 ; \lambda_{-R-\ell+2}, \ldots, \lambda_{-R+r+2})$

$\dots$

$P(b_{i+R} \mid b_{i-L+1} , \dots, b_{i+R-1}, \lambda_{i-\ell+1}, \dots, \lambda_{i+r+1})$

$\dots$

$P(b_{n+L+R-2} \mid b_{n-1}, \dots, b_{n+L+R-1}, \lambda_{n+L-\ell-1}, \dots, \lambda_{n+L+r-1})$

$=$ (substituting f’s for λ’s)

$P(b_0 \mid f_{-R})$

$P(b_1 | b_0 ; f_{-R+1})$

$\dots$

$P(b_{i+R}|b_{i-L+1} , \dots, b_{i+R-1}; f_i)$

$\dots$

$P(b_{n+L+R-2} \mid b_{n-1}, \dots, b_{n+L+R-1} ; f_{n+L-2})$

$=$ (substituting w & ŵ for b’s)

$P(w_{-R}|f_{-R})$

$P(w_{-R+1} \mid \hat{w}_{-R+1} ; f_{-R+1})$

$\dots$

$P(w_i \mid \hat{w}_i ; f_i)$

$\dots$

$P(w_{n+L-2} \mid \hat{w}_{n+L-2} ; f_{n+L-2})$

$=$ (using $\mathbf{\Pi}$ to denote iterated product)

$P(w_{-R}|f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-2} [P(w_i \mid \hat{w}_i ; f_i)] \}$

$=$ (definition of conditional probability)

$P(w_{-R}|f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{n+L-1} [P(w_i, f_i) / P(\hat{w}_i, f_i)] \}$

We observe here that the numerator and denominator of the factors in the iterated product above are similar in form to the result of **2.5 Factorization 1 of Total Probability** and that they may be further factored using the chain rule and the function $M(f_{i-1}, f_i, j)$ similarly applied to the resulting factors.

Now turning to the state model, we first write using the chain rule

$P(\Lambda) =$ (chain rule)

$P(\lambda_{-R-\ell+1})$

$P(\lambda_{-\ell-R+2}|\lambda_{-R-\ell+1})$

$\dots$

$P(\lambda_{i+r+1}|\lambda_{-R-\ell+1}, \dots, \lambda_{i+r})$

$\dots$

$P(\lambda_{n-R+r}|\lambda_{-R-\ell+1}, \dots, \lambda_{n-R+r-1})$

Next we use the simplifying assumption that the state model is of Markov order $\ell + r + 1$, so that we may write

$P(\Lambda) =$ (Markov assumption on states)

$P(\lambda_{-R-\ell+1})$

$$P(\lambda_{-\ell\text{-}R+2} \mid \lambda_{-R-\ell+1})$$

$$\ldots$$

$$P(\lambda_{i+r+1} \mid \lambda_{i-\ell}, \ldots, \lambda_{i+r})$$

$$\ldots$$

$$P(\lambda_{n-R+r} \mid \lambda_{n-R-\ell-1}, \ldots, \lambda_{n-R+r-1})$$

$=$ (substituting f for λ's)

$$P(f_{-R})$$

$$P(f_{-R+1} \mid f_{-R})$$

$$\ldots$$

$$P(f_i \mid f_{i-1})$$

$$\ldots$$

$$P(f_{n+L-1} \mid f_{n+L-2})$$

$=$ (using **Π** to denote iterated product)

$$P(f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{\;n+L-1} [P(f_{-i} \mid f_{-i-1})] \}$$

By combining with the observation model above we have the total probability result

$P(B, \Lambda) =$ (substituting for observation & state models from above)

$$P(b_0 | f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{\;n+L-1} [P(b_{i+R} \mid \hat{w}_i \,;\, f_{-i})] \}$$

$$P(f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{\;n+L-1} [P(f_{-i} \mid f_{i-1})] \}$$

$=$ (combining **Π** expressions)

$$P(b_0, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{\;n+L-1} [P(b_{i+R} \mid \hat{w}_i \,;\, f_{-i}) \, P(f_i \mid f_{i-1})] \}$$

Here again, as in **2.5 Factorization 1 of Total Probability**, we pause to observe that we use this last result and proceed to an implementation with less complexity than in the following based on estimates of the above probabilities tabulated from training data. However, here also we have the motivation to replace b with w in order to allow a vector-based state discriminant.

$P(B, \Lambda) =$ (substituting w for b)

$$P(w_{-R} \mid f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{\;n+L-1} [P(w_i \mid \hat{w}_i \,;\, f_i) \, P(f_i \mid f_{i-1})] \}$$

$=$ (definition of conditional probability)

$$P(w_{-R}, f_{-R}) \{ \mathbf{\Pi}_{i=-R+1}^{\;n+L-1} [P(w_i \,;\, f_i) \, P(f_i \mid f_{i-1}) / P(\hat{w}_i \,;\, f_i)] \}$$

Time did not allow testing using this factorization. Also, this factorization appears to share the same complexity as that of factorization 1, since both require the same number of table lookups per Viterbi table element.

For a sample calculation using the same parameters as that used in **2.5 Factorization 1 of Total Probability**, we let n=4, R=1, L=0, l=r=2, M=1, and we compute the total sequence and state probability as

$P(B, \Lambda) =$

$(n = 4, R = 1, L=0, \ell=r=2, M=1)$

$$P(w_{-1}, f_{-1}) \{ \mathbf{\Pi}_{i=0}^{3} [P(w_i ; f_i) P(f_i \mid f_{i-1}) / P(\hat{w}_i ; f_i)] \}$$

$=$ (expanding $\mathbf{\Pi}$)

$$P(w_{-1}, f_{-1})$$
$$[P(w_0 ; f_0) P(f_0 \mid f_{-1}) / P(\hat{w}_0 ; f_0)]$$
$$[P(w_1 ; f_1) P(f_1 \mid f_0) / P(\hat{w}_1 ; f_1)]$$
$$[P(w_2 ; f_2) P(f_2 \mid f_1) / P(\hat{w}_2 ; f_2)]$$

$=$ (substituting b and λ for w and f)

$$P(b_0, \lambda_{-2}, \ldots, \lambda_2)$$
$$[P(b_1, \lambda_{-1}, \ldots, \lambda_3) P(\lambda_3|\lambda_{-2}, \ldots, \lambda_2) / P(\lambda_{-1}, \ldots, \lambda_3)]$$
$$[P(b_2, \lambda_0, \ldots, \lambda_4) P(\lambda_4|\lambda_{-1}, \ldots, \lambda_3) / P(\lambda_0, \ldots, \lambda_4)]$$
$$[P(b_3, \lambda_1, \ldots, \lambda_5) P(\lambda_5|\lambda_0, \ldots, \lambda_4) / P(\lambda_1, \ldots, \lambda_5)]$$

$=$ (chain rule)

$$P(b_0, \lambda_{-2}, \ldots, \lambda_2)$$
$$[P(b_1 \mid \lambda_{-1}, \ldots, \lambda_3) P(\lambda_3|\lambda_{-2}, \ldots, \lambda_2)$$
$$[P(b_2 \mid \lambda_0, \ldots, \lambda_4) P(\lambda_4|\lambda_{-1}, \ldots, \lambda_3)$$
$$[P(b_3 \mid \lambda_1, \ldots, \lambda_5) P(\lambda_5|\lambda_0, \ldots, \lambda_4)$$

Here again, the result for the trace back in the Viterbi algorithm can be expressed as follows

$$\Lambda^* = \text{argmax}_{\Lambda}\ P(B, \Lambda) = (\lambda_{-2}^*, \ldots, \lambda_4^*)$$

where

$$(\lambda_{-2}^*, \ldots, \lambda_2^*) = \text{argmax}_{(\alpha,\beta,\chi,\delta,\varepsilon)}\ [P(b_0, \lambda_{-2}=\alpha, \lambda_{-1}=\beta, \lambda_0=\chi, \lambda_1=\delta, \lambda_2=\varepsilon)]$$

and

$$\lambda_i^* = \text{argmax}_{\lambda'}\ [P(b_{i-2} \mid \lambda_{i-4}, \ldots, \lambda_i =\lambda') P(\lambda_i =\lambda' \mid \lambda_{i-5}, \ldots, \lambda_{i-1})], \quad \text{for } i=3, 4, 5.$$

2.7 Factor Graph Representation

It has been shown that any higher order HMM can be reduced through a series of steps to a 1st order HMM [**31**]. Hence, we simply assume that for each HOHMM there is an equivalent factor graph representation corresponding to the equivalent 1st order HMM. (See, for example, [**30**].)

2.8 Estimates of Probabilities from Counts

We note that all predictions in this effort are based on state prior, state transition, and emission probabilities which are estimated directly from counts in the training data and without any further refinement. The parameter M in the HOHMM is a key parameter in this regard, indicating whether the resulting estimates for the state prior probabilities have sufficient support from the training data and that the resulting transition and emission estimates are reasonable approximations of the corresponding maximum likelihood estimates (MLE). In such cases, we then assume also that the predicted states of the DNA test sequence are themselves reasonably close to the MLE state predictions.

In Durbin, et al. in [**32**], the authors suggest specific approaches to address the dual concerns of inadequate counts in the training data as well as inadequate approximation to the maximum likelihood estimates for the transition and emission probabilities. For the concern of inadequate data, the authors provide a sample calculation of the posterior mean estimate (PME) using a multinomial distribution for the count likelihood along with a Dirichlet distribution for the prior probability of the multinomial parameters. The resulting PME differs from the MLE for the pure multinomial distribution (the specific count divided by the total count over the training data) by the inclusion of additional parameters effectively serving as pseudocounts. We assume throughout in this effort that counts in the training data are sufficiently large unless indicated otherwise, for example, by an observed reduction in prediction performance due to inappropriate choice for the HOHMM parameter M, the (maximum) base Markov order. (See for example, **3.4.2 Summary of Results for Entire_1_contig of *C. elegans***, page 50.)

Moreover, the issue of inadequate estimates for the state prior probabilities appears to have an impact in only the 1st few columns of the Viterbi algorithm in the HOHMM and is typically not a concern in structural gene prediction where coding regions are not expected to be present at the very beginnings of test sequences. The HOHMM appears to recover from the use of fairly poor estimates for the state prior probabilities (such as a uniform distribution [**24**]) after only a few columns into the Viterbi algorithm. The use of inadequate priors, however, is of much greater importance for prediction of amino acid components of proteins where the prediction at the outset becomes more critical. In such cases, it would be necessary to obtain better estimates of state prior probabilities, etc. using a procedure such as the one cited.

The authors also include a treatment of the Baum-Welch form of expectation maximization (EM) in order to obtain at least a set of local maximum likelihood estimates of the transition and emission probabilities. This iterative calculation would have a substantial impact on training complexity of the HOHMM – not for the transition probabilities where growth is linear in footprint size, F – but for the emission probabilities, where the number of parameters to compute *iteratively* via the Baum-Welch algorithm would be significant. For example, with a training sample window size, W=40, and a maximum base Markov order, M=8, the number of parameters to be estimated would be bounded below as in the following relation:

$$\begin{aligned}&\#\text{ parameters in estimate} \geq 4^{M+1}[N_{xx} + N_{eij}N_p]\\&= (5 + 8*(40\text{-}8))4^{M+1}\\&\approx 6.8 * 10^7\end{aligned}$$

where $N_{xx} = 5$ = # static emission tables (for xx-to-xx-type footprint transitions),
$N_{eij} = 8$ = # position-dependent emission tables (for eij-to-eij-type footprint transitions), and
$N_p = W – M$ = # supportable sample positions for window, W, and base Markov order M

In general, such estimates are beyond the scope of this effort, and the issue of probability estimates is managed with a sufficiently large training data set along with the optimization of other parameters in the HOHMM, such as M, F, L, and R.

2.9 Limitations on Footprint State Size

We observe for sufficiently large footprint state size, F, that the prediction performance eventually becomes degraded. (See for example **3.6.2 Summary of Results for Chromosomes I-V of *C. elegans***, page 61.) The optimum size for F is limited usually by the spatial extent of the anomalous statistics in the region surrounding the coding/non-coding transitions in the DNA sequence data. Moreover, by assumption the base Markov order can be varied as a function of position from the c/nc site in the DNA data. (See **2.3.1 Assumptions**, page 13.) This implies that we are then free to consider only the maximum of the anomalous extents mentioned above in determining the limiting footprint state size, F.

2.10 Alternate Proof of Linear Upper Bound on Footprint States and Transitions

As an additional contribution in this effort, we proceed to show that the growth of the number of allowed transitions among footprint states is linear in the hidden state length F when taken in units of overlapping dimer states, s.

Assumptions:

1) The alphabet, Λ, of primitive hidden states has the *fixed* size N.
2) The allowed alphabet, $A_F \subset \Lambda^{F+1}$, of hidden, footprint states of length F in dimers is a proper subset of (the Cartesian product) Λ^F.

Consider the many-to-one mapping in which all heterogeneous footprints (i.e. of type xy, or eij in our application) by definition are associated with one and only one pair of distinct primitives, x and y. There are (at most in our application) $\binom{N}{2}$ eij-type dimers and each eij-type dimer generates its own set (of size F) of footprints indexed by i=0,1,...,F-1. The splitting factor, K, is introduced to account for the case of the footprint type e_2j being split into the three footprint state types, e_2j_{TAA}, e_2j_{TAG}, and e_2j_{TAG}. Also, there are only N distinct xx-type footprint states.

As a consequence of the above, we observe that an upper bound for $|A_F|$, the size of A_F, is provided by...

$|A_F|$ = # footprint states of size F in dimers
$\leq KF\binom{N}{2} + N$
$= (FK(N-1)/2 + 1)N$ (since $\binom{N}{2} = N(N-1)/2$)

where K = splitting factor defined above,
F = footprint state size in dimers, and
N = # of xx-type footprint states.

We observe that the expression above is clearly linear in F, the length in dimers of the footprint states.

Finally, we observe that (by definition) the set of allowed transitions among footprints states of length F in dimers essentially serves as the new set of footprint states of length F+1 in dimers. Hence, for an upper bound on the number of such transitions we simply substitute F+1 for F in the above relation.

$|A_{F+1}|$ = # footprint states of size F+1 in dimers
= # allowed transitions among footprint states of size F in dimers
$\leq (F+1)K(N-1)/2 + 1)N$

Since the above argument was formulated for fixed but arbitrary F, it follows that the footprint state *transition space* as well as the footprint *state space* are both bounded by linear growth in F. QED.

2.11 Algorithmic Complexity

In this section we compare the algorithmic complexity in both time and memory of the traditional, 1st order HMM with that of the HOHMM. It should be pointed out that the Viterbi algorithm used in the HOHMM testing phase is a special case of *message passing* algorithms, for which the complexity in general depends upon the number of iterations [**30**]. In the following, and as is typically done for the special case of the Viterbi algorithm, we assume that only one iteration is to be performed for both the 1st order HMM and the HOHMM. (For applications involving iterative execution in the testing phase of the HOHMM – or multiple Viterbi passes – see **3.5.3 In-frame Stop Filtering**, page 58 and **4.1 Additional Applications of Iterative HOHMM**, page 68.) Also in the following, in order to specify algorithmic complexity, we use the notation referred to as "big Oh". (See, for example, [**33**].)

2.11.1 HMM Complexity in Comparison

For comparison with the HOHMM, we first consider the complexity of the traditional 1st order HMM. During HMM training, the accumulating and storing of the frequency of occurrence of transitions among the hidden states in the training data set requires time and memory according to the following:

$$C_{\text{HMM, train, time}} = O(T_{\text{train}})$$

and

$$C_{\text{HMM, train, memory}} = O(N^2)$$

where

T_{train} = the length of the training data set,
N = the number of hidden states, and
N^2= the number of transitions, assuming all possible transitions are allowed.

Next, during HMM testing, the Viterbi algorithm is used in order to dynamically determine (and track via back pointer) the most probable previous state (and hence the most probable path of states ending) at each state, while doing so at each point in the testing data set. Thus, the time and memory complexity involved in the Viterbi algorithm is given by

$$C_{\text{HMM, test, time}} = O(T_{\text{test}} N^2)$$

and

$$C_{\text{HMM, test, memory}} = O(T_{\text{test}} N)$$

where

T_{test} = the length of the testing data set and N as above.

In the following section we compare the above results with the algorithmic complexity of the HOHMM.

2.11.2 HOHMM Training Complexity

We examine the complexity of the HOHMM for a genome of DNA nucleotide base size, T; coding density D, incorporating the number of genes as well as some notion of average number of exons and/or introns per gene; and S denoting an average length for both features, exons and introns, as well as for the junk extents between genes. We will also need to designate the (maximum) base Markov order, M; footprint state size (in dimers), F; and both the left and right base emission extents, L and R, respectively. Finally, we will need to designate a sampling window size, W, that is chosen sufficiently large as to accommodate both the L and R base extents along with the maximum base Markov order, M.

For convenience we consider only the theoretical training context, where strings of the maximum base string length, M=8, are counted as the sample window is advanced along the testing data set. (In actual implementation, the samples are first extracted and written to disc for future analysis off line.)

Training consists of counting the frequency of occurrence of substrings in the sample window as that window advances along the test data according to the given dimer type. Recall that there are 2 major dimer types, eij and xx, with each type including 8 and 5 dimer subtypes, respectively. The training process can be itemized as follows.

1) O(8) for eij dimers (pre-base ≠ post-base), 8 subtypes: `e0i0`, `e1i1`, `e2i2`, `i0e1`, `i1e2`, `i2e0`, `e2j`, `je0`.
    a. (O(TD)) for each annotated c/nc transition..
    b. (O(W)) for each position in the sample window(with support permitting according to maximum base Markov order, M)..
    c. (O(4^{M+1})) count the frequency of occurrence of string size M+1 (storage only)
2) O(5) for xx dimers (pre-base = post-base) – 5 subtypes: exon0(`e0e1`), exon1(`e1e2`), exon2(`e2e0`), `ii`, `jj`.
    a. Case C/NC
        i. Coding (exon) (O(TD)) for each annotated c/c transition..
        ii. Non-coding (intron, junk) (O(T(1-D)) for each annotated nc/nc transition..
    b. (O(1)) for center position only in the sample window..
    c. (O(4^{M+1})) count the frequency of occurrence of string size M+1 (storage only)

Thus, the time and memory complexity of the HOHMM training process is given by..

$$C_{\text{HOHMM, train, time}} = O(TDW)$$
$$C_{\text{HOHMM, train, memory}} = O(W4^M)$$

for training set size T, coding density D, sample window size W, and maximum base Markov order M.

2.11.3 HOHMM Testing Complexity

2.11.3.1 HOHMM Testing Time Requirement

Recall that testing consists of the following:

1) Constructing the Viterbi table (considering columns advancing left to right).
    a. Initializing the first column with state prior probabilities
    b. for each observation string..

        i. Computing a new column of greatest path probabilities, and
        ii. Computing a new column of back pointers in order to record a most probable state path – at each row in the column.

2) Retracing the most probable path of states through the table of back pointers as constructed above.

The size of the Viterbi table is determined by

1) length = T + R + F/2 -1
2) width = # footprint transitions = 13 + 20F

The footprint transition types are

1) for Dom=1: 20(F-1)
2) for Dom=2: 13 + 20 = 33]

The time and memory requirements are detailed as follows.

1) O(13 +20F) = O(F) Initialization of allowed transitions.
2) O(T+R+F/2) Construction of Viterbi table for each column.
    a. O(20(F-1)) = O(F) For each eij-to-eij (Dom-1) footprint transition.
        i. Lookup the (log) emission probability of the emitted base located at extent R for the given transition.
        ii. Lookup the (log) probability of the given transition.
    b. O(33) = O(1) For each xx-to-xx or xx-to-eij (Dom=2) footprint transition.
        i. O(1) For each allowed transition originating from the given xx source footprint.
        ii. O(L +R) For each base position in the base emission extent, L+R, lookup the (log) emission probability of the observed, emitted base for the given transition.
3) O(T+R+F/2) Viterbi traceback

Based on the above, the complexity of the HOHMM testing process is written initially and then further simplified as follows.

$$C_{\text{HOHMM, test, time}} = O(F +(T+R+F/2)(F+L+R) + T+R+F/2)$$

$$= O((T+R+F/2)(F+L+R) + T + R) \qquad \text{for } T > F$$

$$= O((T+R+F/2)(F+L+R)) \qquad \text{for } T > F$$

$$= O(T(F+L+R)) \qquad \text{for } T > F\text{, since } R+F/2 < F+L+R$$

In comparison with the HMM, we observe that the role of N^2 in the HMM is now filled by the expression (F + L + R). In particular we point out the linearity in F for fixed L and R, as indicated in the sample set of time trials shown in Figure 5.

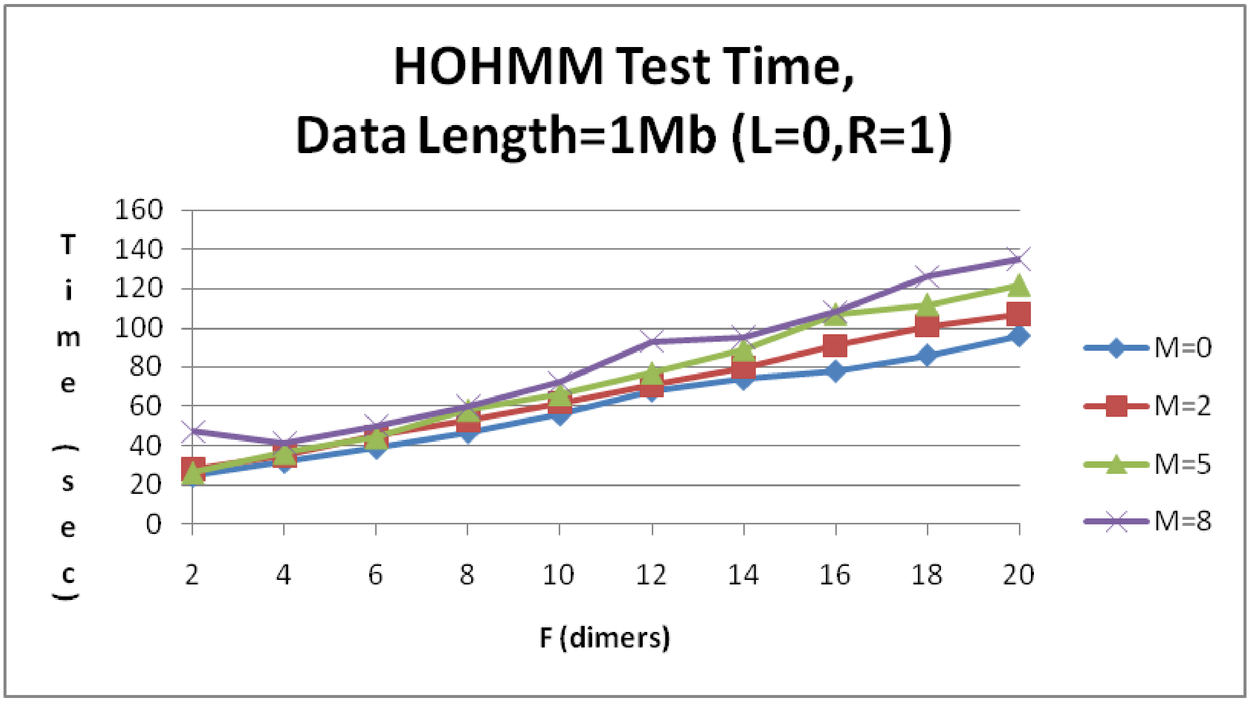


Figure 5 A sample of HOHMM test times for test data length 1Mb

2.11.3.2 HOHMM Testing Memory Requirement

We observe that only two consecutive columns of the Viterbi probability table are required at any step in the Viterbi decoding. Thus the maximum memory requirement in testing the HOHMM is similar to that of training with the exception of an added term for storage of Viterbi traceback information. This additional term scales according to both footprint size and data length.

$$C_{\text{HOHMM, test, memory}} = O(W*4^{M} + FT)$$

for sample window size W, maximum order M, footprint size F, and data length T.

2.12 HOHMM Summary

We have provided two alternative factorizations for the HOHMM including the following:

1) an emission-dependent emission model, where the emission probabilities depend not only on the hidden states as in the traditional HMM but also on other (here neighboring) observations (also referred to as autoregression [**26**]),
2) a higher order state transition model, where the state transitions at the single primitive level of the traditional, 1st order HMM have been replaced by higher order state transitions encoded as extended states.

Also, the complexity of the first factorization was shown to be linear in the footprint state size and confirmed with sample time trials. Finally, from the formal similarity of the two factorizations, the second factorization appears to have the same complexity as the first, though time did not allow confirmation via sample time trials for the second factorization.

3 HOHMM Implementation and Results

3.1 Implementation - Hardware/Software Setup

All (additional) high-level coding for this work was done in Perl – predominantly for the following purposes:

1) data preprocessing
2) distribution and control of computational tasks, and
3) post processing and summarizing of results

(For an overview of the code, see **7.6 Code Design for HOHMM**, etc., page 141.)

3.1.1 Perl and SMP for Genomic Signal Processing

It is worth noting that although Perl has become the de facto choice for most purposes in genomic signal processing, it appears that Perl development per se has lagged substantially in its support of symmetric multiprocessing (SMP) hardware. Using the facilities in the core packages “threads” and “threads::shared”, the application developer, can use multiple Perl system threads at once, but whether such threads are allowed to run simultaneously depends upon the OS-implementation [**34**]. Experience has shown, predominantly in the Linux/Ubuntu environment, that that no more than 2 OS-level processes are allocated at any one time though as many as 8 Perl threads – one per core processor on Sun UltraSparc hardware – have been requested by the application. This leaves any remaining processors in any multicore hardware platform unused by the Perl code and hence idle – contrary to the impression that the programmer might have had that his or her threads were being executed simultaneously.

Admittedly, additional support exists in Perl via add on modules (e.g. “forks”) for true concurrency via native threading (including the extra complexity of managing inter-process shared memory). However, the HOHMM training and testing processes involves extensive sampling and counting tasks, which may take great advantage in the use of shared memory for the accumulation of counts. (Testing tasks are fundamentally sequential as done here, though parallelizable as such but not without the added complexity of splicing together sequential prediction results.) Such processing certainly lends itself to the recent increase in availability of multicore, SMP hardware platforms – rather than just distributed processing. Nonetheless, some simple preliminary tests of the extension modules for native threading have indicated that (the shared memory portion of) run time execution during training per se would have increased by a factor of ~10^6.

Thus the future use of Perl as the de facto choice for higher volume genomic data processing appears to be in question. For that reason, though time did not allow in this effort, further investigation is warranted into future porting/converting of the Perl portion of the HOHMM to another language with perhaps better integration with native-OS threading. Whereas Python, in its core modules, does not appear to use native-OS threads, extensions exist such as the Python module “multiprocessing” and Jython (Python + Java) which may work well but for which time did not allow further investigation in this effort. (See also [**35**].)

3.1.2 HOHMM Application Features

The following are some features worth noting of the training and testing code developed in this effort.

1) Support for a single, Perl-based, hence *fully programmable*, user configuration input file for specifying both training and testing tasks to be run within a given session, where all training tasks must complete before any testing tasks are initiated.

2) Support for remotely concurrent, locally multi-threaded (via Perl threads – see limitations above in **3.1.1 Perl and SMP for Genomic Signal Processing**) processing as specified in the user configuration file on multiple instances of hardware running Unix/Linux with public key-authentication.
3) Support for task submittal via user configuration file (though in separate sessions) in the following environments:
    a. Platforms with direct job scheduling (vis-à-vis see next item) with any such Perl-supported operating system, including MS Windows, Unix, and Linux.
    b. Controlled task scheduling via LoadLeveler on IBM/AIX.
4) A well defined, job-scheduling interface, allowing for the support for any other such environment with special job scheduling, proprietary or otherwise – with the moderate, incremental development of a task scheduling module compliant to that interface.
5) Support for reuse of a single user configuration file across multiple sessions involving different groups of training and/or testing tasks – via the specification by the user of two additional (typically quote-delimited) runtime, input parameters (in addition to the configuration file name) defining lists of regular expressions to be used as search strings in order to select the desired subset of training and testing task names, respectively, as defined in the configuration file.
6) An extensive list of diagnostic options supported in the user configuration file for training and testing, including the following.
    a. (training) global sample output, in-frame stop sample filtering, short sample inclusion policy, relative entropy logging,
    b. (testing) sequence gain factor, maximum count of Viterbi passes for in-frame stop filtering, in-frame stop penalty log-factor, several options for logging in-frame stop filtering, and options for logging values from consecutive columns of the Viterbi probability and backpointer tables.
7) Limited support is provided for protection against denial of service attacks due to multiple submissions of the same job. This is accomplished on the client platform by checking whether a server task process is already in progress with the given runtime parameters on the given server (local or remote).

3.1.3 OS Environments Used

The OS environments used in this effort include the following:

1) 5 Sun UltraSparc, dual quad-core servers each with 8GB of RAM (though not without inconsistencies in available RAM due to variations in policy of memory acquisition of the various graphics driving software)
2) Various IBM/AIX clusters of the LONI (Louisiana Optical Network Initiative [**28**]) network, including predominantly Neptune and Lacumba, where each such cluster includes one front end node and 13 compute nodes, where each node houses 8 processors and 16GB of RAM.

3.1.4 Software Tools Used

The software tools used in this effort include the following:

1) Perl development: Easy Eclipse for LAMP, Version 1.2.2.2 for WindowsXP [**36**]
2) C development: Easy Eclipse for C and C++, Version 1.3.1.1 for WindowsXP [**37**]
3) C compilation: GCC, the GNU Compiler Collection [**38**]

4) 3D Plotting: Gnuplot [**39**]
5) 2D Plotting: Charts in MS Excel 2007 for WindowsXP

3.2 Method of Measurement for Prediction Performance

In general, the measure of prediction performance was taken at the two levels, full exon and individual base (nucleotide) levels, according to the specifications established by Burset and Guigo in [**17**]. In particular, there is one point of departure from the cited method. (See **3.2.3 Expectation of the Measure vs. Measure of the Expectation** below.) The need for both base- and exon-level measurements is based on the existence of 2 types of discernible variations in information content present in the DNA sequence data – 1) over the extent of coding regions compared to non-coding regions and 2) in the vicinity of coding/non-coding (c/nc) boundaries. For reference, the following excerpt is worth noting from the authors' discussion of measurement.

"Evaluation at the exonic structure level provides complementary information about the accuracy of the programs compared to that provided by evaluation at the coding nucleotide level. At the coding nucleotide level we are measuring how well the sequence coding regions are located – the search by content component of the gene structure prediction programs – while at the exonic structure level, we are measuring how well the sequence atomic signals (splice sites and start and stop codons) are identified – the search by signal component of the programs. High accuracy at the coding nucleotide level does not necessarily imply high accuracy at the exonic structure level. For instance, a program can have high sensitivity at the coding nucleotide level, because most of the coding nucleotides have been identified, but very low sensitivity at the exonic structure level, because most splicing signals have been incorrectly predicted."

3.2.1 Accuracy at the Base or Nucleotide Level

The first of the two accuracy measures used in this effort, at the base or nucleotide level, is given by

snsp_avg = (sn + sp)/2,

where
sn = TP/(TP + FN) and
sp = TP/(TP + FP),

such that
TP = true positive count (correctly predicted actual coding)
FP = false positive count (falsely predicted actual non-coding)
FN = false negative count (falsely predicted actual coding)

We note that the authors have used an alternative form of specificity from the usual form

sp=TN/(TN+FP), as explained, for example, in [**40**].

In the context of gene prediction, with typically high concentrations of junk, the contribution from the quantity TN= true negative (or correctly predicted actual non-coding) can overwhelm FP in an otherwise moderately accurate prediction. Thus it seems reasonable to exclude any contribution from TN in order to avoid "very large noninformative values" of the resulting specificity measurements.

For the sake of convenience, their “Approximate Correlation(AC)” for the base level accuracy measure was replaced with the simple average of base level sensitivity and specificity. Moreover, the published results were converted to the simpler measure, the average of sn and sp, for the sake of reference in the test results for this effort as is reported below in **3.5 The Benchmark Data Set ALLSEQ**.

As the authors explain, AC is derived from the measure referred to as “Average Conditional Probability” or ACP, where ACP in general is a combination of 4 ratios and is defined as the average over those ratios of the 4 that are defined.

$$ACP = (1/4) [TP/(TP + FN) + TP/(TP + FP) + TN/(TN + FP) + TN/(TN + FN)]$$

Thus ACP has the dual advantage over all other measures considered that:

1) by convention, it is defined for all values of TP, TN, FP, and FN, and
2) it provides equal weighting for coding and non-coding regions regardless of their relative size.

The quantity AC mentioned above is simply a transformation of ACP to cover the interval [-1, 1].

$$AC = (ACP – 0.5) * 2$$

As such, AC then provides a ready comparison to the traditionally preferred measure, the Pearson product-moment correlation coefficient, which is not defined for all values of TP, TN, FP, and FN. [**17**]

3.2.2 Accuracy at the Full Exon Level

The second of the two accuracy measures, at the full exon level, presents a much greater challenge as it requires the successful prediction of multiple events or positions in the DNA test sequence. These events include the start and end positions of exons as well as the continuation of the exon at all intermediate points. The full exon accuracy is given by

SNSP_AVG = (SN + SP)/2,

where
SN = (number of correct exons)/(number of actual exons) and
SP = (number of correct exons)/(number of predicted exons)

It should be noted that this measure for full exon accuracy does not allow for any improvement due to *partial* exon prediction. More specifically, the exon level accuracy can only be improved by the precise prediction of one or more *entire* exons – at both start and end positions.

3.2.3 Expectation of the Measure vs. Measure of the Expectation

Besides the use of the simpler average (sn + sp)/2 for the base accuracy in this effort in place of the AC measure mentioned above, there is another point of departure from the method of measurement used by the authors Burset and Guigo as discussed in their work [**17**] as well as in [**4**], concerning the handling of multiple sequences. In both of the cited evaluations, the base or nucleotide accuracy for the entire test data set was taken as the average or expectation of the base accuracy measurements of the individual DNA test sequences. This same approach using expectation was taken for the exon accuracy as well. This has the effect of weighting genes with shorter and fewer exons more heavily in the base and exon level accuracy measurements, respectively. Moreover, this effect can become extremely pronounced in cases such as both of the cited evaluations, where all DNA sequences tested contain only a *single gene*.

In contrast in this effort the numbers of correct (and separately incorrect) predictions were first summed over all test sequences and then the measurements were computed from those sums for the exon and base level measurements, respectively. Consider an extreme case, for example, where these two different methods can yield different results as shown in Table 11 below.

Table 11 Comparison of methods for testing performance on multiple sequences

Sequence name	# of exons in sequence (N)	Case 1		Case 2	
		# of true positives (TP)	SN(=TP/N)	# of true positives (TP)	SN(=TP/N)
Sequence A	10	10	1	9	.9
Sequence B	1	0	0	1	1
E(SN)			.5		.95
SN(E)			10/11=.91		10/11=.91

As we see from the examples in the table, the latter method as used here in this effort does not discriminate between the two cases where only one exon was missed as such in the two different contexts. Granted that in the context of the cited evaluations, this method of measurement appears appropriate for the data sets used, where it is known in advance (and may be *leveraged* as such in the design of the program being tested) that one and only one gene is contained in every instance for the test sequences in the test data set. However, in what is a more realistic context of raw genomic data there are extenuating circumstances.

1) Raw genomic sequences can contain unknown amounts of genes.
2) Raw genomic can involve significant local variations in coding density, in which case the detection of motifs signaling the starts and ends of genes becomes a higher priority.
3) In principal, scoring at the exon level in effect designates the *exon* as the fundamental unit being counted rather than the *gene*, whereas according to the method cited more complex genes would be weighted the same as simpler genes with fewer exons.

As indicated above, in each case of the 3 data sets used in this effort, the measurements for both the exon and base level prediction differ somewhat from the method used in the cited evaluations. Moreover, of the 3 data sets tested in this effort, ALLSEQ is the only data set consisting entirely of only single-gene, DNA sequences. Hence, it is the only case in this effort where the cited method of measurement is somewhat more appropriate. Thus, for reference purposes, the results of the HOHMM for ALLSEQ in this effort have been measured precisely according to the cited method – in addition to the method used here and more appropriately so for DNA sequences containing many genes. (See **5.1 Conclusions from HOHMM Results**, page 79.)

3.3 HOHMM Results

For a discussion of method of measurement, see **3.2 Method of Measurement for Prediction Performance**, page 46.

There are 4 main sections of results. The first 3 sections are for reporting on results of the HOHMM per se and the last section shows the results of computations of relative entropy as will be explained. In general, all results for the HOHMM in this effort have been determined using the method of measure of the expectation (of true positives, for example, over all sequences) rather than the expectation of the measure (of sensitivity and/or specificity). (See **3.2.3 Expectation of the Measure vs. Measure of the Expectation**, page 47.) As an exception and for direct comparison with the results cited in [**17**] for the ALLSEQ data set, the results using the cited method of measure have also been included in **5.1 Conclusions from HOHMM Results**, page 79.

The first 3 sections include extensive testing results of the HOHMM for 3 different data sets, respectively, taken in order of increasing size of the data sets. For each of these 3 sets of testing trials, the results include a set of 2D graphs summarizing the best accuracy (where the maximum is taken over parameters L and R) plotted vs. (maximum) base Markov order M and footprint size, F. Moreover, in the case of each data set, the summary plots are accompanied in **7 Appendix** by an extensive set of 3D graphs plotting accuracy for individual values of L and R. In each of the 3D graphs, by the smoothness of the surfaces depicted, these 3D plots serve overall as a demonstration of stability of the HOHMM prediction over variations in all 4 parameters, M, F, L, and R.

In particular, the first 2 of the 3 sections of prediction results are simple, brute force parameter searches with data sets of increasing size, the first ~200kb (entire_1_contig) and the second ~2Mb (ALLSEQ), where the training and testing data sets are identical. For reference purposes in the latter, we also display the published results in [**17**]. (See also **3.2 Method of Measurement for Prediction Performance** above for the discussion on the method used to determine accuracy for a collection of sequences.) The third section shows the results of 5-fold cross validation with the largest data set tested, ~70Mb (Chromosomes I-V of *C. elegans*).

3.4 The Data Set Entire_1_contig for *C. elegans*

3.4.1 Data Preparation

Concerning the term *contig*, in quoting the original paper [**41**] the website [**42**] defines contig as follows.

‘Definition of a contig

In order to make it easier to talk about our data gained by the shotgun method of sequencing we have invented the word "contig". A contig is a set of gel readings that are related to one another by overlap of their sequences. All gel readings belong to one and only one contig, and each contig contains at least one gel reading. The gel readings in a contig can be summed to form a contiguous consensus sequence and the length of this sequence is the length of the contig.’

Moreover, in a BLAST analysis performed on entire_1_contig [**43**], the variety of alignments returned indicates that it is most likely a *training contig*, where evidently the segments of DNA in question were not necessarily overlapping yet have simply been concatenated into a single file.

The properties of the resulting data set are summarized below in Table 12. In particular we note the data size ~200Kb, the smallest but not the least dense of the 3 data sets studied here.

Table 12 Properties of the entire_1_contig data set

# Bases	Coding Density	Sequences			Introns			Exons		
		Total	BP	Avg. Len.	Total	BP	Avg. Len.	Total	BP	Avg. Len.
265018	0.21	64	111375	1740.23	221	55650	251.81	285	55725	195.53

3.4.2 Summary of Results for Entire_1_contig of *C. elegans*

For detailed results for entire_1_contig, including combined 3D surface and contour graphs of both exon- and base-level accuracy for all values of the parameters M, F, L, and R tested, see **7.1 Detailed Results for Entire_1_contig of *C. elegans***, page 88.

As mentioned above, both here and in the case of the 2nd data set tested, ALLSEQ, the training and testing data sets are identical. Thus we consider these results as *ideal* only and as a (brute force) parameter search rather than a valid test of prediction accuracy. Time did not allow to complete the parameter search with this data set for a local maximum in the exon nor in the base level predictions. (See, however, **3.6.2 Summary of Results for Chromosomes I-V of *C. elegans***, page 61.)

Figure 6 and Figure 7 below are graphs of exon- and base-level maxima, respectively, of HOHMM's prediction performance, where each plotted value represents a maximum taken independently of the other plotted values over the parameters L and R. These graphs illustrate the enhanced performance of HOHMM, e.g. M5 (nucleotide=.99, exon=.99) over simpler HMM's, e.g. M0 (.81, .63) using only the intrinsic information in the data – excluding any extrinsic information, such as EST's, protein homology, etc.

In comparing the results of this data set to the other results in this effort, the quality of the best result (SNSP average = 99.6% for M=5, F=12) can be attributed to the small size of the training data set as well as adherence among the donor and acceptor splice sites to the consensus sequences, gt and ag, respectively. Moreover, the reduced performance at full exon level for M=8 compared to that for M=5 is an additional indication of insufficient training size reflected in lack of support for M=8 probability estimates at splice sites. (Time did not allow the computation and inclusion of relative entropy values for entire_1_contig. See **3.7 Relative Entropy of C/NC Transitions**, page 65.)

For raw genomic data, preconditioning would include a removal of the majority of the junk state portion of the training set prior to training, such as extraction of open reading frames (ORF's) or ORF filtering. (See also [**44**]). For an HOHMM-based approach to ORF filtering, see **4.1.1 Sequence-gain Preprocessing**, page 68.

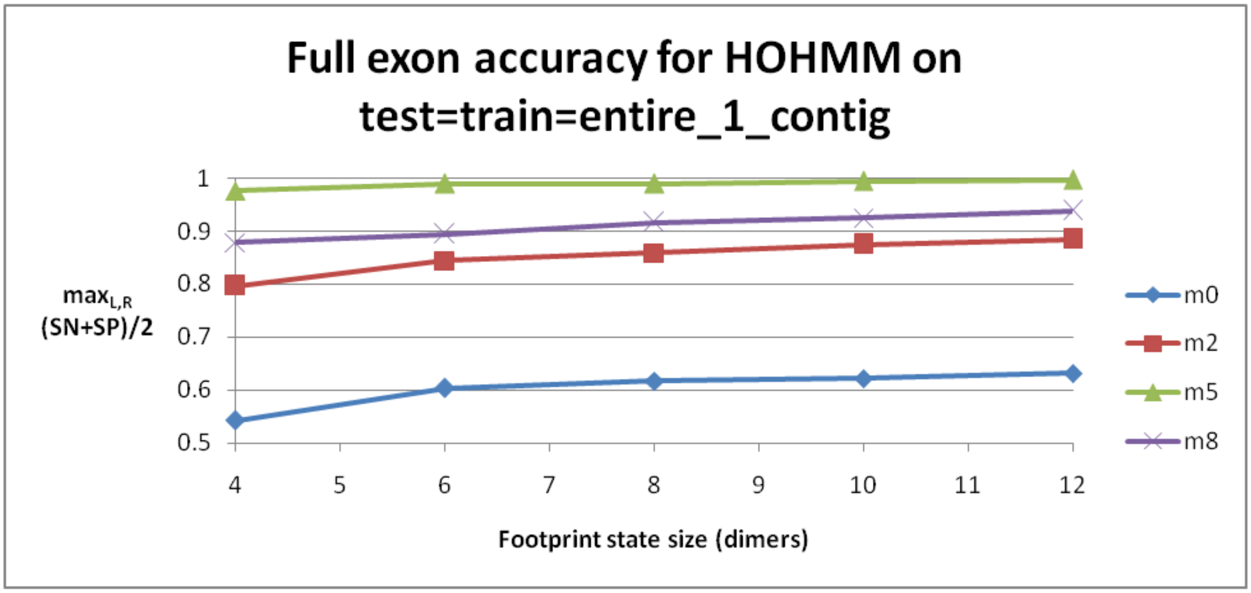


Figure 6 Maximum full exon HOHMM performance for data entire_1_contig

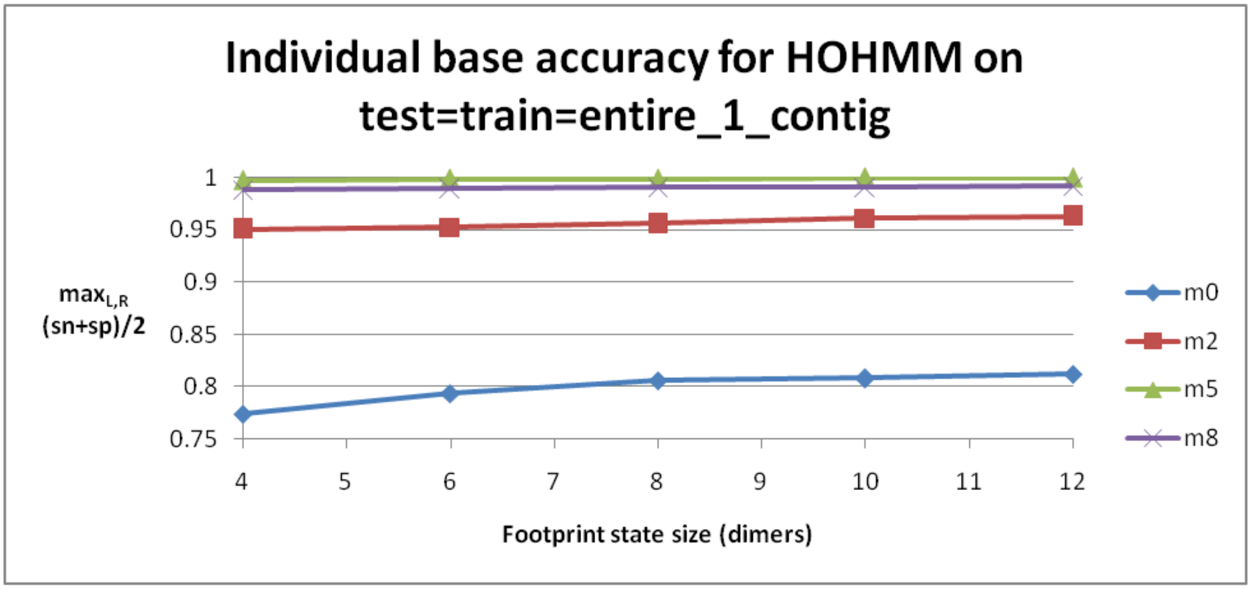


Figure 7 Maximum base level HOHMM performance for data entire_1_contig

3.5 The Benchmark Data Set ALLSEQ

3.5.1 Data Preparation

In [**17**], the authors include the steps used in order to arrive at the ALLSEQ data set [**45**], which we quote here and reiterate in itemized form further below.

> "The sequences that constituted the test set to benchmark the programs analyzed were obtained from the vertebrate divisions of Genbank Release 85.0 (October 15, 1994). The set of all genomic DNA sequences from these divisions encoding at least one complete protein coding gene was initially considered. In practice the sequences were extracted, annotated with the keyword ''DNA'' in the LOCUS field, containing at least one ''CDS Key'' in the FEATURES table.
> From this initial set, the following sequences were discarded: the sequences encoding at least one incomplete protein product (the ''CDS Location'' indicated an incomplete protein product or the ''CDS Key'' was labeled with the Qualifier ''/partial''), the sequences for which the exact location of the protein coding regions was not unambiguously determined (the ''CDS Location'' contained the Operator ''one-of''), the sequences encoding protein coding genes in the complementary strand (the ''CDS Location'' contained the Operator ''complement''), the sequences encoding protein coding genes defined in data-entries other than the one corresponding to the sequence (the Operator ''join'' in the ''CDS Location'' contained the Accession Number of a database entry), the sequences encoding pseudogenes (the "CDS Key" was labeled with the Qualifier ''/pseudo''), the sequences more than one gene or alternatively spliced forms of the same gene (containing more than one ''CDS Key'' in the FEATURES table), and the sequences encoding protein coding genes without in trons (the ''CDS Location'' described a single continuous coding region)."
>
> "We obtained, in this way, the set of all vertebrate genomic DNA and sequences in the database encoding one complete (and only one) spliceable functional protein product in the forward strand. There were 1410 sequences in this set. The integrity of the data set was further enforced by discarding the following sequences: the sequences whose protein coding segment did not start with the codon ATG (74 sequences), the sequences whose protein coding segment did not end with a stop codon (100 sequences), the sequences whose protein coding segment was not a multiple of three (15 sequences), the sequences with donor sites lacking the dinucleotide GT as the first dinucleotide after the termination of the exons and with acceptor sites lacking the dinucleotide AT as the first dinucleotide before the initiation of exons (133 sequences), and the sequences whose protein coding segment contained stop codons in-frame (1 sequence)."
>
> "Finally, sequences corresponding to immunoglobulins and histocompatibility antigens were also discarded, as well as additional pseudogenes - as indicated in the sequence entry DEFINITION

field; 1043 sequences remained. After discarding three sequences longer than 50,000 bp, the final test set was obtained by selecting only the sequences with ‘‘date of entry’’ after January 1993, essentially equivalent to selecting the sequences entered into GenBank or modified since Release 74. We attempted, in that way, to minimize the overlap between the test set constructed here and the training sets used during the development of the programs analyzed. Al though it is not always possible to infer from the published papers the database release used to build the sequence set on which a given program has been trained, it appears that only Xpound and GenLang among the programs analyzed may have included sequences in their training sets entered into GenBank after Release 74. Total lack of overlap between the test set used here and the training sets of the programs analyzed, though, cannot be guaranteed for those programs accessed through Internet servers, which may have been updated since they were first published.”

“There were 570 sequences in the final test set, totaling 2,892,149 bp. There were 2649 coding exons, corresponding to 444,498 coding bp (which gives a coding density of about 15%). We will refer to this set as the ALLSEQ set. A subset of ALLSEQ was considered separately. It comprised those sequences in ALLSEQ that did not show a significant similarity to the vertebrate sequences entered into the database prior to January 1993. We will refer to this set as NEWSEQ. Our aim here was to build a test set of sequences that did not show similarity to the sequences used in the training sets of the programs analyzed. Performance of the programs in this set may provide the most objective estimation of the real performance of the programs, or at least of their ability to implement the general properties of the genic sequences and not only the particularities of the examples they learn from. In practice, to build the NEWSEQ set we compared the protein sequence encoded by each sequence in the ALLSEQ set with the set of protein sequences encoded by the vertebrate genomic DNA sequences among the 1043 sequences resulting from the above procedure that had been entered into the GenBank database prior to January 1993. To compare the protein sequences we used the BLASTP program (Altschul et al., 1990). Only the sequences in ALLSEQ for which no HSP score greater than 100 was found when compared with the above set of sequences were included in the NEWSEQ set. There were 196 sequences in this set. Although sequences in the NEWSEQ data set are on average substantially longer (6838 bp) than sequences in the overall ALLSEQ data set (5073 bp), coding density and G / C content were essentially identical in both sets (coding density is 0.14 in NEWSEQ and 0.15 in ALLSEQ; G / C content is 0.50 in NEWSEQ and 0.49 in ALLSEQ). It does not therefore appear that genes encoded by the NEWSEQ sequences are intrinsically more difficult to predict than genes encoded by sequences in the overall ALLSEQ set."

1) Select the set of all sequences encoding at lease one complete protein from the vertebrate divisions of GenBank Release 85.0 (October 15, 1994).
2) From the above discard the following:
    a. Any sequence encoding at least one incomplete protein.
    b. Any sequence for which the exact coding regions was not unambiguous.
    c. Any sequence encoding a protein in the complementary (reverse encoding) strand.
    d. Any sequence containing a gene or part of a gene associated with other sequences.
    e. Any sequence encoding a pseudogene [1] (via "CDS Key" value "/pseudo").
    f. Any sequence encoding more than one gene or alternative splicing of a gene.
    g. Any sequence encoding a gene without introns.
3) From the 1410 sequences resulting from the above the following further discards were made:
    a. Any sequence whose coding segment did not start with the start codon ATG.
    b. Any sequence whose coding segment did not end with a stop codon (TAA, TAG, TGA).
    c. Any sequence whose coding segment was not a multiple of 3 in length.
    d. Any sequence with any intron not beginning with GT and/or ending with AT (sic).
    e. Any sequence whose coding segment contained an in-frame stop codon.
4) The following additional discards were made:
    a. Sequences for immunoglobulins, histocompatibility antigens and additional pseudogenes not discarded using previous criteria.
    b. 3 sequences longer than 50,000 bp.
5) One final selection was made from the sequences surviving the above in that the sequence's date of entry postdated Release 74 of Genbank (January, 1993) – intended as such to minimize the overlap of the resulting test set with training sets for the programs tested in [**17**].

As mentioned previously, because the training and testing sets were identical, we consider these results as a brute force parameter search yielding what to expect in the ideal case and not necessarily a valid test of prediction performance. At this writing, no suitable, separate training set is attainable, since a recent query via web-based utilities at NCBI [**46**] indicated that Genbank no longer contains any vertebrate sequence entries dated during or prior to December, 1992. Thus a cross-validation analysis would seem to be the only viable alternative for testing with the ALLSEQ data set. However, it is worth noting that whereas the authors in [**17**] did physically separate the test set from the training set by a date of entry criterion, no direct determination was made as to the actual degree of overlap between the testing and training data sets. This observation is duly noted in [**4**], with the additional observation that such overlap was inevitable since the ALLSEQ data set did contain the "vast majority" of vertebrate sequences available at the time. It is reasonable then to consider these results a fair comparison with those

[1] "Pseudogenes are defunct relatives of known genes that have lost their protein-coding ability or are otherwise no longer expressed in the cell." [**64**]

reported by Burset and Guigo, though perhaps still at risk of overestimating performance accuracy for the same reason.

The authors of [**4**] also provide an informative discussion and collection of references on exon and intron durations among others. In characterizing one reference in particular [**47**], the authors observe “that the in-phase hexamer measure, which measures the frequency of occurrence of oligonucleotides of length six in a specific reading frame, is the most effective” for inclusion in gene finding. Moreover, the authors have assembled their own test data set, called HMR195 [**48**], based on sequences submitted to Genbank after August 1997. Whereas this may once again raise concern of overlap similar that of the ALLSEQ tests, the authors and others [**49**] argue that an independent test data set “should not necessarily exclude the sequences that are similar to the sequences in the training set, because, realistically, the unannotated sequence submitted to the gene-finding program might also be similar to an already known and characterized sequence that has been use in the program's training set.” As such, this data set was chosen after the time had passed for the training of the programs tested, thus addressing the authors’ dual purposes of bypassing the training/testing overlap issue of the ALLSEQ data set as well as constructing a more appropriate test of the more recently developed gene finding software at that time. By such time, many gene finding programs had begun to incorporate higher order dependencies in genomic data into their models.

Though time did not allow here, the HMR195 test set and reported results would likely present a more appropriate comparison to HOHMM performance than those using ALLSEQ for the following reasons.

1) The data set HMR195 is more recent than ALLSEQ.
2) The tests reported using HMR195 have substantially less risk of overestimating performance accuracy due to less overlap with the training sets involved, and
3) The authors indicate that all 7 programs tested in their report used intrinsic information only – “coding statistics and signal models” learned directly from the respective training sets.

Thus, a valid performance test would conceivably involve a training set formed from the complement of the test set – or vertebrate sequence entries during or before August, 1997.

We proceed nonetheless for now with the results of the parameter search using the ALLSEQ dataset, whose properties are summarized below in Table 13.

Table 13 Properties of the ALLSEQ data set

# Bases	Coding	Sequences			Introns			Exons		
	Density	Total	BP	Avg. Len.	Total	BP	Avg. Len.	Total	BP	Avg. Len.
2892149	0.15	570	1754950	3078.86	2079	1310452	630.33	2649	444498	167.80

3.5.2 Summary of Results for Benchmark Data ALLSEQ

For detailed results for ALLSEQ, including combined 3D surface and contour graphs of both exon- and base-level accuracy for all values of the parameters M, F, L, and R tested, see **7.2 Detailed Results for ALLSEQ,** page 99.

Figure 8 and Figure 9 below contain plots for exon- and base-level maxima, respectively, over the parameters L and R of HOHMM's prediction performance. Here again for ALLSEQ, they illustrate the enhanced performance of HOHMM over simpler prediction models, including the (null hypothesis result) HOHMM for which the base Markov parameter, M=0. It is worth noting that the HOHMM uses only the *intrinsic* information in the data – making no use of *extrinsic* information, such as EST's, protein homology, etc. Also, here as for the entire_1_contig data set, time did not allow to complete the parameter search with this data set for a local maximum in the exon nor in the base level predictions. (See, however, **3.6.2 Summary of Results for Chromosomes I-V of *C. elegans***, page 61.)

In comparing the results of this data set to the other results in this effort, the quality of the best result can be attributed to the increased size of the training data set (despite the decreased coding density) as well as adherence among the donor and acceptor splice sites to the consensus sequences, gt and ag, respectively. Moreover, the enhanced performance at full exon level for M=8 compared to that for M=5 is an additional indication of sufficient training size in support for M=8 probability estimates at splice sites. (Here, as for entire_1_contig, time did not allow the computation and inclusion of relative entropy values. See **3.7 Relative Entropy of C/NC Transitions**, page 65.)

4 For reference to the original benchmark results, the plots also best performing predictors from the original benchmark study, intrinsic and extrinsic genomic information, respectively. (See measurement, 3.2 Method of Measurement for Prediction Performance, page 46.) At both the full exon- and base- levels, the HOHMM measurements for M=8 and as such for all values of F tested (again using only intrinsic information from the data) appear higher consistently and by a discernable margin than both the top performing, intrinsic-only competitor, FGENEH, as well as the top performing, intrinsic-and-extrinsic competitor, GeneID+. For further discussion including testing methodology, see 5 Discussion and Future Work Discussion and Future Work

, page 79.

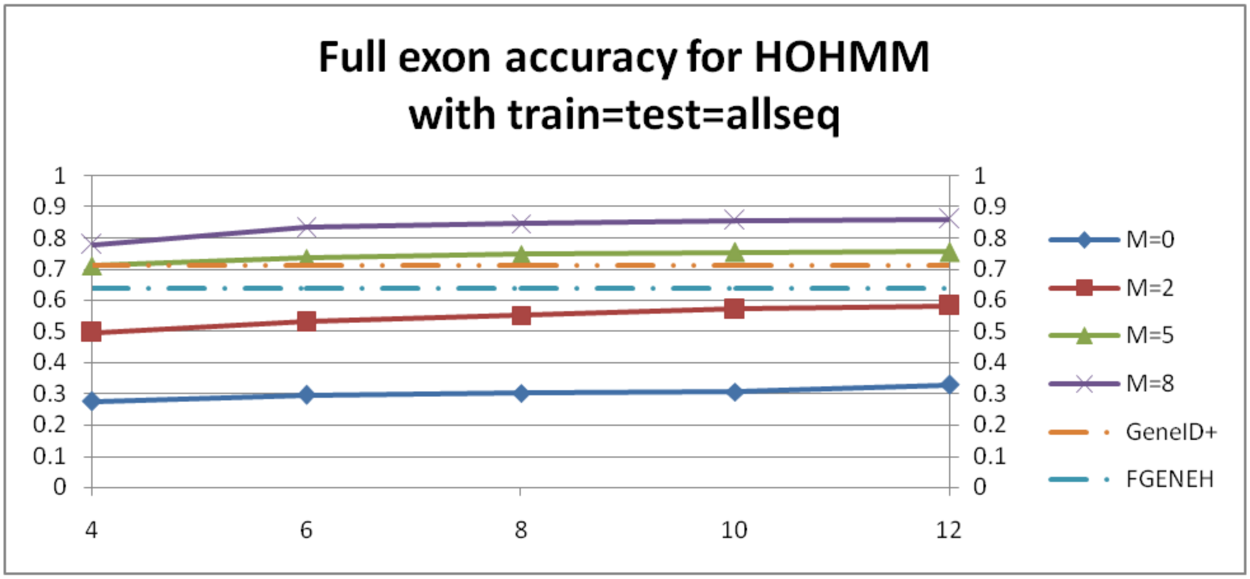


Figure 8 Maximum full exon HOHMM performance for data ALLSEQ

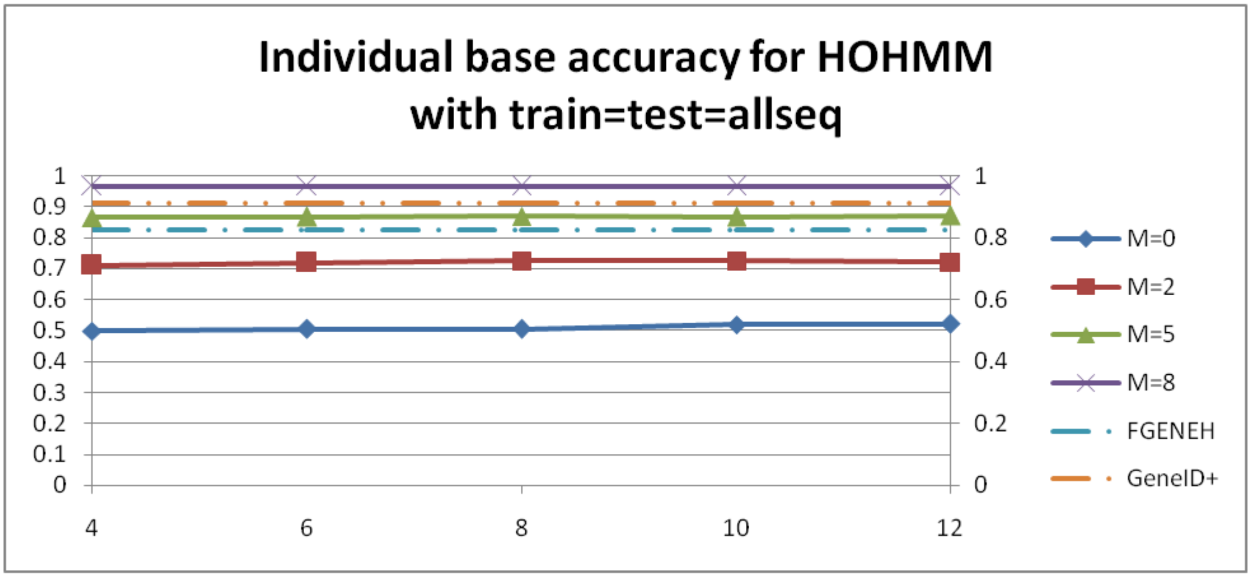


Figure 9 Maximum base level HOHMM performance for data ALLSEQ

4.1.1 In-frame Stop Filtering

The occurrence of a stop codon (TAA, TAG, or TGA) that is in frame (starting at reading frame 0) and located internally in an exon – rather than in the expected last codon position in the last exon of a gene – is referred to as an "in-frame stop".

4.1.1.1 In-frame Stop Data Preparation

The authors state that no sequence with an in-frame stop was allowed in forming the ALLSEQ data set. However, in one sequence with 3 exons, OCTNFBETA, an in-frame stop was artificially introduced in its internal exon as the result of our method in practice of disambiguating any ambiguous nucleotide position in the data simply by overwriting it with the nucleotide A. The annotation supplied by the website for the sequence OCTNFBETA is shown here among its neighbors in the annotation file provided.

```
OCAPOAIG 510 552 701 854 1319 1919
OCTNFBETA 502 576 660 765 963 1375
OCUTGLOB 3302 3356 5643 5830 6158 6190
```

The annotation above for OCTNFBETA indicates there are 3 exons in the base positions 502-576, 660-765, and 963-1375. In this case, an ambiguously sequenced codon, TNA, occurring at base position 687 in the internal exon 660-765 was inadvertently changed into the in-frame stop codon TAA, but was used nonetheless as a convenience for this investigation into in-frame stop filtering. Also, training was performed using all 570 sequences, excluding the single exon from OCTNFBETA containing the (artificially introduced) in-frame stop. Nonetheless, testing did include the sequence OCTNFBETA in its entirety.

Otherwise, the issue of in-frame stops and their filtering is not present per se in the ALLSEQ data set nor in the raw *C. elegans* genomic data nor the entire_1_contig of *C. elegans*.

4.1.1.2 In-frame Stop Filtering Algorithm

In general, in-frame stops are known to occur in some organisms, but such occurrences are considered rare, and such organisms require special biomolecular mechanisms in order to avoid premature termination of the translation process. We make no effort here to distinguish the true genomic in-frame stops from those that are false, presumably as artifacts of the curation process. As such, we view the in-frame stop as an error in the curation process and thus (provide the option to) attempt to actively filter such an event out by penalizing it during possibly multiple passes of the Viterbi prediction process. This is done according to user-specified values for both penalty factor and maximum number of passes. It is worth noting that a tradeoff has been made in the design of the algorithm, where reduced complexity of implementation was favored over the impact to execution time. This tradeoff involves the use of multiple passes through the Viterbi prediction results, first for counting, then for recording the locations of in-frame stops, etc. These locations are then penalized in subsequent Viterbi passes.

The pseudo-code for iterative, in-frame stop filtering is as follows.

1) Perform the 1st pass of Viterbi table construction and optimum path traceback as usual.
2) Scan the predicted exon coding segments and make note of the e_0 (frame 0) positions of all the in-frame stops – including any in-frame stops interrupted by introns as well.

3) Perform the next Viterbi pass, encountering each such flagged position and penalizing any footprint transition contributing to the continued e_0 prediction of the base emitted at the flagged site.
4) Repeat steps 2 and 3 above until either all in-frame stops have been removed or the specified maximum number of passes has been met.

4.1.1.3 Results of In-frame Stop Filtering on OCTNFBETA

Again only the one sequence, OCTNFBETA, was tested due to its (artificially introduced) in-frame stop. Recall from above that the goal of in-frame stop filtering is to discover the in-frame stop codon and heavily penalize the corresponding Viterbi probability sufficiently so that some alternate path is chosen as more probable in the subsequent Viterbi pass.

A value of -10 was chosen for the log-penalty factor and used throughout. This was informed by observations of the change in log-probability values in the Viterbi probability table. The average decrease in log-probability between consecutive columns in the Viterbi probability table was on the order of -1 for most rows, including those rows responsible for promoting the in-frame stop interpretation $e_0e_1e_2$ where the penalty was applicable, though decreases as much as -4 were observed in some other rows.

In order to test the in-frame stop filtering, it was first necessary to identify all those prediction results without the iterative, in-frame stop filtering where the in-frame stop was predicted as such. As shown in Table 14 below, of all the combinations of HOHMM parameters tested, only 4 specific combinations resulted in the detection of *any* portion of the internal exon, 660-768, whose predicted start position is highlighted in the table. More specifically, 2 of those combinations naturally excluded the in-frame stop after the 1st Viterbi pass, but the other 2 combinations required a 2nd Viterbi pass in order to filter out the in-frame stop. In all 4 cases, both boundary exons were successfully predicted after the 1st Viterbi pass.

Table 14 Combinations of HOHMM parameters resulting in in-frame stop after 1st Viterbi pass

M	F	L	R	HOHMM predicted exons for OCTNFBETA			
				After Viterbi pass 1		After Viterbi pass 2	
				Start	End	Start	End
5	4	8	0	502	576	502	576
				660	768	711	768
				963	1375	963	1375
5	4	8	1	502	576	502	576
				711	768	711	768
				963	1375	963	1375
5	4	9	0	502	576	502	576
				660	768	711	768
				963	1375	963	1375
5	4	9	1	502	576	502	576
				711	768	711	768
				963	1375	963	1375

In order to show the results via the standard 3D graphs, it would be necessary to modify the annotation provided for the sequence OCTNFBETA by deleting the internal exon 660-768. Time did not allow for the generation of the standard 3D graphs, but in place of those graphs, we simply state the result here as already shown in Table 14 above. Though limited in scope, these results are a positive indication of the efficacy of this approach for in-frame stop filtering.

4.2 Chromosomes I-V of *C. elegans* With 5-fold Cross-validation)

The data for Chromosomes I-V of *C. elegans* were obtained from release WS200 of Wormbase [**18**]. We note that the sixth and final chromosome of *C. elegans*, designated for legacy reasons as Chromosome X, was excluded from this analysis as it is known to have substantial differences in gene encoding properties as compared to Chromosomes I-V [**24**].

4.2.1 Data Preparation and Procedure for Chromosomes I-V of *C. elegans*

The following steps were used in order to prepare the data prior to training and testing.

1) The data was scanned for in-frame stops, and ultimately no in-frame stops were detected as confirmed by Wormbase curator Paul Davis [**43**].
2) The data was scanned for alternative splicing, and 2974 (14.5%) out of a total of 20515 sequences represent alternative splicing – including some forward encoded alternative splicings interfering with other, reverse encoded alternative splicings.
3) In order to avoid the complexities involved in the prediction of alternative splicings, the *transitive closure* with respect to overlap of all alternative splicings was deleted from the data and the remaining annotation was appropriately offset in compensation for the deletions. For all branches of all alternative splicing sequences – along with any sequences interfering with them - the following segments, s, were deleted:
   a. s=the 5'- UTR, where (15b< length(s) <=200b) (15=WS/2: See item 7 below)
   b. s=the 3'- UTR, where (15b< length(s) <=3kb), and
   c. s=the entire coding sequence, CDS, including exons and introns
4) In order to avoid both the complexity of segmented prediction as well as any bias toward any specific subset of chromosomes during cross-validation, the following were performed:
   a. Both data and annotation files for all 5 chromosomes were divided into a total of 70 autonomous chunks of nominal size 1Mb and minimum size 500kb.
   b. The resulting 70 chunks were then evenly distributed into five (5) groups for 5-fold cross-validation.
5) Training was performed independently on each of the above chunk groups with a sampling window size of first WS=30, then WS=40.
6) Counts from training on chunk groups 1-4 were combined to form probability estimates used to test on chunk group 5, then training on 2-5 for testing on 1, and so on.
7) Results reported are derived from the sums of the various corresponding counts (such as full exon match, exon-base match, etc.) resulting from the 5 sets of test trials above.

Table 15 Summary of data reduction in *C. elegans*, Chromosomes I-V

Summary of data reduction in *C. elegans*, Chromosomes I-V						
File	# sequences	# alt.	% alt.	# exons	# alt.	% alt.
CHROMOSOME_I	3537	619	17.50%	24295	5251	21.61%
CHROMOSOME_II	4161	629	15.12%	25427	5106	20.08%
CHROMOSOME_III	3277	590	18.00%	21541	4400	20.43%
CHROMOSOME_IV	3886	560	14.41%	24390	4478	18.36%
CHROMOSOME_V	5654	576	10.19%	32135	4201	13.07%
Total	20515	2974	14.50%	127788	23436	18.34%

Table 16 Properties of data set *C. elegans*, Chromosomes I-V (reduced)

# Bases	Coding Density	Sequences			Introns			Exons		
		Total	BP	Avg. Len.	Total	BP	Avg. Len.	Total	BP	Avg. Len.
69024243	0.20	14813	28391591	1916.67	67093	14742264	219.73	81906	13647511	166.62

4.2.2 Summary of Results for Chromosomes I-V of *C. elegans*

For detailed results for Chromosomes I-V of *C. elegans*, including combined 3D surface and contour graphs of both exon- and base-level accuracy for all values of the parameters M, F, L, and R tested, see **8.3 Detailed Results for Chromosomes I-V of *C. elegans***, page 110.

It is worth noting that for this data set, twice as many trial sets were run – corresponding to twice as many values of the parameter F for footprint size. The reason for this is to capture at least a local maximum in prediction performance for both the full exon and base level predictions. The tabulated results indicate that a local maximum for the exon and base level predictions were attained at F=15 and F=12, respectively.

Table 17 and Table 18 below show the resulting values and Figure 10 - Figure 13 below contain plots for the average over the 5 folds of HOHMM's prediction performance for exon- and base-level, respectively, with maxima implicitly taken over the parameters L and R.

In comparing the results of this data set to the other results in this effort, the reduced performance at full exon level for M=8 compared to that for M=5 (as in entire_1_contig) is an indication of insufficient training size reflected in lack of support for M=8 probability estimates at splice sites. (See **4.3 Relative Entropy of C/NC Transitions**, page 65.)

In spite of a coding density greater than ALLSEQ, the lower prediction accuracy for this data set compared to those of entire_1_contig and ALLSEQ can be attributed in part to the 5-fold cross-validation procedure used for this data set and not for the latter. Moreover, the level of preconditioning is lower for this data set, such as allowance in the data for disagreement with the consensus sequences, gt and ag, at donor and acceptor splice sites, respectively, as well as the allowance of reverse encodings.

In order to improve the prediction accuracy for this data set and others so conditioned, we would seek to include additional, similar genomes such as other round worms besides *C. elegans* for incorporation in the training data set. With this data set alone the number of c/nc sample instances is on the order of 10^3. In general, this is short of the support required (4^9 or ~10^5) for the use of 8th order Markov dependency among the bases at the c/nc transition sites.

Finally, there are other pending modifications to the core HOHMM application. (See **6 Discussion and Future** Work, page 79.)

Table 17 Exon level HOHMM accuracy of 5-fold cross-validation trials on *C. elegans* I-V

F\M	0	2	5	8
2	0.39985	0.60708	0.66158	0.49026
4	0.40564	0.6231	0.67501	0.51028
6	0.43967	0.71362	0.71037	0.52767
8	0.45253	0.72424	0.71945	0.54618
10	0.47289	0.73006	0.72376	0.55752
12	0.48044	0.73376	0.72772	0.568
14	0.48529	0.73995	0.7022	0.5438
16	0.49135	0.74236	0.69982	0.54655
18	0.49534	0.74172	0.70111	0.54737
20	0.49858	0.74303	0.70267	0.55007

Table 18 Base level HOHMM accuracy of 5-fold cross-validation trials on *C. elegans* I-V

F\M	0	2	5	8
2	0.68591	0.8715	0.88803	0.81979
4	0.69029	0.87465	0.89216	0.82666
6	0.69809	0.88878	0.89734	0.83259
8	0.70397	0.89225	0.89914	0.83991
10	0.71345	0.89408	0.90027	0.84244
12	0.71832	0.89529	0.90123	0.84541
14	0.72336	0.89796	0.89587	0.83792
16	0.72617	0.89893	0.89601	0.83902
18	0.72882	0.89886	0.89655	0.83852
20	0.73082	0.89913	0.89682	0.83938

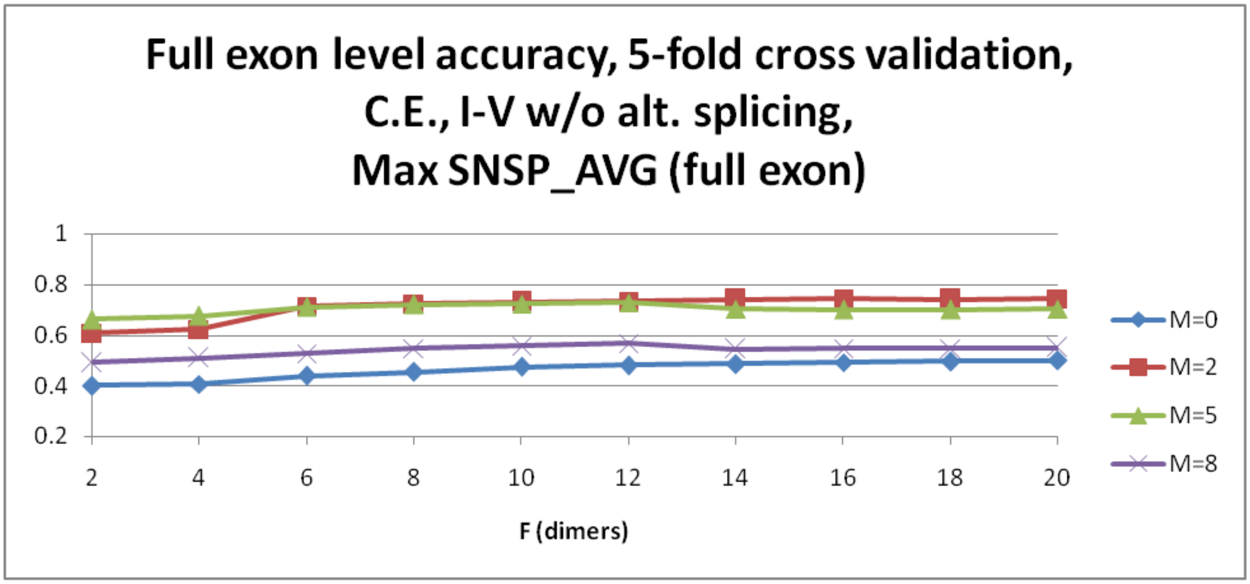


Figure 10 Full exon level accuracy for *C. elegans* 5-fold cross-validation (F view)

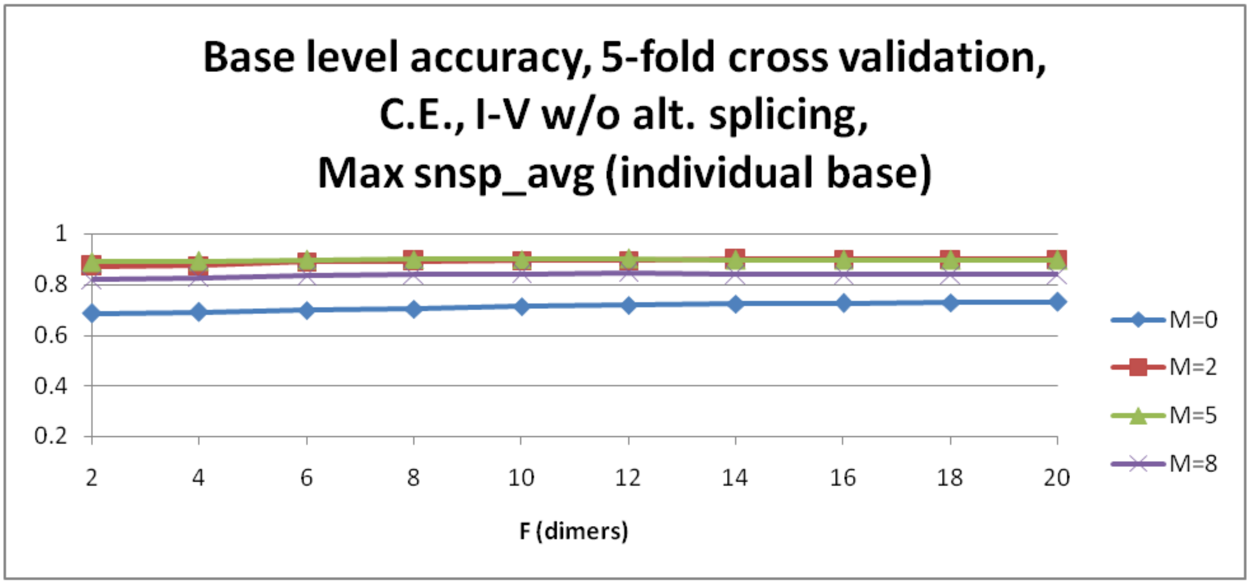


Figure 11 (Nucleotide) Base level accuracy for *C. elegans* 5-fold cross-validation (F view)

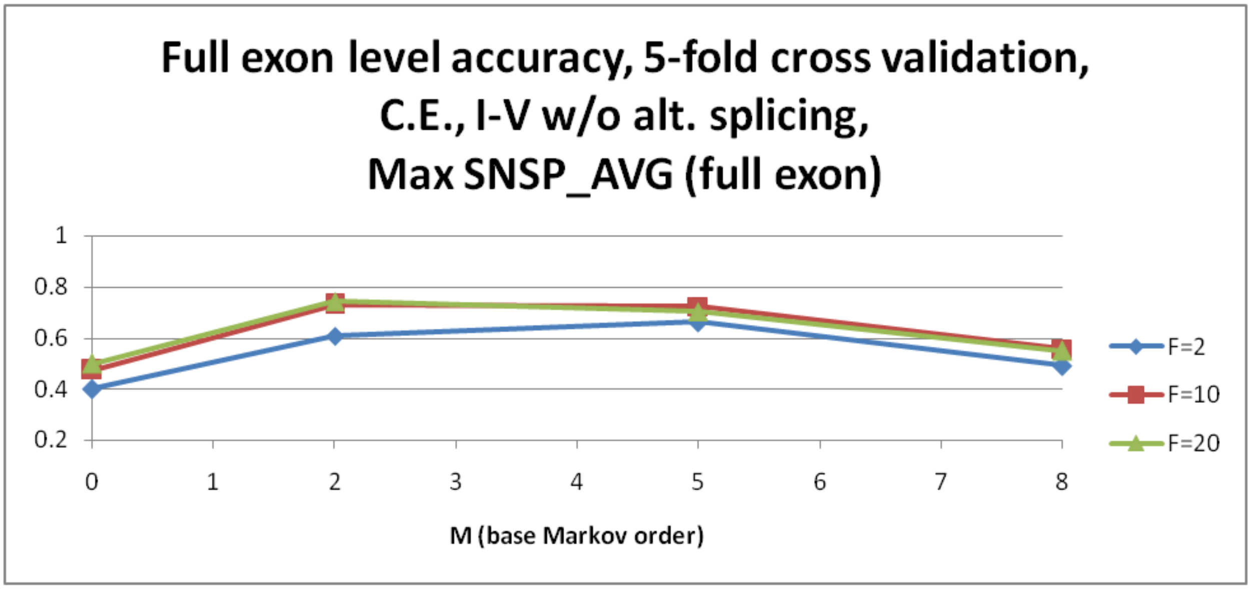


Figure 12 Full exon level accuracy for *C. elegans* 5-fold cross-validation (M view)

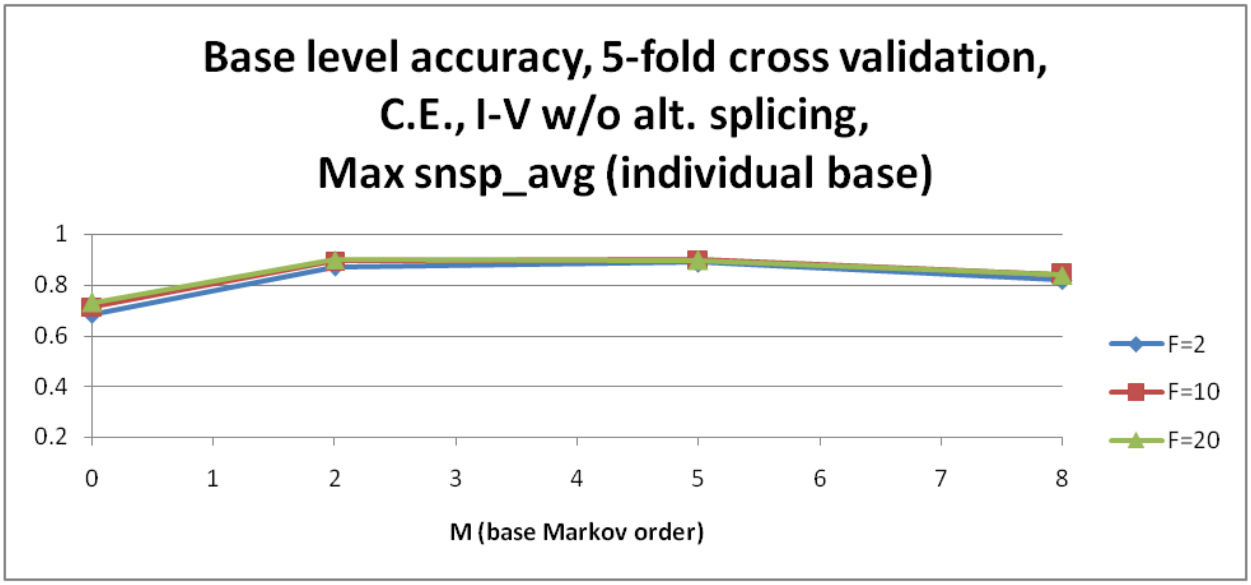


Figure 13 (Nucleotide) Base level accuracy for *C. elegans* 5-fold cross-validation (M view)

4.3 Relative Entropy of C/NC Transitions

In addition to their extensive bibliography in [**32**], the authors include an informative example of a relative entropy analysis comparing 1st order to 0th order dependence of consecutive bases in a sample of 757 acceptor splice sites taken from the human genome. The example clearly indicates a significant level of 1st order to 0th order relative entropy in the two positions of the AG consensus. Moreover, in the typical eukaryotic genome, there are significant levels of relative entropy between transition and non-transition states for higher orders of dependency between consecutive bases. (See, for example, the discussion and coding and signal strength in [**50**].)

Thus, as part of a comprehensive parameter search for the HOHMM, this higher order relative entropy can be an invaluable input in parameter tuning of the HOHMM, including the determination of optimal base Markov order, M, as a function of both the eij-transition state as well as the position of the base in question relative to the eij-dimer generating the given eij-transition state. In this effort, time did not allow the variation of the base Markov order as such – one specific scheme for specifying the base Markov order as a function of state and position was chosen at the outset and used throughout.

There is yet another aspect of parameter tuning for the HOHMM. As already suggested above in **2**, the HOHMM with its extended observation or base emission string allows for the use of a vector-based state transition, e.g. splice site, sensor as an alternative to the typical scalar-based discriminator of both the HMM and the HOHMM. In the work reported in [**25**], such an alternative sensor was used – with promising results – in a scope limited to donor and acceptor splice sites. There the kernel of an SVM was constructed based on vectors whose components were taken, among other possibilities, as the emission probabilities of positions in a window centered on donor and acceptor splice sites, respectively. The resulting set of SVM-based discriminators attained promising results when used to determine locations of splice sites in test sets containing true and false splice sites for several different vertebrate species. Again, though time did not allow the incorporation of an SVM-based classifier into the HOHMM, the scalar discriminator of the generalized Viterbi algorithm of the HOHMM (for stepwise determination of the most probable state) could readily be replaced with an SVM-based, vectorized discriminator – for each of 8 different transition contexts as listed in the following.

Recall that the HOHMM used herein includes 8 distinct c/nc transitions (4 each of c→nc and nc→c). For each such transition probability there is the corresponding non-transition probability as the prediction proceeds from left to right along the testing data, such as in the ordered pairs (ei, ee)x3, (ie, ii)x3, (je_0, jj), and (e_2j, ee). Thus we compute of the relative entropy of the transition probability relative to the corresponding non-transition probability for all positions, i, (as support allows) in a sampling window of fixed and uniform length, as enumerated in the following:

1. H(P(je_0,M,i;P(jj,M))
2. H(P(e_0i_0,M,i;P(ee(i), M))
3. H(P($e_{01}i_{01}$,M,i;P(ee(i), M))
4. H(P($e_{02}i_{02}$,M,i;P(ee(i), M))
5. H(P(i_0e_1,M,i;P(ii, M))
6. H(P(i_1e_2,M,i;P(ii, M))
7. H(P(i_2e_0,M,i;P(ii, M))
8. H(P(e_2j,M,i;P(ee(i),M))

where M represents the base Markov order, and
ee(i) represents the cyclical variation among the three frame specific dimers e_0e_1, e_1e_2, and e_2e_0.

For detailed results of relative entropy, including individual graphs of relative entropy for each of the 8 c/nc transition types enumerated above, see **8.4 Detailed Results of Relative Entropy for Chromosomes I-V of *C. elegans***, page 131.

As a summary, Figure 14, is a 3-by-4 arrangement of 12 graphs of the same information as in the initial 16 but grouped differently according to the transition dimer classes, je_0/e_2j, ei, and ie, signifying gene boundaries, donor splice sites, and acceptor splice sites, respectively. These graphs are arranged vertically according to the base Markov order of the related probabilities and horizontally according to the transition dimer classes just mentioned above. In general, any extended region of elevated relative entropy could serve as a promising source of (position-dependent) emission probabilities. The corresponding emission probabilities would in turn be used as vector components, possibly in combination with emission probabilities of different Markov orders, in an SVM-based, c/nc transition discriminator.

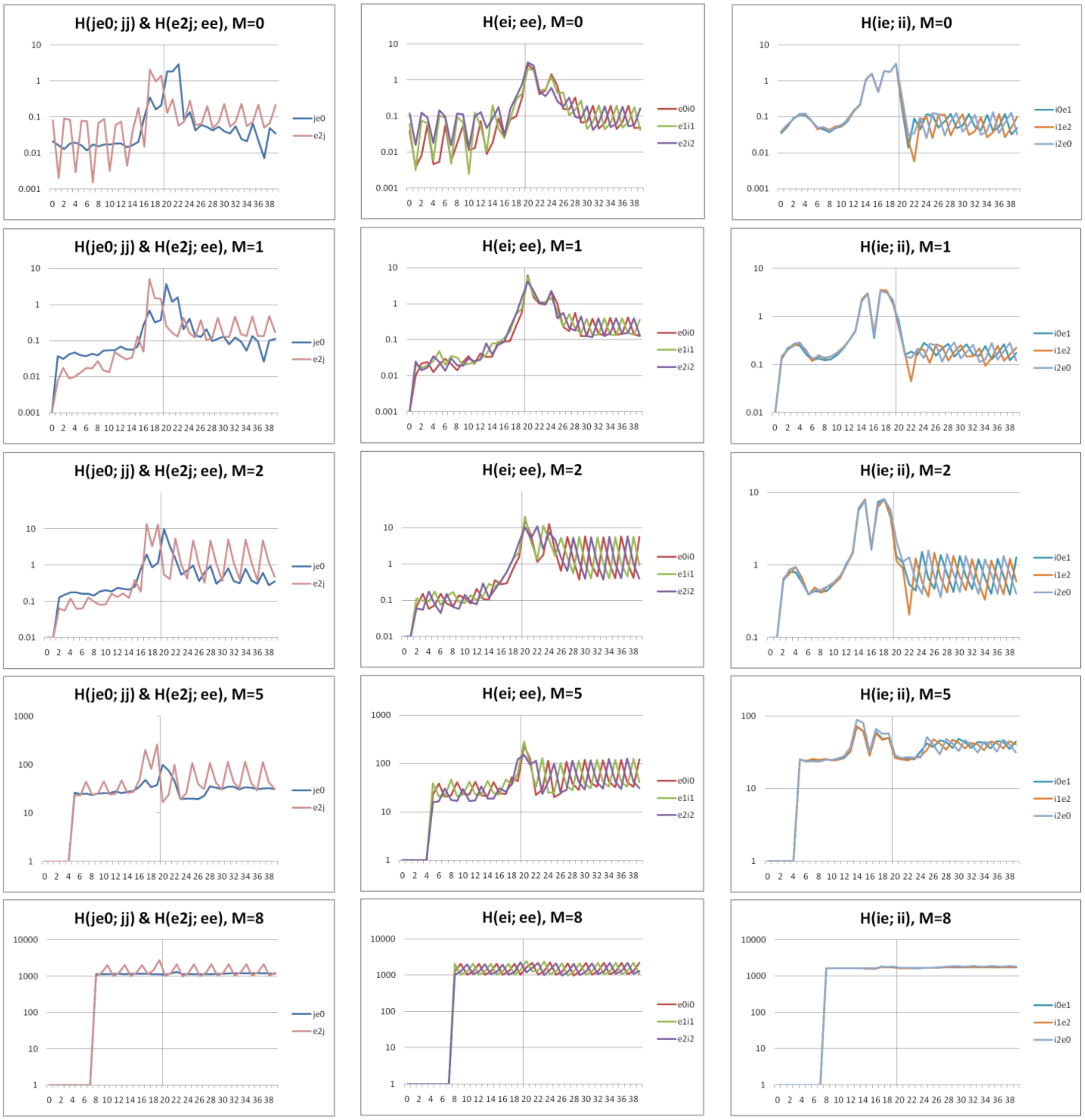


Figure 14 Summary of relative entropy, H(P(eij, M, i), P(xx, M,i)), for reduced C.E., I-V with log scaling. (Graphs are arranged vertically according to base Markov order, M, and horizontally according to classes of transition dimer, je_0/e_2j(left), ei(middle), and ie(right).)

5 Other HMM Applications

5.1 Additional Applications of Iterative HOHMM

The following sections outline uses of an iterative HOHMM for various preprocessing applications.

5.1.1 Sequence-gain Preprocessing

We suggest the use of an iterative HOHMM as a preprocessing tool for identification of *open reading frames*[2] to be used on genome-scale data sets where sufficient annotation is already known and available for use in training. The term *sequence gain* refers to the factor applied to the computed probability of transition from primitive state `j` to e_0 – or the onset of a coding sequence. The pseudo-code for this application is as follows.

1) Pass 1: Run HOHMM with train=test in order to determine optimum HOHMM parameters, M, F, L, R.
2) Pass 2: Rerun HOHMM, again with train=test and optimum parameter values determined above but variable sequence gain, G – in order to determine coding sequence neighborhoods as well as optimum sequence gain, G, according to genomic sequence density.
3) Pass 3: Rerun HOHMM, with same training set as above and optimum M, F, L, R, but G=1 on sequence segments resulting from previous step.

5.1.2 Iterated Motif Detection/Boosting

Here also, we assume that the training has been performed using a genome-scale data set. The identification of motifs can help to predict gene start- and stop-sites, where the signals are usually weaker than that of splice sites.

1) Pass 1: Run HOHMM on test data with optimum M, L, F, and R.
2) Pass 2: Count 12/15-mers and identify those instances (and their predicted state context from above) occurring well above, for example, 2x the otherwise average count according to uniform distribution. Also determine ratio of peak-to-uniform-average frequency.
3) Pass 3: Rerun HOHMM on test data applying gain (in the predicted context) to 12/15-mers identified above, where gain is equal to ratio determined above.

5.1.3 Alternative In-frame Stop Filtering

As an alternative method to that investigated above (see **3.5.3 In-frame Stop Filtering**, page 58), the same procedure as for motif boosting above could be performed, only applying a penalty factor (< 1) instead of gain factor (> 1).

5.2 HMM with Duration

5.2.1 Revisiting the Conventional HMM

An HMM consists of 2 main computations (by Durbin et al. in [5]):

1. Baum-Welch Iteration of recursively defined forward/backward probabilities (symbols x, states π)

[2] "Strings of codons uninterrupted by stop codons are known as **open reading frames** (ORFs) and are a distinguishing feature of many prokaryotic and eukaryotic genes." [**61**]

$f_\ell(i+1) = P(x_1 \ldots x_{i+1}, \pi_{i+1} = l) = e_\ell(x_{i+1}) \pi_k f_k(i) a_{k\ell}$

$b_k(i) = P(x_{i+1} \ldots x_L | \pi_i = k) = \pi_\ell a_{k\ell} e_\ell(x_{i+1}) b_\ell(i+1)$

2. Viterbi Path Determination

$v_\ell(i) = e_\ell(x_i) \max_k(v_k(i-1) a_{k\ell})$

$ptr_\ell(i) = \mathrm{argmax}_k (v_k(i-1) a_{k\ell})$

(We note as mentioned in [**29**] that yet a 3rd type of computation can be made in order to determine the probability that a given sequence of observations would be emitted by a given HMM, once having been fully specified by the appropriate transition and emission probabilities.)

In the HMMwD described here, each of the stationary transition probabilities $a_{k\ell}$ are replaced by a dwell-time dependent update factor. The forward/backward probabilities used in the standard HMM-EM algorithm occur when evaluating the sequence probability $p(Z_{0 \ldots L-1})$ by breaking $p(Z_{0 \ldots L-1})$ into two pieces via use of a single hidden variable treated as a Bayesian parameter: $p(Z_{0 \ldots L-1}) = \Sigma_k\, p(Z_{0 \ldots i}, s_i=k) p(Z_{i+1 \ldots L-1}, s_i=k) = \Sigma_k\, f_{ki} b_{ki}$, where $f_{ki} = p(Z_{0 \ldots i}, s_i=k)$ and $b_{ki} = p(Z_{i+1 \ldots L-1}, s_i=k)$. Given stationarity, the state transition probabilities and the state probabilities at the ith observation satisfy the trivial relation $p_{qi} = \Sigma_k a_{kq} p_{k(i-1)}$, where $p_{qi} = p(S_i=q)$, and $p_{q0} = p(S=q)$, and the latter probabilities are the state priors. The trivial recursion relation that is implied can be thought of as an operator equation, with the operation of the product by a_{kq} followed by summation (contraction) on the k index. The operator equation can be rewritten using an implied summation convention on repeated Greek-font indices (Einstein summation convention): $p_q = a_{\beta q} p_\beta$. Transition-probabilities in a similar operator role, but now taking into consideration local sequence information via the emission probabilities, are found in recursively defined expressions for the forward variables, $f_{ki} = e_{ki}(a_{\beta k} f_{\beta(i-1)})$, and backward variables, $b_{ki} = a_{k\beta} e_{\beta(i+1)}\, b_{\beta(i+1)}$. The recursive definitions on forward and backward variables permit efficient computation of observed sequence probabilities using dynamic programming tables. It is at this critical juncture that side information must mesh well with the states (column components in the table), i.e., in a manner like the emission or transition probabilities. Length information, for example, can be incorporated via length-distribution-biased transition probabilities (introduced in [**23**]), and that is what is experimentally validated done here.

5.2.2 HMM with Duration via Cumulant Transition Probability

The material in this and following sections is taken largely from [**19**]. The transition probabilities for state e to remain in state e, an ee transition can be computed as: Prob(ee|L_e = L) = Prob($L_e \geq$ L+1)/Prob($L_e \geq$ L). The transition probabilities out of state 'e' can have some subtleties, as shown in the following where the states are exon (e), intron (i), and junk (j). In this case, the transition probabilities governing the following transitions, (jj) –> (je), (ee) –> (ej), (ee) –> (ei), (ii) –> (ie) are computed as: Prob(ei| L_e = L) = Prob(L_e = L)/Prob($L_e \geq$ L) x 40/(40 + 60) and Prob(ej| L_e = L) = Prob(L_e = L)/Prob($L_e \geq$ L) x 60/(40 + 60), where the total number of (ej) transitions is 60 and the total number of (ei) transitions is 40.

The pseudo code to track the critical length information, on a cellular basis in the dynamic programming table, goes as follows:

```
1) Maintain separate counters for the junk, exon and intron regions.
2) The counters are updated as follows:
    a. The exon counter is set to 2 for a (je) – (ee) transition
    b. The exon counter gets incremented by 1 for every (ee) – (ee) transition
```

3) Prob($L_e \geq L+1$) is computed as: Prob($L_e \geq L+1$) = 1 - $\sum_{i=1..L}$ Prob($L_e = i$). Hence we generate a list such that for each index $k > 0$, the value 1 - $\sum_{i=1..k}$ Prob($L_e = i$) is stored.

As a brief aside, simplifying from a 3-state model, {e,i,j}, to a 2-state model, {e,i}, after n occurrences of state e, the 2 cases of update factor to handle are:

$P(e_{n+1}|ee..e_n) = P(L_e \geq n+1)/P(L_e \geq n)$
$P(i|ee..e_n) = P(L_e = n)/P(L_e \geq n)$

Similarly for n occurrences of state i, and there are no probability splitting ambiguities upon exiting state e as there is only one state to exit to in the 2-state system, (and there are no differences in the Viterbi and Forward/Backward transition probability alterations).

Consider, as an example, a simple extension of our 2-state notation to cover [a time series of] N+1 states: $\{e, i^1,...,i^N\}$. Suppose we are interested in the probability of an i after seeing a length 4 segment of e-states:

$P(i|eeee) = 1\text{-}P(e|eeee) = 1\text{-} P(L_e{=}5)/P(L_e{=}4)$

There are two types of transition rule that appear to result, one for Viterbi, with its maximum operation on paths, and one for Forward/Backward, with its sum operation. For the 2-state case, N=1, and these update rules involving "splitting" factors all become the same (see Results in [**19**]):

Viterbi update:
(1) Difference splitting: $p(i^k|eeee)=[1\text{-}P(L_e{=}5)/P(L_e{=}4)] * [1+P(i^k|e)–Avg_z P(Z|e)]$
(2) Ratio splitting: $p(i^k|eeee)=[1\text{-}P(L_e{=}5)/P(L_e{=}4)] * P(i^k|e)/[Avg_z P(Z|e)]$

In this situation we are not maintaining a sum rule on probabilities, here we are viewing each path thru the table in a manner consistent with the maxprob evaluation.

Forward/Backward update:
(1) Difference splitting: $p(i^k|eeee)=[1\text{-}P(L_e{=}5)/P(L_e{=}4)] * [1+P(i^k|e)–Avg_z P(Z|e)]/N$
(2) Ratio splitting: $p(i^k|eeee)=[1\text{-}P(L_e{=}5)/P(L_e{=}4)] * P(i^k|e)/[N*Avg_z P(Z|e)]$

The equation above with the factor $[1+P(i^k|e)–Avg_z P(Z|e)]/N$ provides a suitable "splitting factor", as i^k and e probabilities sum to one, remain positive, and have other nice properties. The splitting factor is not unique, however, as case (2) makes clear.

It is important to note that we appear to have some freedom on splittings of probabilities upon exiting a state (when we are using the length distribution cumulants to describe transition probabilities, etc.). This is merely an associated effect of that length distribution incorporation – now upon exiting that length distribution our main factor is $P(i^k|$some prior length of e), that factor is blind to the appropriate splittings amongst the "not e" states, and we must incorporate another factor to deal with the probability splitting – in the case of forward/backward, this is chosen to obey a probability sum to 1 on all cases, on Viterbi this must maintain each path's probability with proper weighting with respect to the others (consistent with the max-path operation).

5.2.3 Real-time Application – Cheminformatics

The conventional Hidden Markov Model (HMM) is first order and uses fixed transition probabilities for remaining in a given state, which leads to a geometric length distribution for remaining in that state [**32**] – i.e., conventional HMMs automatically impose geometric length distributions on their same-state regions, like exon or intron lengths or blockade level durations. An HMM-with-Duration (HMMwD) is an HMM where true, or a much more complete, knowledge of the length distributions on same-state regions is incorporated into the model. (See, for example, [**32**].) Here we describe a novel HMMwD where the non-geometric length distribution information is incorporated via dwell-time dependent transition probabilities for transitions from a state to itself [**19**]. New experimental results are shown, and compared to an exact HMMwD (described in [**51**]).

Part of the novelty of the new "cellular" HMMwD that is proposed is that it is defined at the cell-level in its dynamic programming table construction, much like the conventional HMM, with one column's computation only dependent on information held in the prior column (in an overall table computation involving a single pass through the table). Our HMMwD can be defined for either the Viterbi or the Forward/Backward algorithms (see Methods in [**19**]). This is convenient because we have a method for distributed HMM processing based on such table computations (see [**23**]). That method is shown to work very effectively for the Viterbi algorithm (similar distribution methods for the Forward/Backward distribution algorithm are also discussed there). The end-result of all of this is that very sophisticated feature extraction tools can be brought to bear for real-time pattern recognition informed (PRI) operation.

5.2.4 Real-time Processing Hardware/Software Setup

For reference in the following, we provide in this section an overview of the real-time setup for the cheminformatic context. Though time did not allow application of the HMMwD to actual cheminformatic signals in this effort, we include results of this (implicit) HMMwD on synthetic signals in the following section, **5.2.5 HMM with Duration Experimental Tests**. (For an additional, offline analysis of cheminformatic signals with explicit HMMwD see [**51**].)

In the real-time setup, a client machine is configured with a data acquisition interface (DAQ [**52**]) for direct transfer of control signals and data to and from the lab apparatus. The client is also placed on a network with compute servers available for requesting HMM-based feature extraction from signals acquired by the client from the lab apparatus. The client is able to concurrently perform the following using a single 1.5GHz processor with no hyper-threading:

1. Acquire data from the DAQ buffer at 50 KHz sample rate.
2. Update the client GUI screen with the acquired data – though a 10x data decimation was required in order to avoid DAQ overflow due to delays in reading from the DAQ buffer
3. Perform tFSA logic to screen for signals resulting from molecular capture events at the nanopore channel
4. Send capture-signals as long as 100ms in duration at a rate 10 per second to a waiting 1.5GHz processor, Linux-based TCP/IP server for HMM-based feature extraction
5. Receive extracted HMM features from the TCP/IP server

6. Compute the classification of the HMM features with tolerance via a binary SVM previously trained on 9gc and 9at bphp signals.
7. According to the user's preset preference, issue a control signal to the DAQ resulting in ejection of the undesirable molecule so determined from the nanopore channel site.

5.2.5 HMM with Duration Experimental Tests

Results for the implicit HMMwD are presented below in Figure 15 - Figure 22, along with the comparative results from an explicit HMMwD (for a 2-state system) included for comparative analysis (a detailed analysis with the explicit HMMwD is given in [**51**]). The explicit 2-state HMMwD has 2n states, where n is the number of dwell bins in the quantization of the dwell-time distribution. The computations needed scale quadraticly in n. So, for n=1000, the explicit HMMwD can take 1,000,000-times longer to compute than the HMMwD described here (or the conventional HMM). Performance of the new HMMwD is given in comparison to the conventional HMM also shown in the figures. The simplicity of the binary signal case actually provides a challenging case for differentiating the performance of the HMMwD methods from the conventional HMM, so the preliminary results shown here show promise for the overall validity and utility of the method.

In another test, we compare the performance of our 2-state HMMwD to explicit HMMwD and conventional HMM (see Table 19 and Table 20 below), with varying size data generation and length distribution representation. The comparative scores shown are not optimised for use of the internal HMMwD, but are meant to explore the validity of the model and the behaviour of the prediction accuracy of the different methods – which consistently appear in the graphs in order of increasing performance as conventional HMM (good), implicit HMMwD (better), explicit HMMwD (best) – but the latter at much greater computational cost. The results clearly demonstrate the superior performance of the HMM-with-duration over its simpler, HMM without Duration, formulation. With use of the EVA-projection method this affords a robust means to obtain kinetic feature extraction. The HMM with duration is, thus, critical for accurate kinetic feature extraction, via a pairing of the HMM-with-Duration stabilization with EVA-projection [**19**].

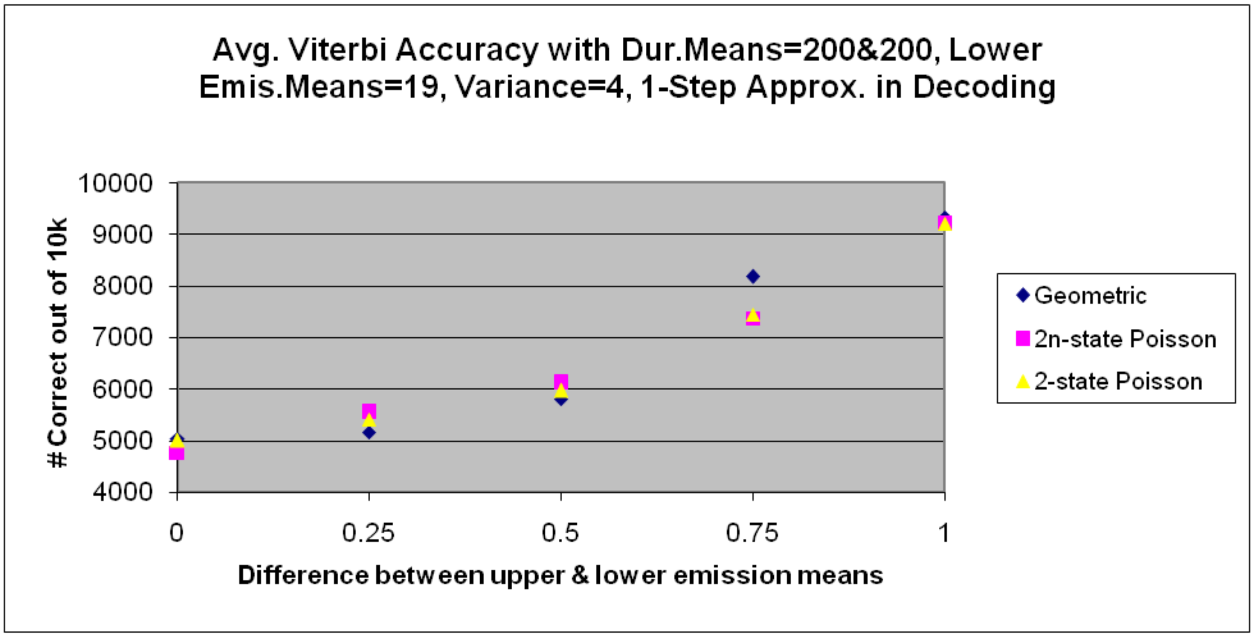


Figure 15 Average accuracy of Viterbi decoding using 1-step approximation of 1k-step generating duration model

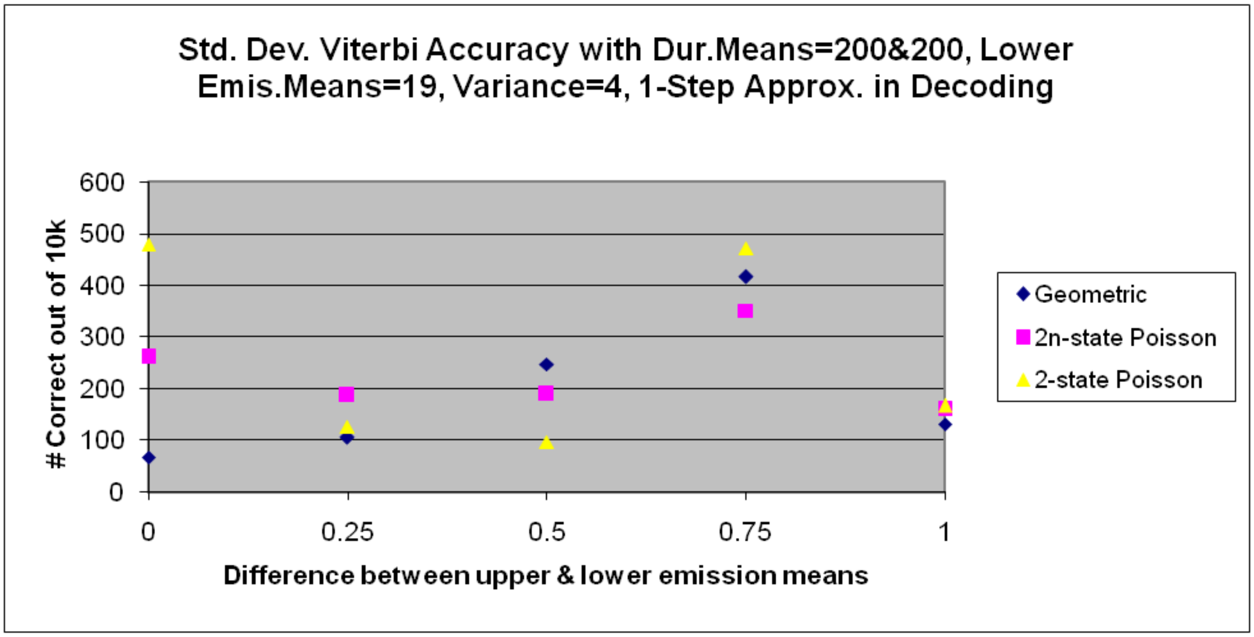


Figure 16 Standard deviation of accuracy of Viterbi decoding using 1-step approximation of 1k-step generating duration model

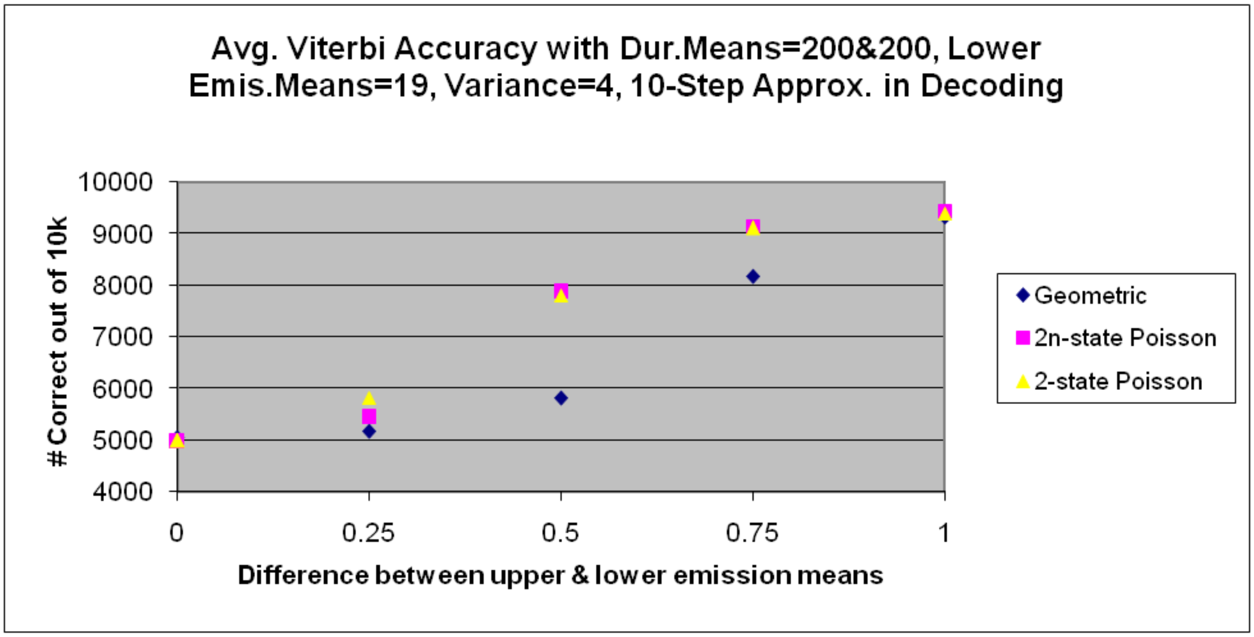


Figure 17 Average accuracy of Viterbi decoding using 10-step approximation of 1k-step generating duration model

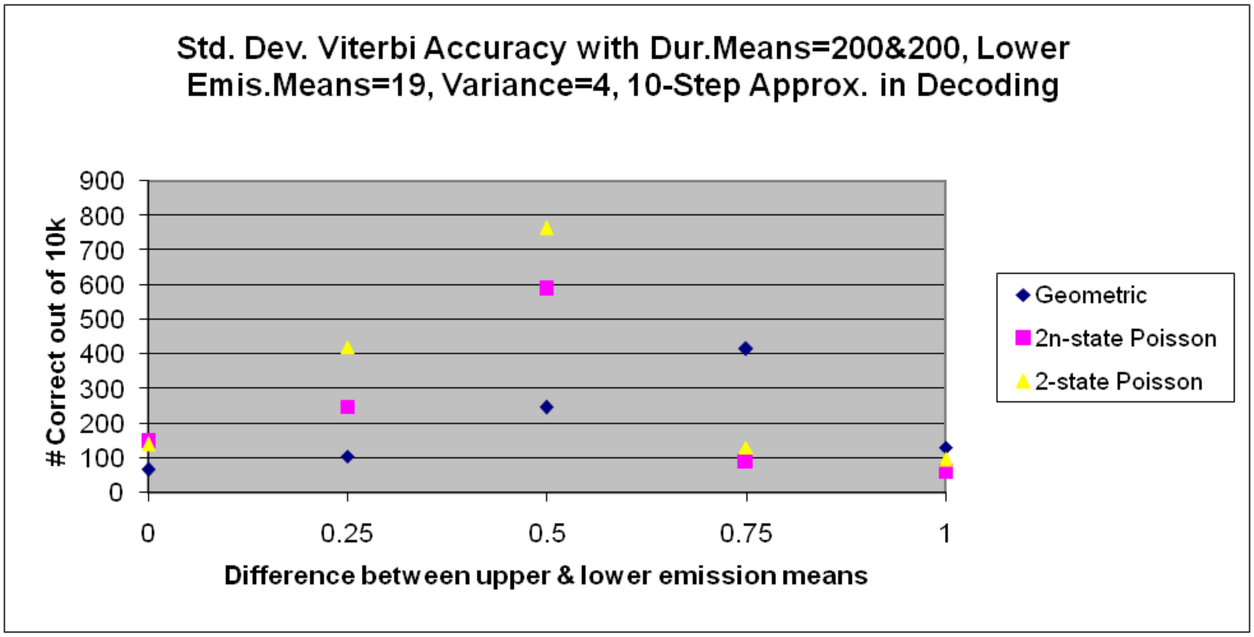


Figure 18 Standard deviation of accuracy of Viterbi decoding using 10-step approximation of 1k-step generating duration model

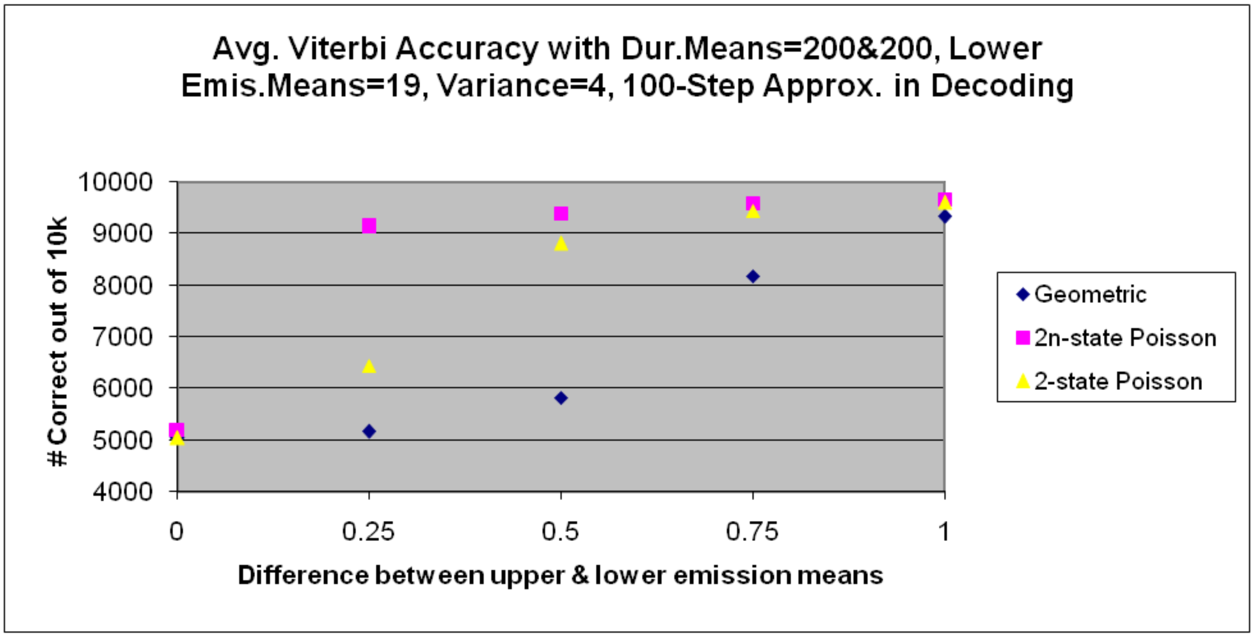


Figure 19 Average accuracy of Viterbi decoding using 100-step approximation of 1k-step generating duration model

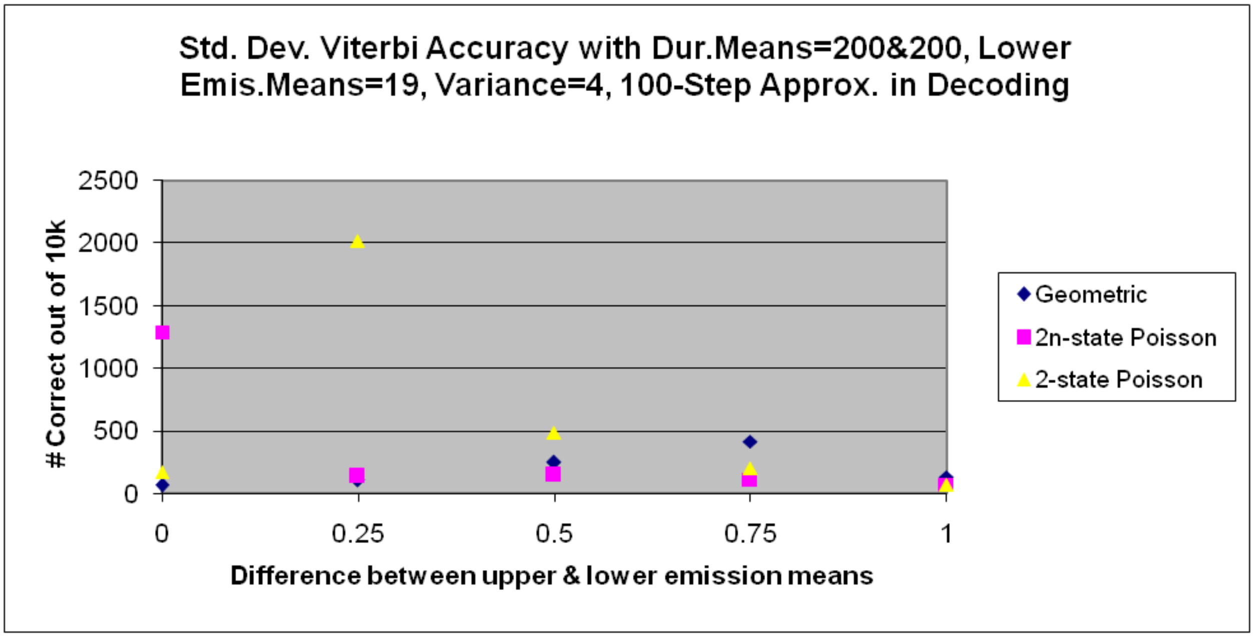


Figure 20 Standard deviation of accuracy of Viterbi decoding using 100-step approximation of 1k-step generating duration model

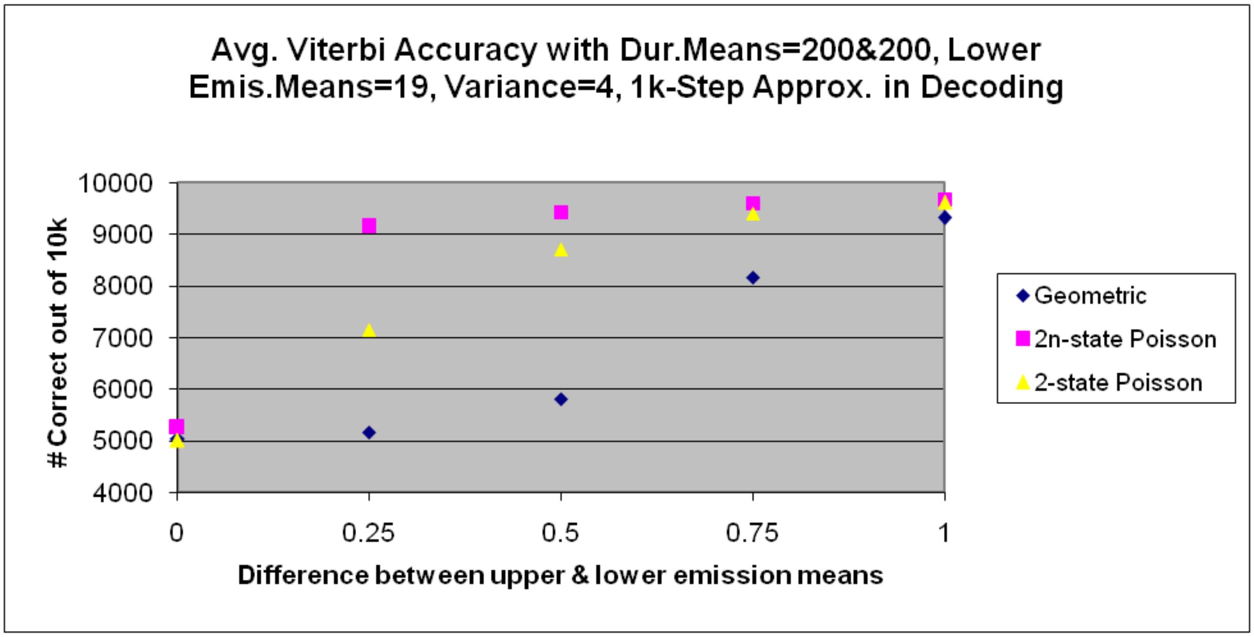


Figure 21 Average accuracy of Viterbi decoding using *same* 1k-step decoding duration model as 1k-step generating duration model

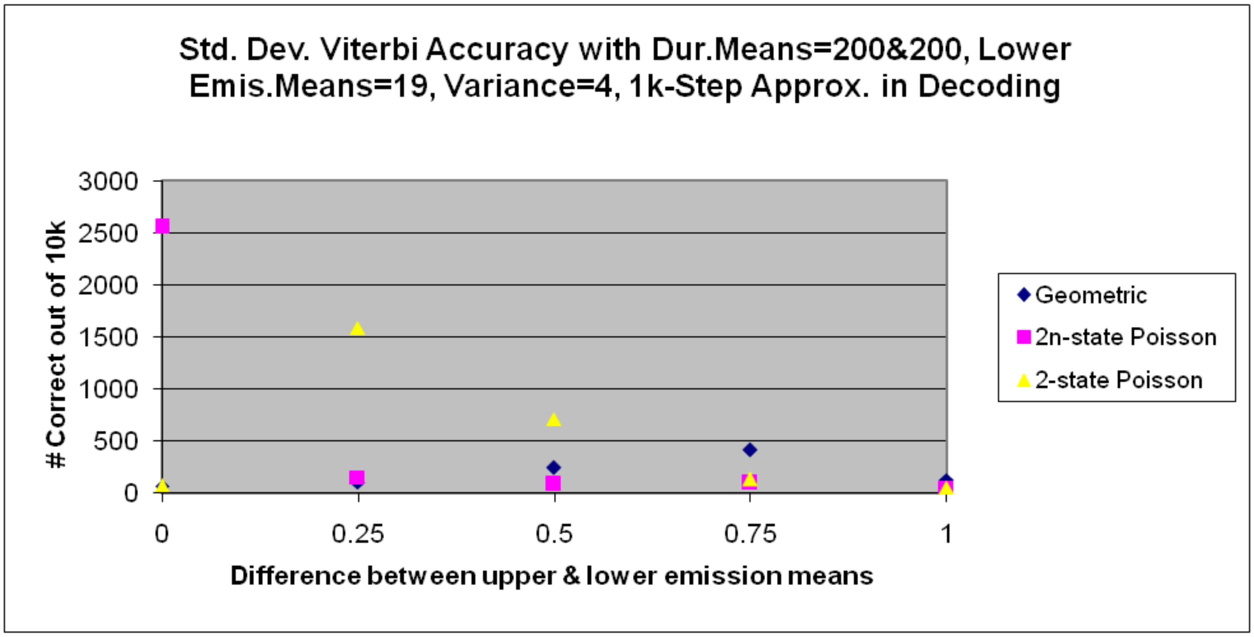


Figure 22 Standard deviation of accuracy of Viterbi decoding using *same* 1k-step decoding duration model as 1k-step generating duration model

In Table 19 below, for all row entries the average dwell time of both the upper and lower signal levels increases proportionately with bin-count. Thus the resolution of the duration distribution for both the signal-generating and the signal-decoding HMM's are identical for all row entries. Upper and lower mean durations are .2 and .6 times that of the bin count, respectively. Upper and lower mean emissions are 50.0 and 45.0, respectively, whereas the emission variance for both upper and lower is 20.0. In other tests (see Figure 15 - Figure 22 above), the decoding duration distribution is an increasingly refined step function approximation of the generating duration distribution. For the data shown in this table, both upper and lower dwell-time distributions are 2-component beta-mixtures that are monotonically decreasing. The numbers shown represent the accuracy of Viterbi decoding out of an input signal 1 M samples in length.

Table 19 Compared accuracy of Viterbi decoding for 2-component beta-mixture duration models

Bin #	2-state Geometric HMM	2n-state Explicit HMMwD	2-state Implicit HMMwD
10	743876	832514	756765
100	951916	960272	953151
1000	995388	995408	995381

In Table 20 below, as with Table 19 above, for all row entries the average dwell time of both the upper and lower signal levels increases proportionately with bin-count. Thus the resolution of the duration distribution for both the signal-generating and the signal-decoding HMM's are identical for all row entries. Upper and lower mean durations are .2 and .6 times that of the bin count, respectively. Upper and lower mean emissions are 50.0 and 45.0, respectively, whereas the emission variance for both upper and lower is 20.0. In other tests (see figures above), the decoding duration distribution is an increasingly refined step function approximation of the generating duration distribution. For the data shown in this table, both upper and lower durations are 3-component beta-mixtures with humps in the aggregate.

Table 20 Compared accuracy of Viterbi decoding for 3-component beta-mixture duration models

Bin #	2-state Geometric HMM	2n-state Explicit HMMwD	2-state Implicit HMMwD
10	720050	795723	741579
100	947062	957753	948505
1000	994978	995013	995005

5.3 Controlled Acquisition via SVM

This and the following section are taken from [**21**] for a more detailed perspective of a first functioning prototype for controlled cheminformatic signal acquisition using SVM-base signal classification in real time.

Single biopolymers (DNA, RNA, or polypeptide) can be examined using an alpha-hemolysin channel detector. When a biopolymer is present in an alpha-hemolysin channel it can produce a highly structured ionic current blockade pattern, where the lifetimes at various sub-blockade levels reveal information about the kinetics of the biopolymer (resulting from interactions with the channel, other molecules, or from conformational changes). Here we describe integration of LabVIEW/LabWindows [**53**] automation capabilities with our “in-house” Channel Current Cheminformatics (CCC) software. Data acquired with LabVIEW/LabWindows is passed to the CCC software, on a streaming real time basis, for analysis and classification. The classification results are then quickly returned to the LabVIEW/LabWindows automation software for experimental feedback control. The prototype signal processing architecture is designed to rapidly extract useful information from noisy blockade signals. A fast, specially designed, generic Hidden Markov Model is used for the channel current feature extraction. Classification of feature vectors obtained by the HMM (for each individual blockade event) is then done by Support Vector Machine, an approach which automatically provides a confidence measure on each classification. SVMs are fast discriminators (once trained), for which strong discrimination is possible without the over-fitting complications common to neural net discriminators. SVMs also offer numerous multiclass and clustering formulations for managing complicated channel current signals (or for analyzing a stochastic sequential signal in general).

Real-time control of a nanopore detector, based on live, streaming measurements, and sufficiently fast pattern recognition identification of any blockading ("captured") analyte, holds great promise for single molecule experiments. Real-time *sampling* control of a nanopore detector, alone, can boost nanopore detector sampling productivity by orders of magnitude, depending on the mix of desirable signal classes vs. undesirable in the data being analyzed, and an example of such an experiment is the focus of the proof-of-principle experiment performed here. If there is a 1 to 100 ratio of desirable to undesirable, for example, then one obtains desirable signal sampling only about 1% of the time with a passive sampling system. With pattern recognition informed sampling this can potentially be changed to desirable signal sampling almost 99% of the time. In a real-time setting the challenge is to perform the HMM feature extraction sufficiently quickly (whereas the SVM is trained off-line, so operates very quickly on-line). In this work we show that this can be accomplished, with pattern recognition used to identify DNA molecules within the first few hundred milliseconds of their blockade of the detector channel.

We establish a real-time experimental setup for PRI sampling control via integration of LabWindows automation capabilities with our "in-house" Channel Current Cheminformatics (CCC) methods. Data acquired with LabWindows is passed to a network of CCC software clients, on a streaming real time basis, for analysis and classification. The classification results are then quickly returned to the LabWindows automation software for experimental feedback control.

5.4 Pattern Recognition Informed Feedback via LabWindows Automation

A typical blockade signal has phases with stationary statistics, which reveals information about the kinetics of the biopolymer resulting from interactions with surroundings, or from undergoing conformational changes. LabVIEW Automation software is used to help manage the feedback linkage between patch-clamp amplifier measurements and in-house cheminformatics software. This has been used to demonstrate molecular identification in the first 100 ms of capture, with return of classification information to the control of the amplifier – for voltage-controlled sample ejection if desired.

The LabWindows Server initiates the distributed CCC computations by sending data to a cluster of Linux Clients via a TCP/IP channel. The Linux clients run the expensive HMM analysis as distributed processes (similarly for off-line SVM training). The sample classification is then used by the Server to provide feedback to the nanopore apparatus to increase the effective sampling time on the molecules of interest (meant to boost nanopore detector productivity by magnitudes, as described in the Background [**21**]).

6 Discussion and Future Work

6.1 Conclusions from HOHMM Results

For reference, in the following discussion, the top performing results from the evaluations performed in [**17**] and [**4**] are included in Table 21 and Table 22 below, including values for the (nucleotide) base level accuracy converted from the AC measurement to the simpler E[(sn+sp)/2]. (See **3.2.1 Accuracy at the Base or Nucleotide Level**, page 46.)

Table 23 shows the top results of the HOHMM for the data sets and parameter values tested in this effort, including in each case the optimum values for the parameters M and F, though optimum values for L and R are not shown. Recall that the method of measurement used in this effort differs somewhat from that of the cited evaluations. (See **3.2.3 Expectation of the Measure vs. Measure of the Expectation**, page 47.) For additional reference, Table 24 shows the maximum accuracy specifically for the ALLSEQ data set at both the base and exon levels using the method of measurement in the cited evaluations.

Table 21 Top 2 performers in the evaluation by Burset and Guigo testing with ALLSEQ

Software Name	Nucleotide level				Full Exon Level		
	E[sn]	E[sp]	AC	E[(sn+sp)/2]	E[SN]	E[SP]	E[(SN+SP)/2]
FGENEH	0.77	0.88	0.78±0.26	0.825	0.61	0.64	0.64±0.33
GeneID+	0.91	0.91	0.88±0.16	0.91	0.73	0.70	0.71±0.29

Table 22 Top 3 performers in the evaluation by Rogic, et al., testing with HMR195

Software Name	Nucleotide level				Full Exon Level		
	E[sn]	E[sp]	AC	E[(sn+sp)/2]	E[SN]	E[SP]	E[(SN+SP)/2]
Genie	0.91	0.90	0.89±0.16	0.905	0.71	0.70	0.71±0.30
Genscan	0.95	0.90	0.91±0.12	0.925	0.70	0.70	0.70±0.32
HMMgene	0.93	0.93	0.91±0.13	0.93	0.76	0.77	0.76±0.30

Table 23 Maximum accuracy of HOHMM for the parameter values tested

Data set Name	Nucleotide level					Full Exon Level				
	sn	sp	(sn+sp)/2	M	F	SN	SP	(SN+SP)/2	M	F
entire_1.	1	0.999	0.999	5	12	1	0.993	0.997	5	12
ALLSEQ	0.978	0.954	0.966	8	4	0.919	0.803	0.861	8	12
Chr. I-V	0.938	0.864	0.901	5	12	0.775	0.711	0.743	2	20

Table 24 Maximum accuracy of HOHMM for ALLSEQ using the cited method of measurement

Data set Name	Nucleotide level					Full Exon Level				
	E[sn]	E[sp]	E[(sn+sp)/2]	M	F	E[SN]	E[SP]	E[(SN+SP)/2]	M	F
ALLSEQ	0.987	0.961	0.974	8	12	0.917	0.847	0.882	8	12

Thus in the case of the ALLSEQ data set, the HOHMM appears to perform better with the cited method of measure than the method used otherwise in this effort. Recall that the cited measure weights those sequences with fewer and shorter exons more heavily in that it is more forgiving of prediction errors in sequences with more and longer exons – or greater genomic complexity. Hence, the difference between the two measures of the same HOHMM outcome serves as indication that those sequences in the ALLSEQ data set on average with greater genomic complexity pose more of a challenge than the other sequences in the same data set.

This effectively places additional conditioning on the ALLSEQ data set, which could have contributed to the authors' choice of measurement method over the method used otherwise in this effort. These two methods of measurement have been termed averaging *by gene* and *by base*, respectively [**54**]. The authors' argue in [**17**] that averaging by base "appears to have little justification from a statistical standpoint" for the following reasons.

1) Averaging by gene provides an "[u]nbiased estimation of...the expected performance of the programs when analyzing a given anonymous DNA sequence."
2) Determination of protein product similarity necessarily restricts the focus of testing to a gene-centric perspective and measurement.

Nonetheless, it appears that these arguments have been chosen to suit the context of data sets such as ALLSEQ, where all DNA sequences (not entirely anonymously) are chosen to contain only a single gene – or, in other words (with pun intended), the end justifies the means.

Moreover, in the case of Table 24 above, it is worth noting that this measure of HOHMM performance for the data set ALLSEQ has been obtained using the cited method of measurement and can now be directly compared with that of the programs in the cited evaluation [**17**] for the same data set, ALLSEQ. Thus we observe that the HOHMM's performance clearly exceeds that of the top performing program, GeneID+, of the cited evaluation by substantial margins (~7% and ~24%) at both base- and exon-levels, respectively. It should also be noted that the program cited, GeneID+, among others uses such *extrinsic* information as "amino acid similarity searches" in the process of forming its prediction, whereas the HOHMM used in this effort uses only the *intrinsic* information contained in the DNA sequence data alone.

6.2 Future Work

Within the *de facto* testing framework that has evolved from [**17**] and more so in [**4**], have promoted the co-evolution of certain properties of the competing gene finders that would not serve any useful purpose in the wider context of testing with less preconditioned, raw genomic data.

6.2.1 Advantages of Other Gene Finders

The advantages of other gene finding methods that are supported in the current testing framework are as follows.

1) They assume that a single gene exists in the window of the given test sequence. (This can be a substantial advantage in the identification of extended-length motifs.)
2) They assume that a single read/encoding direction is sufficient, with no added complexity of simultaneous modeling of reverse encodings.
3) They assume no alternative splicing, where encodings can interleave and overlap.
4) Gene finders in [**4**] incorporated *explicit* HMM-with-Duration (HMMwD) modeling at the cost of significant increase in computational time. (In [**19**], we have demonstrated an approximation (exact in the 2-state case) in order to accomplish the same modeling process. This in turn helped motivate the exact HMM with binned duration by Dr. Winters-Hilt and Zuliang Jiang [**20**]. Thus we have yet to incorporate

the HMMwD in order to gain the advantage of excluding anomalously short or weak exons or introns, as such at significantly less cost in computational time.)

5) Gene finders in [4] also used signal recognition at splice sites, etc. in order to boost performance.
6) Depending on the structure of the information used in the incorporated HMM signal sensors, some of the (other) gene finders introduce extrinsic information. (For example, see [55].)

6.2.2 Other Issues of Other Gene Finders

Other issues stemming from the above are:

1) In principal, the added structure of the testing framework cited above, promotes a restricted machine learning environment, where intervention is required at first to isolate single-gene sequences, etc.
2) The result is a restriction of evolution towards a clean formalism of identification of alternative splicing, etc.
3) An object-oriented, Perl code base has been developed by Dr. Winters-Hilt in order to extend the HOHMM developed here to handle more complex, overlapping annotations [23], but is beyond the scope of this thesis and hence not discussed any further here.

6.3 Summary

Here we provide a more detailed summary of the main contribution in this thesis. (For a summary of all contributions in this effort, see **1.4 Contributions**, page 4.) The main contribution in this thesis is a new type of generalized HMM (or HOHMM) is described. This model involves both observations and states of extended length in a generalized clique structure, where the extents of the observations and states are incorporated as parameters in the new model. This clique structure was intended to address the following 2-fold hypothesis.

1) The introduction of extended observations would take greater advantage of the information contained in the observed tables of strong, higher order, position-dependent, signal statistics in DNA sequence data taken from extended regions surrounding donor and acceptor splice sites and
2) The introduction of extended states would attain a natural boosting by repeated look-up of the tabulated statistics associated in each case with the given type of coding/non-coding (c/nc) boundary.

Moreover, an efficient implementation for the HOHMM method is proven theoretically and demonstrated experimentally. The HOHMM approach enables a stronger HMM-based framework for the identification of complex structure in stochastic sequential data. One of the main applications of the HOHMM is the identification of eukaryotic gene structure. It is shown that the performance of the HOHMM-based gene-finder is equal to that of the 3 best gene-finders in use today, GENIE, GENSCAN and HMMgene [4]. The method shown here, however, is the bare-bones HMM implementation and an early version not yet benefitting, for example, from the use of signal sensors to strengthen localized encoding information, such as splice site recognition. An SVM-based improvement has been designed by Dr. Winters-Hilt to integrate directly with the approach introduced here, as discussed in [23]). Support Vector Machines are found to outperform neural-net-based, splice site discriminators (as used in the GENIE gene finder [55]). Thus, once incorporated into a future gene-finder implementation, the approach of HOHMM with SVM, etc. is expected not simply to tie, comparatively, with the best gene finders, but significantly outperform them.

By analogy the context of this work can be described as building a chassis and engine for a faster, more powerful car. The standard work that remains to make it competitive for racing -- modified exhaust, suspension, choice of tires, airfoils, etc., is less critical, standard, and not pursued in the thesis (an area of interest for future work). The standard work that remains would include – but would not be limited to – the following:

1) pre-processing including the following:
    a. filtering raw genomic data in order to restrict minimum ORF length,
    b. genetic algorithms for optimized search for HOHMM parameters;
2) additional in-line processing such as the following:
    a. correlation of predicted genes with known post-splicing and post-transcription products,
    b. incorporate 2-state subsystems with duration modeling of introns and exons for enhanced splice site recognition,
    c. multi-track transition tracking for alternative splicing,
    d. gap interpolation for enhanced detection of motifs as well as gene starts and ends; and
3) post processing including identification of extremely short ORF lengths.

7 Bibliography

[1] Hideki Noguchi, Jungho Park, and Toshihisa Takagi, "MetaGene: prokaryotic gene finding from environmental genome shotgun sequences," *Nucleic Acids Res*, vol. 34, no. 19, pp. 5623-5630, Nov. 2006.

[2] L Taher et al., "AGenDA: homology-based gene prediction," *Bioinformatics*, vol. 19, no. 12, pp. 1575-7, Aug 12 2003.

[3] MJ van Baren, BC Koebbe, and MR Brent, "Using N-SCAN or TWINSCAN to predict gene structures in genomic DNA sequences.," *Current Protocols In Bioinformatics*, vol. Chapter 4, p. Unit 4.8, Dec 2007.

[4] Sanja Rogic, Alan K Mackworth, and B.F. Francis Ouellette, "Evaluation of Gene-Finding Programs on Mammalian Sequences," *Genome Res. 2001. 11: 817-832*, pp. 817-832, 2001.

[5] I. Dunham, N. Shimizu, B.A. Roe, S. Chissoe, and et al., "The DNA sequence of human chromosome 22," *Nature*, vol. 402, no. 6761, pp. 489-95, Dec 2 1999.

[6] GQ Hu, JT Guo, YC Liu, and H Zhu, "MetaTISA: Metagenomic Translation Initiation Site Annotator for improving gene start prediction," *Bioinformatics*, vol. 24, no. 14, pp. 1843-5, Jul 15 2009.

[7] G Wang, T Yu, and W Zhang, "WordSpy: identifying transcription factor binding motifs by building a dictionoary and learning a grammar," *Nucleic Acids Res*, vol. 33, no. Web Server issue, pp. W412-6, Jul 1 2005.

[8] MQ Zhang, "Using MZEF to find internal coding exons," *Current Protocols In Bioinformatics*, vol. Chapter 4, p. Unit 4.2, Aug 2002.

[9] S Degroeve, Y Saeys, B De Baets, P Rouzé, and Y Van de Peer, "SpliceMachine; predicting splice sites from high-dimensional local context representations," *Bioinformatics*, vol. 21, no. 8, pp. 1332-8, Apr 15 2005.

[10] RV Davuluri, "Application of FirstEF to find promoters and first exons in the human genome," *Current Protocols In Bioinformatics*, vol. Chapter 4, p. Unit 4.7, May 2003.

[11] H Liu, H Han, J Li, and L Wong, "DNAFSMiner: a web-based software toolbox to recognize two types of functional sites in DNA sequences," *Bioniformatics*, vol. 21, no. 5, pp. 671-3, Mar 1 2005.

[12] S Sonnenburg, A Zien, and G Rätsch, "ARTS: accurate recognition of transcription starts in human," *Bioniformatics*, vol. 22, no. 14, pp. e472-80, Jul 15 2006.

[13] Sören Sonnenburg, Gabriele Schweikert, Petra Philips, and Jonas Behr, "Accurate splice site prediction using support vector machines," *BMC Bioinformatics*, vol. 8 (Suppl 10), no. S7, Dec 21 2007.

[14] N Bellora, D Farré, and Albà M Mar, "PEAKS: identification of regulatory motifs by their position in DNA sequences," *Bioinformatics*, vol. 23, no. 2, pp. 243-4, Jan 15 2007.

[15] Jagath C. Rajapakse and Loi Sy Ho, "Markov Encoding for Detecting Signals in Genomic Sequences," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) archive*, vol. 2, no. 2, pp. 131-42, Apr 2005.

[16] Enrique M. Muro et al., "Identification of gene 3 ' ends by automated EST cluster analysis," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 51, pp. 20286-20290, Dec 23 2008.

[17] Moises Burset and Roderic Guigo, "Evaluation of Gene Structure Prediction Programs," *Genomics*, vol. 34, pp. 353-367, 1996.

[18] (2009, March) WormBase web site, http://www.wormbase.org, release WS200, date 20 Mar 2009. [Online]. http://ws200.wormbase.org/

[19] Stephen Winters-Hilt and Carl Baribault, "A novel, fast, HMM-with-Duration implementation – for application with a new, pattern recognition informed, nanopore detector," in *Proceedings of the Fourth Annual MCBIOS Conference. Computational Frontiers in Biomedicine. BMC Bioinformatics 2007, 8(Suppl 7):S19. (http://www.biomedcentral.com/1471-2105/8/S7/S19)*, 1 November 2007.

[20] Stephen Winters-Hilt and Zuliang Jiang, "TBD," *(Submitted for publication)*, 2009.

[21] Stephen Winters-Hilt et al., "Cheminformatics methods for novel nanopore analysis of HIV DNA termini," in *Third Annual MCBIOS Conference. Bioinformatics: A Calculated Discovery. BMC Bioinformatics 2006, 7(Suppl 2):S22. (http://www.biomedcentral.com/1471-2105/7/S2/S22)*, 26 September 2006.

[22] A.M. Eren et al., "Pattern Recognition Informed Feedback for Nanopore Detector Cheminformatics," *Submitted for publication*.

[23] Stephen Winters-Hilt, "Hidden Markov Model Variants and their Application," in *Third Annual MCBIOS Conference. Bioinformatics: A Calculated Discovery*, 26 September, 2006, pp. BMC Bioinformatics 2006, Volume 7(Suppl 2):S14.

[24] Stephen Winters-Hilt, "Personal Notes,".

[25] Stephen Winters-Hilt and Brian Roux, "Hybrid MM/SVM structural sensors for stochastic sequential data," in *Proceedings of the Fifth Annual MCBIOS Conference. Systems Biology: Bridging the Omics. BMC Bioinformatics 2008, 9(Suppl 9):S12. (http://www.biomedcentral.com/1471-2105/9/S9/S12)*, 12 August 2008.

[26] Timo Koski, *Hidden Markov Models for Bioinformatics*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

[27] Canonical Ltd. (2009, October) ubuntu. [Online]. http://www.ubuntu.com/

[28] Louisiana Optical Network Initiative. LONI Systems. [Online]. http://www.loni.org/systems/

[29] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. New York: John Wiley & Sons, Inc., 2001.

[30] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory, Vol 47, No. 2, February 2001*, pp. 498-519, 2001.

[31] J.A. Du Preez and D.M. Weber, "High-order hidden Markov modelling," in *Communications and Signal Processing, 1998. COMSIG '98. Proceedings of the 1998 South African Symposium on*, University of Cape Town, Rondebosch, 7-8 Sept. 1998, pp. 197-202.

[32] R. Durbin, Eddy S., Krogh A., and Mitchison G., *Biological Sequence Analysis, Probabilistic models of proteins and nucleic acids*, 82005th ed. Cambridge: Cambridge University Press, 1998.

[33] John E. Savage, *Models of Computation, Exploring the Power of Computing*. Reading, MA: Addison Wesley Longman, Inc., 2000.

[34] Dan Sugalski. perlthrtut. [Online]. http://perldoc.perl.org/perlthrtut.html#What-Is-A-Thread-

Anyway?

[35] Mathieu Fourment and Michael R Gillings, "A comparison of common programming languages used in bioinformatics," *BMC Bioinformatics 2008; Vol. 9, pp. 82.*, February 5 2008.

[36] nexB, Inc. EasyEclipse for LAMP. [Online]. http://www.easyeclipse.org/site/distributions/lamp.html

[37] nexB, Inc. EasyEclipse Distributions. [Online]. http://www.easyeclipse.org/site/distributions/index.html

[38] The GNU Project. GCC, the GNU Compiler Collection. [Online]. http://gcc.gnu.org/

[39] Thomas Williams and Colin Kelley. gnuplot homepage (sic). [Online]. http://www.gnuplot.info/

[40] Pang-ning Tan, Michael Steinbach, and Vipin and Kumar, *Introduction to Data Mining*. Boston: Pearson Education, Inc., 2006.

[41] R. Staden, "A new computer method for the storage and manipulation of DNA gel reading data," *Nucleic Acids Res. 8*, pp. 3673-3694, 1980.

[42] R. Staden. What is a contig? [Online]. http://staden.sourceforge.net/contig.html

[43] Paul Davis. Personal Emails. [Online]. pad@sanger.ac.uk

[44] Alex Liu, (Master's Thesis), Month Day, Year.

[45] Genome BioInformatics Research Laboratory. (2005, Oct.) Resources & Datasets. [Online]. http://genome.imim.es/databases/genomics96/index.html

[46] National Center for Biotechnology Information (NCBI). Entrez Nucleotide. [Online]. http://www.ncbi.nlm.nih.gov/sites/entrez?db=nucleotide

[47] J.W. Fickett and C.-S. Tung, "Assessment of protein coding measures.," *Nucleic Acids Res. 20:6441–6450*, pp. 6441-6450, 1992.

[48] Sanja Rogic. HMR195 dataset. [Online]. http://www.cs.ubc.ca/~rogic/evaluation/dataset.html

[49] E.E. Snyder and Stormo G.D., "Identification of protein coding regions in genomic DNA.," *J. Mol. Biol. 248:1-18*, pp. 1-18, 1995.

[50] James W. Fickett, "The gene identification problem: An overview for developers," *Computers Chem. Vol 20, No. 1*, pp. 103-118, 1996. [Online]. http://www.nslij-genetics.org/gene/1996.html

[51] Alexander Churbanov, Carl Baribault, and Stephen Winters-Hilt, "Duration learning for analysis of nanopore ionic current blockades," in *Proceedings of the Fourth Annual MCBIOS Conference. Computational Frontiers in Biomedicine. BMC Bioinformatics 2007, 8(Suppl 7):S14. (http://www.biomedcentral.com/1471-2105/8/S7/S14)*, 1 November 2007.

[52] Wikimedia Foundation, Inc. Data acquisition. [Online]. http://en.wikipedia.org/wiki/Data_acquisition

[53] National Instruments, Inc. NI LabVIEW. [Online]. http://www.ni.com/labview

[54] S. Dong and D.B. Searls, "Gene structure prediction by linguistic methods," *Genomics*, vol. 23, pp. 540-551, 1994.

[55] Martin G. Reese, Frank H. Eeckman, David Kulp, and David Haussler, "Improved splice site detection in Genie," *RECOMB '97: Proceedings of the first annual international*

conference on Computational molecular biology, pp. 232-240, January, 1997.

[56] R. Iqbal, M. Landry, and Winters-Hilt S, "DNA Molecule Classification Using Feature Primitives," *BMC Bioinformatics, 7 Suppl 2: S15. [8 pages]* , pp. 1-8, 2006, Sept. 26.

[57] S. Winters-Hilt, A. Yelundur, C McChesney, and M. Landry, "Support Vector Machine Implementations for Classification & Clustering," *BMC Bioinformatics, 7 Suppl 2: S4*, pp. 1-18, 2006, Sept. 26.

[58] Stephen Winters-Hilt, "Nanopore Detector based analysis of single-molecule conformational kinetics and binding interactions," in *Third Annual MCBIOS Conference. Bioinformatics: A Calculated Discovery. BMC Bioinformatics 2006, 7(Suppl 2):S21. (http://www.biomedcentral.com/1471-2105/7/S2/S21)*, 26 September 2006.

[59] David J. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge, UK: Cambridge University Press, 2004.

[60] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, pp. 4-16, January 1986.

[61] Dan E. Krane and Michael L. Raymer, *Fundamental Concepts of Bioinformatics*. San Francisco: Benjamin Cummings, 2003.

[62] S. Winters-Hilt and Z. Jiang, "HMMD with position and zone dependent emissions," *Paper in preparation*, 2009.

[63] Wikipedia. (2009, November) Gene prediction. [Online]. http://en.wikipedia.org/wiki/Gene_prediction

[64] Wikipedia. (2009, November) Pseudogene. [Online]. http://en.wikipedia.org/wiki/Pseudogene#cite_note-pmid3909943-0

[65] J Besemer and M Borodovsky, "GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses," *Nucleic Acids Research*, vol. 33, no. Web Server Issue, pp. W451-4, Jul 1 2005.

[66] JH Do and DK Choi, "Computational approaches to gene prediction," *Journal Of Microbiology*, vol. 44, no. 2, pp. 137-44, Apr 2006.

[67] M Stanke and B Morgenstern, "AUGUSTUS: a web server for gene prediction in eukaryotes that allows user-defined constraints," *Nucleic Acids Research*, vol. 33, no. Web Server Issue, pp. W465-7, Jul 1 2005.

[68] JE Allen, WH Majoros, M Pertea, and SL Salzberg, , vol. 7 Suppl 1, pp. S9.1-13, Aug 2006.

[69] M Borodovsky, R Mills, J Besemer, and A Lomsadze, "Prokaryotic gene prediction using GeneMark and GeneMark.hmm.," *Current Protocols In Bioinformatics*, vol. Chapter 4, p. Unit 4.5, May 2003.

[70] M Borodovsky, A Lomsadze, N Ivanov, and R Mills, "Eukaryotic gene prediction using GeneMark.hmm," *Current Protocols In Bioinformatics*, vol. Chapter 4, p. Unit 4.6, May 2003.

[71] WH Majoros, M Pertea, and SL Salzberg, "TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders," *Bioinformatics*, vol. 20, no. 16, pp. 2878-9, Nov 1 2004.

[72] MJ de Hoon, S Imoto, J Nolan, and S Miyano, "Open source clustering software," *Bioinformatics*, vol. 20, no. 9, pp. 1453-4, Jun 12 2004.

[73] D Shinozaki, T Akutsu, and O Maruyama, "Finding optimal degenerate patterns in DNA

sequences," *Bioinformatics*, vol. 19 Suppl 2, pp. ii206-14, Oct 2003.

[74] T Frickey and G Weiller, "Mclip: motif detection based on cliques of gapped local profile-to-profile alignments," *Bioinformatics*, vol. 23, no. 4, pp. 502-3, Feb 15 1007.

[75] Xin He, Xu Ling, and Saurabh Sinha, "Alignment and Prediction of cis-Regulatory Modules Based on a Probabilistic Model of Evolution," *PLoS Computational Biology*, vol. 5, no. 3, p. Article No. e1000299, Mar 2009.

[76] Gautier Koscielny et al., "ASTD: The Alternative Splicing and Transcript Diversity database," *Genomics*, vol. 93, no. 3, pp. 213-220, Mar 2009.

[77] Catherine Mathé, Marie-France Sagot, Thomas Schiex, and Pierre Rouzé, "Current methods of gene prediction, their strengths and weaknesses," *Nucleic Acids Research*, vol. 30, no. 19, pp. 4103-4117, August 2002.

[78] Ian Korf, "Gene finding in novel genomes," *BMC Bioinformatics*, vol. 5, no. 59, May 14 2004.

[79] Mahmood Akhtar, Eliathamby Ambikairajah, and Julien Epps, "Digital Signal Processing Techniques for Gene Finding in Eukaryotes," in *Image and Signal Processing, Lecture Notes in Computer Science; Volume 5099*. Berlin / Heidelberg: Springer, 2008, pp. 144-152.

[80] Sourav Chatterji and Lior Pachter, "Reference based annotation with GeneMapper," *Genome Biology*, vol. 7, no. R29, Apr 5 2006.

8 Appendix

8.1 Detailed Results for Entire_1_contig of *C. elegans*

The graphs in Figure 23 - Figure 42 depict the detailed results of prediction accuracy of the HOHMM on the data set entire_1_contig for *C. elegans* using the HOHMM parameters values M (=0, 2, 5, 8), F(=4, 6, 8, 10, 12), L(=0-9) and R(=0-9). In each figure, the smoothness of each plotted surface serves as an indication of the stability of the HOHMM over the entire range of parameter values used.

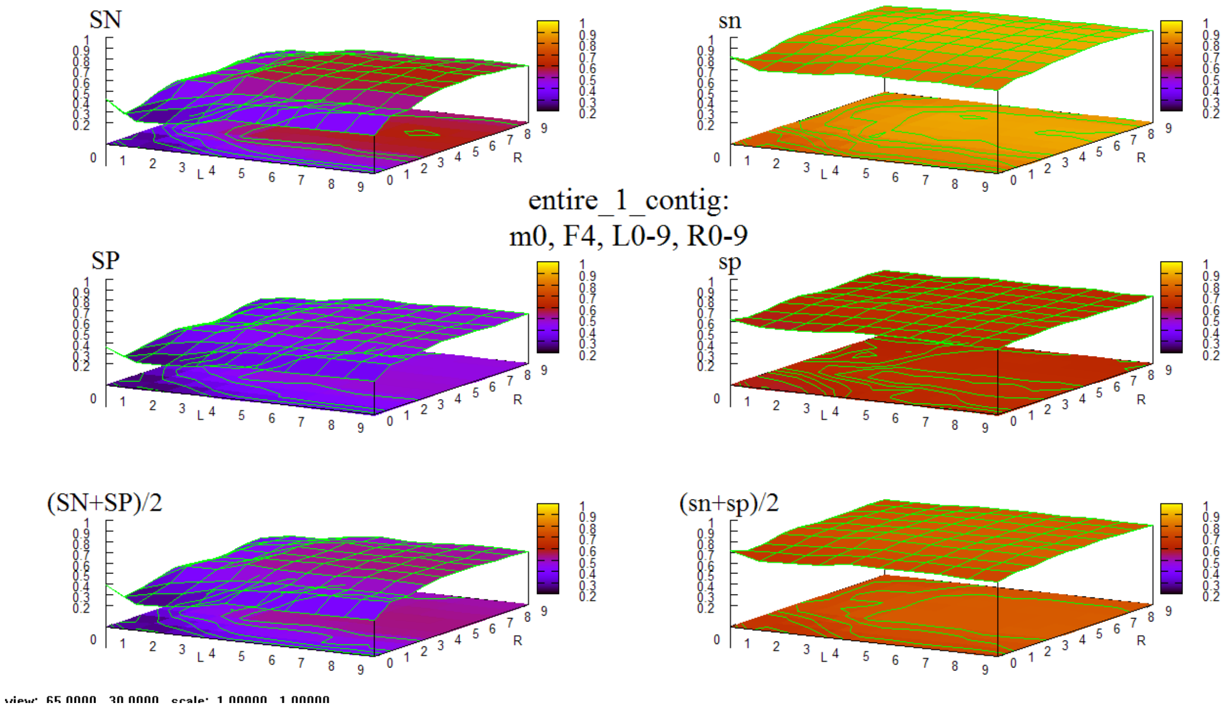


Figure 23 HOHMM performance for data entire_1_contig using m=0, F=4, L=0-9,R=0-9

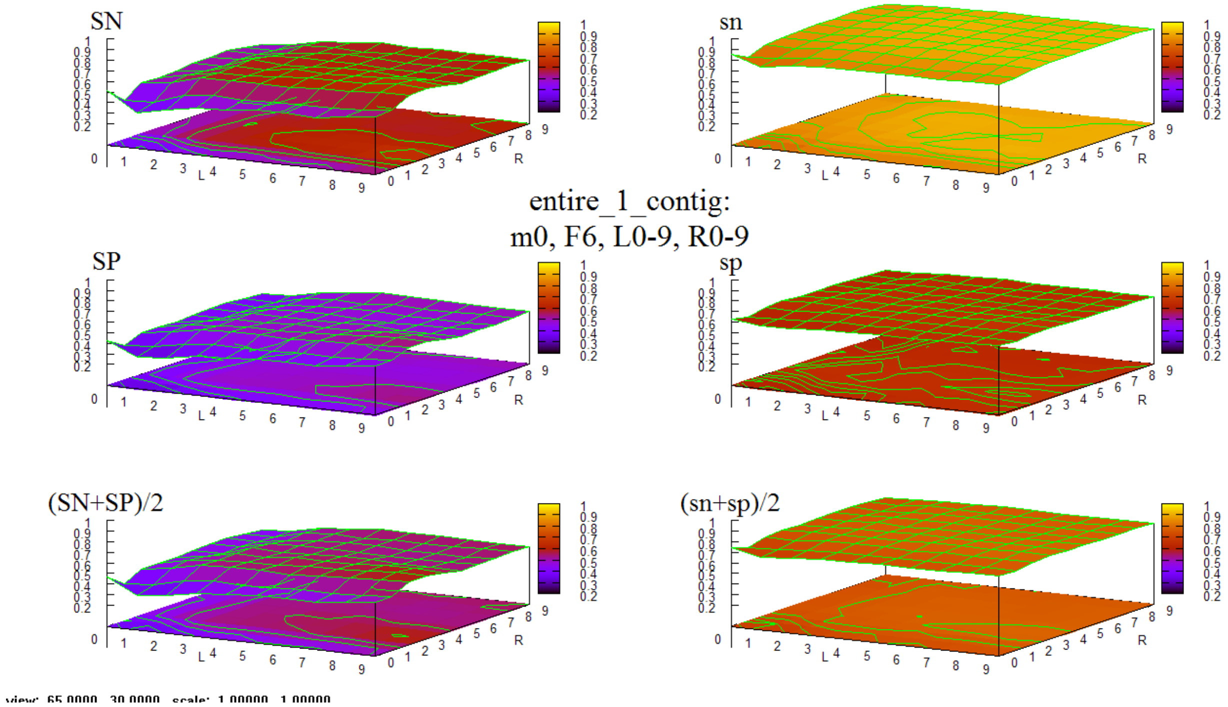


Figure 24 HOHMM performance for data entire_1_contig using m=0, F=6, L=0-9,R=0-9

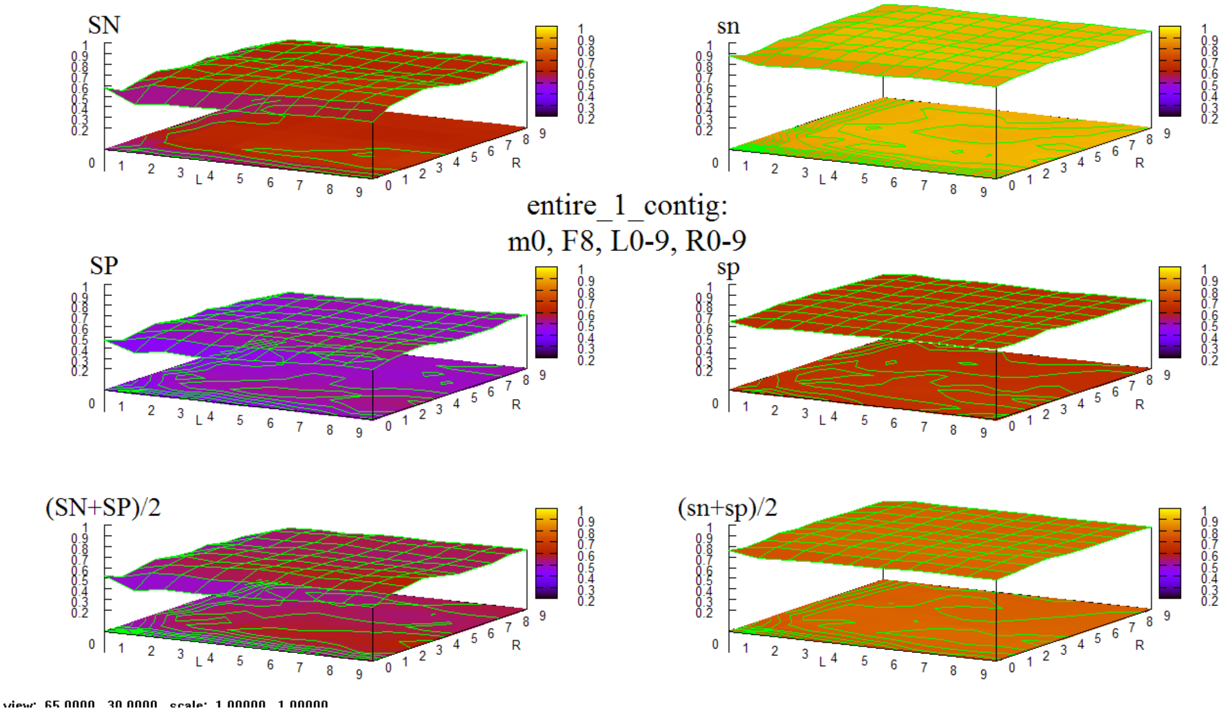


Figure 25 HOHMM performance for data entire_1_contig using m=0, F=8, L=0-9,R=0-9

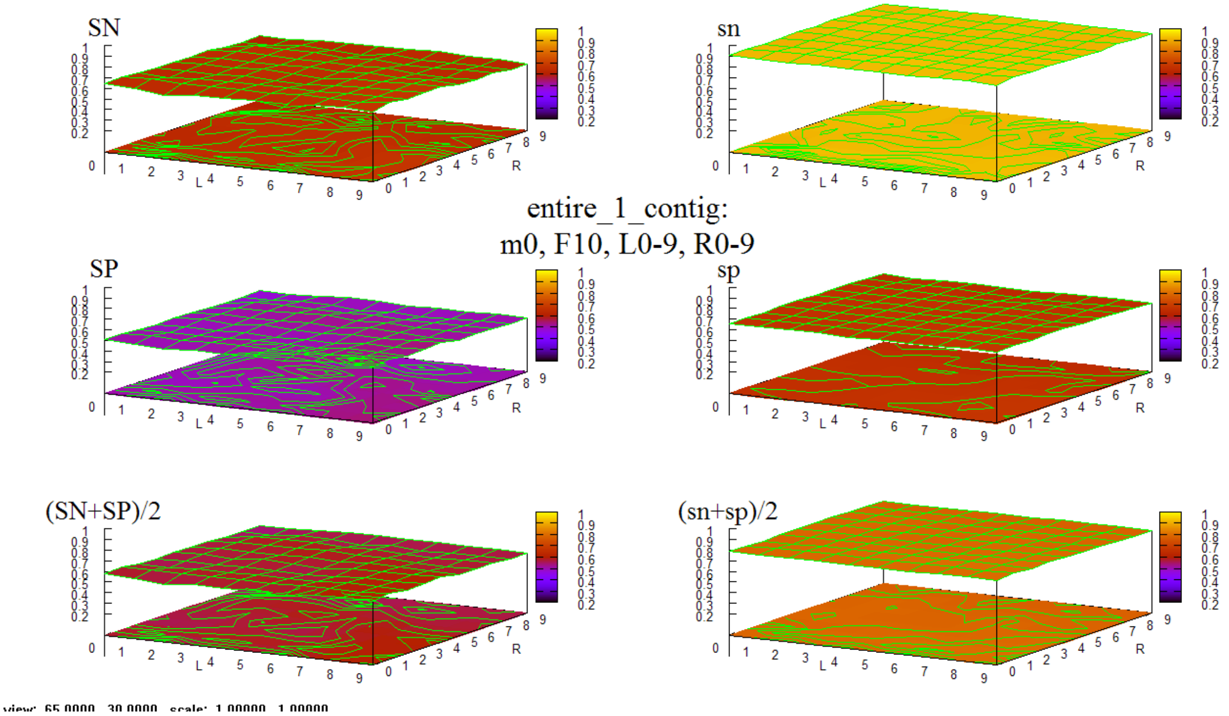


Figure 26 HOHMM performance for data entire_1_contig using m=0, F=10, L=0-9,R=0-9

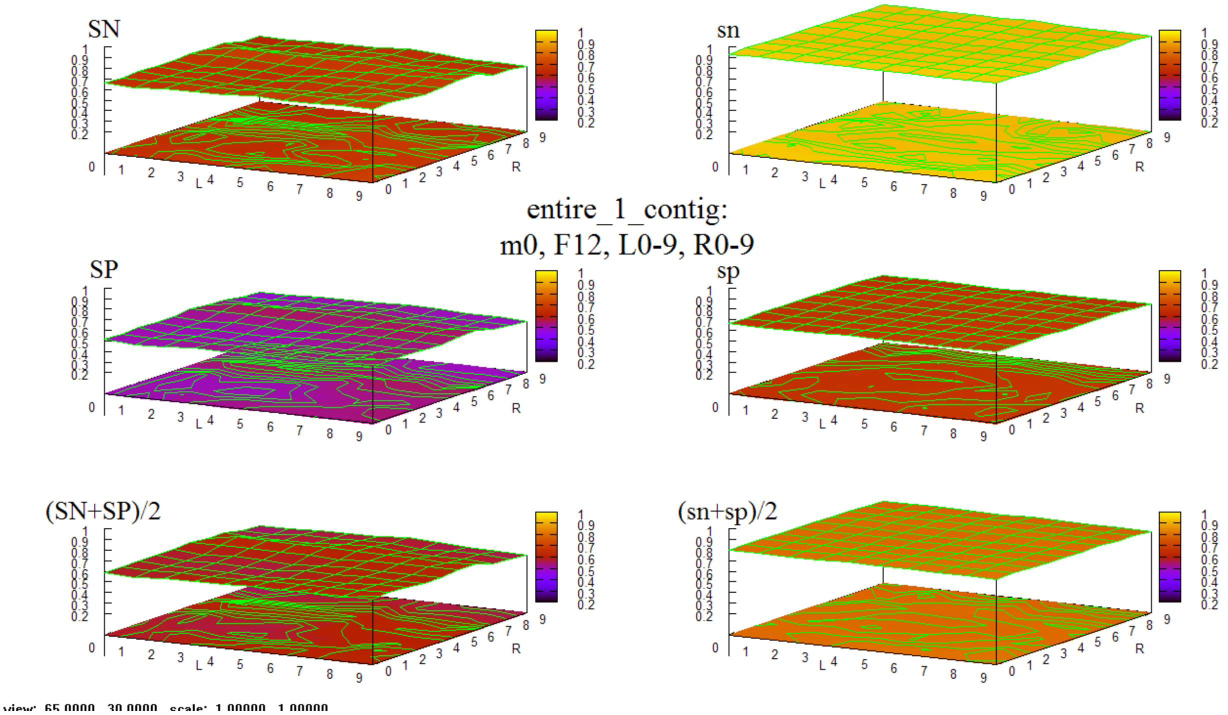


Figure 27 HOHMM performance for data entire_1_contig using m=0, F=12, L=0-9,R=0-9

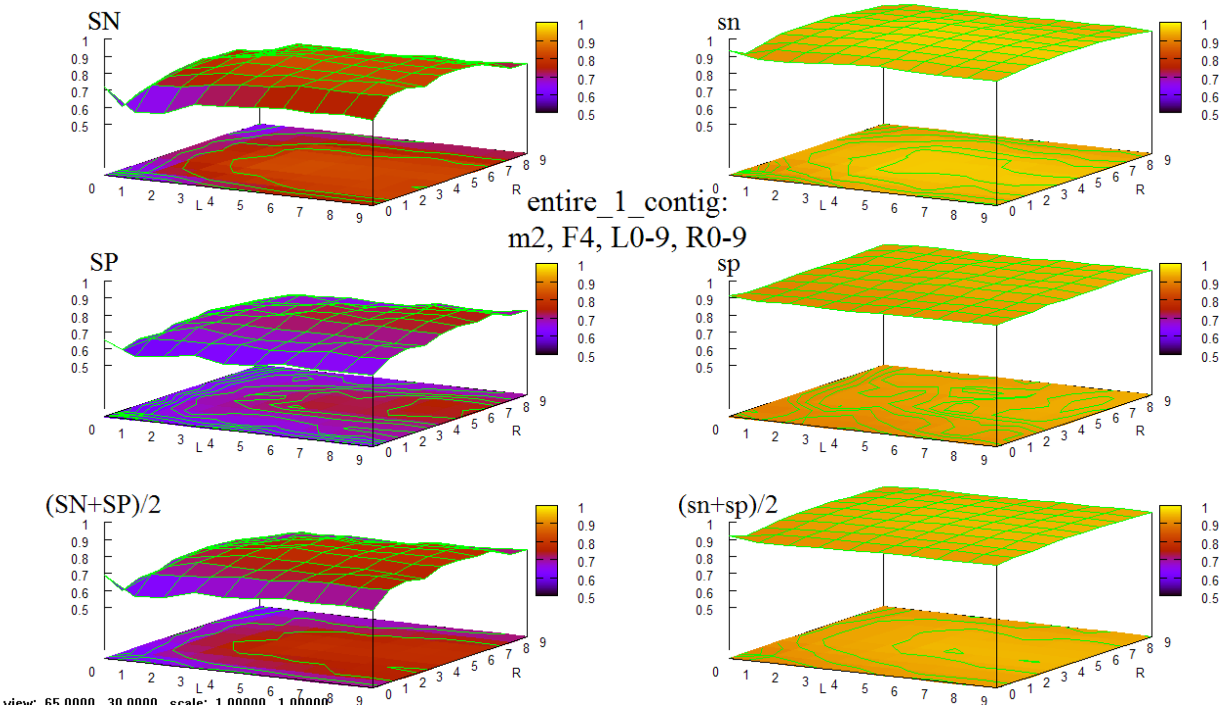


Figure 28 HOHMM performance for data entire_1_contig using m=2, F=4, L=0-9,R=0-9

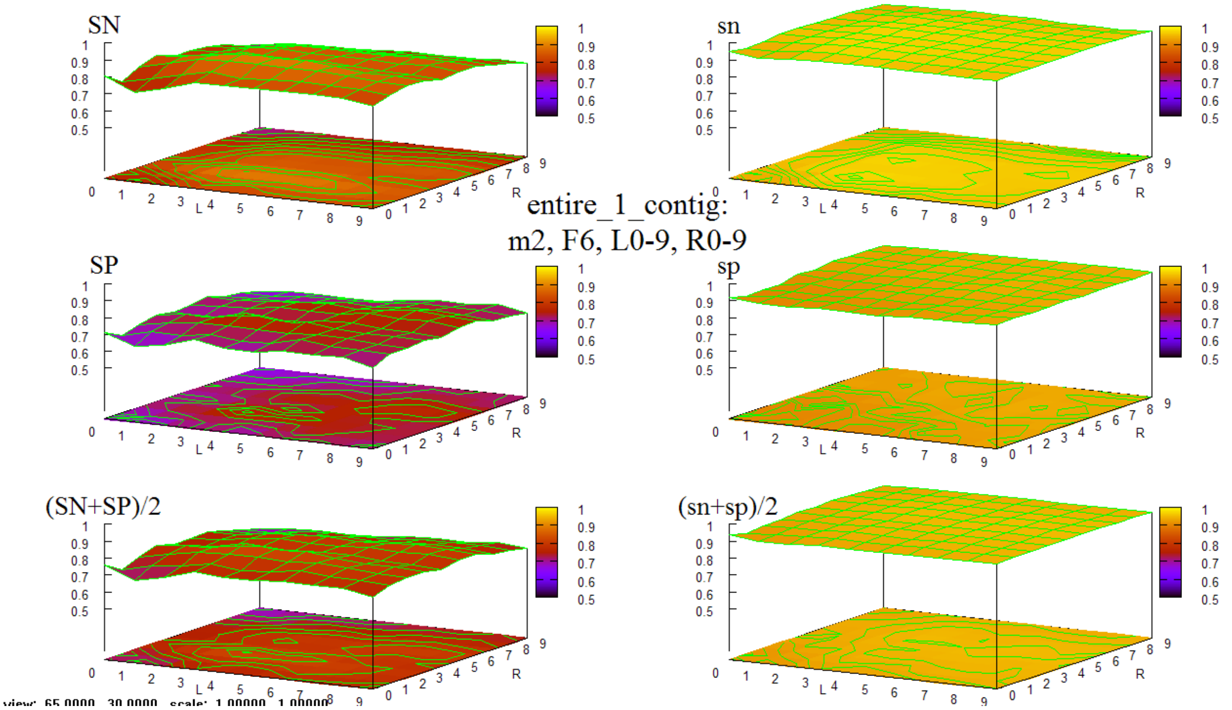


Figure 29 HOHMM performance for data entire_1_contig using m=2, F=6, L=0-9,R=0-9

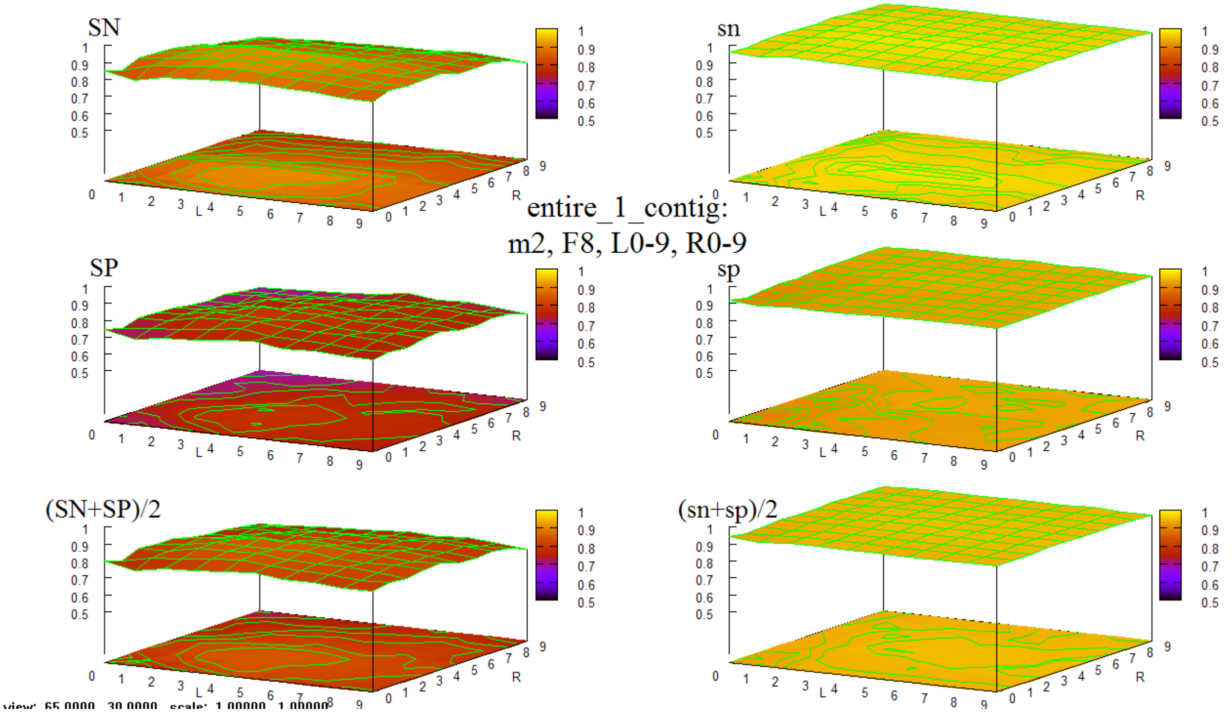


Figure 30 HOHMM performance for data entire_1_contig using m=2, F=8, L=0-9,R=0-9

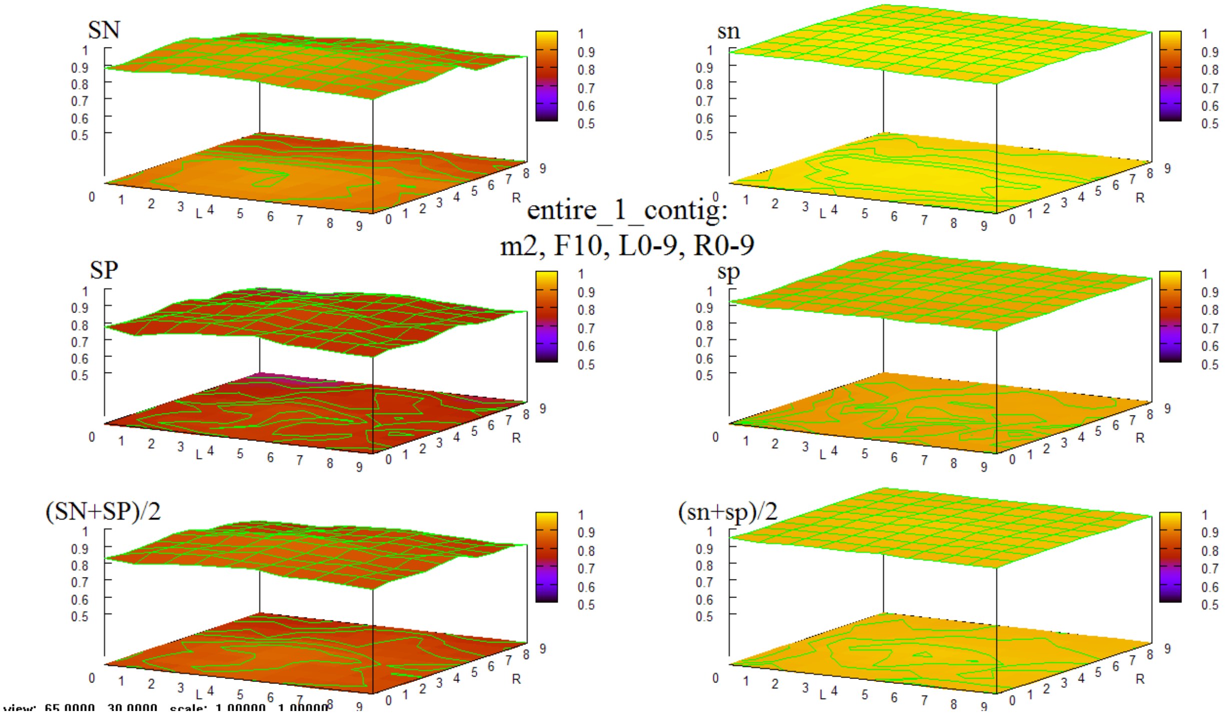


Figure 31 HOHMM performance for data entire_1_contig using m=2, F=10, L=0-9,R=0-9

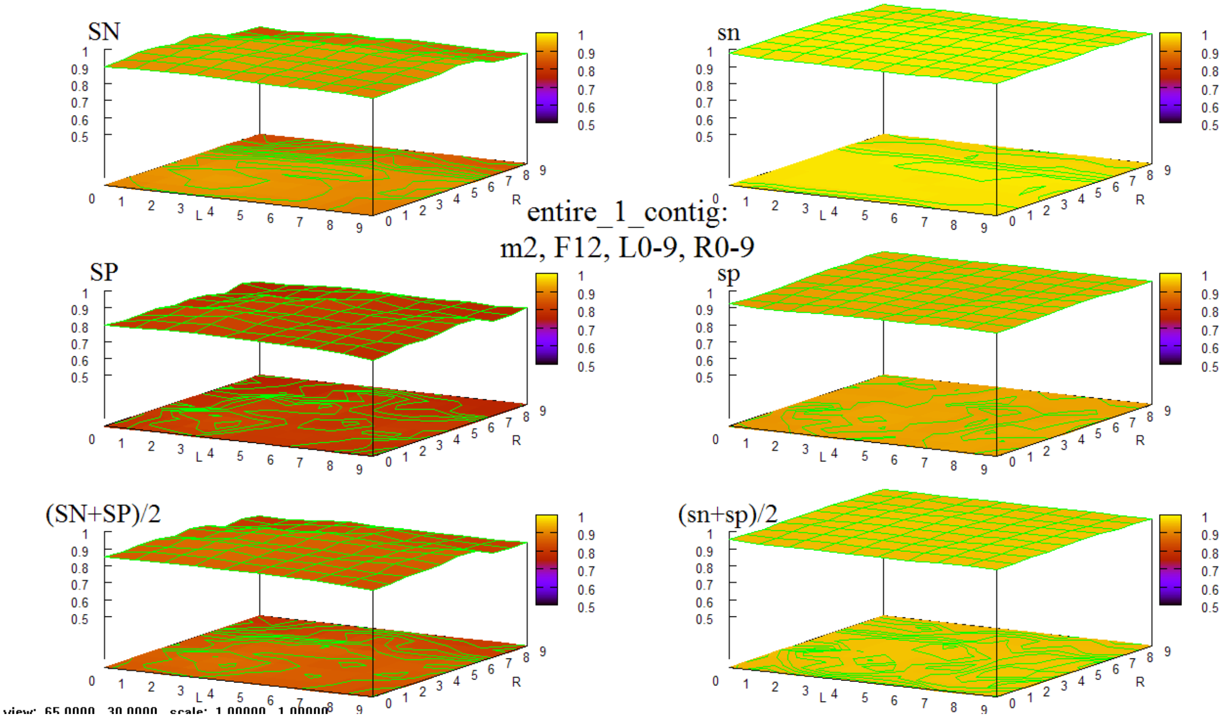


Figure 32 HOHMM performance for data entire_1_contig using m=2, F=12, L=0-9,R=0-9

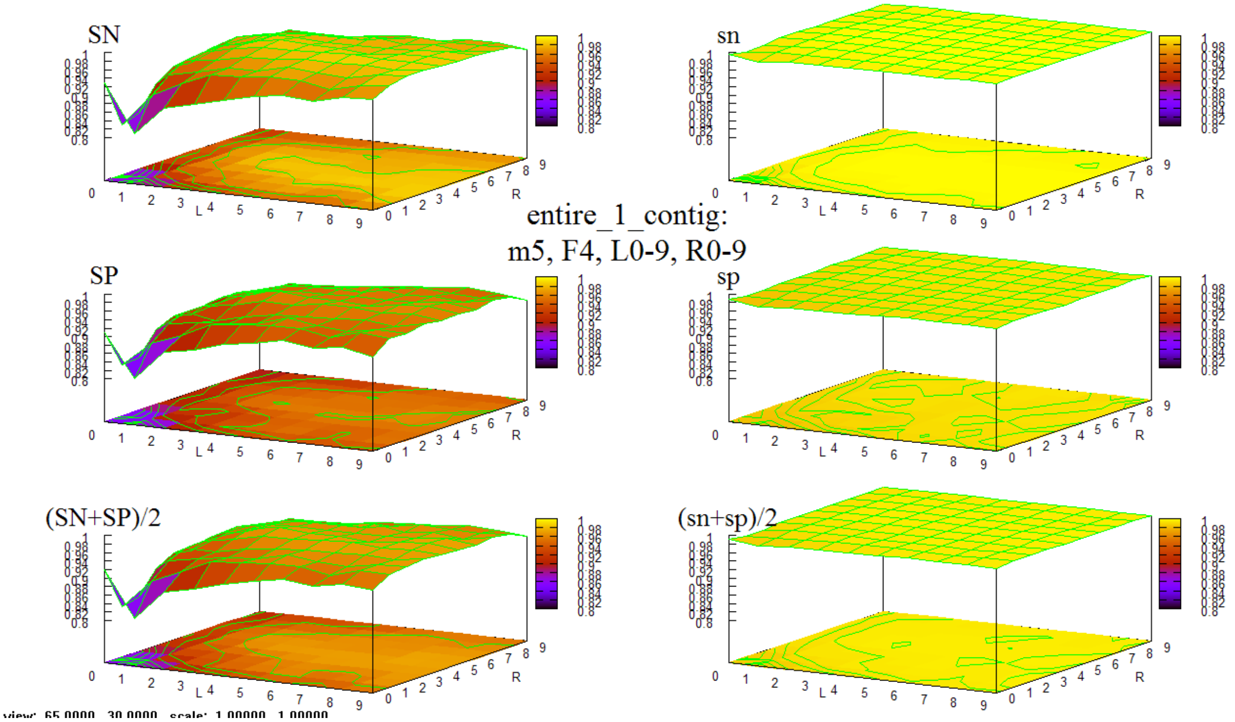


Figure 33 HOHMM performance for data entire_1_contig using m=5, F=4, L=0-9,R=0-9

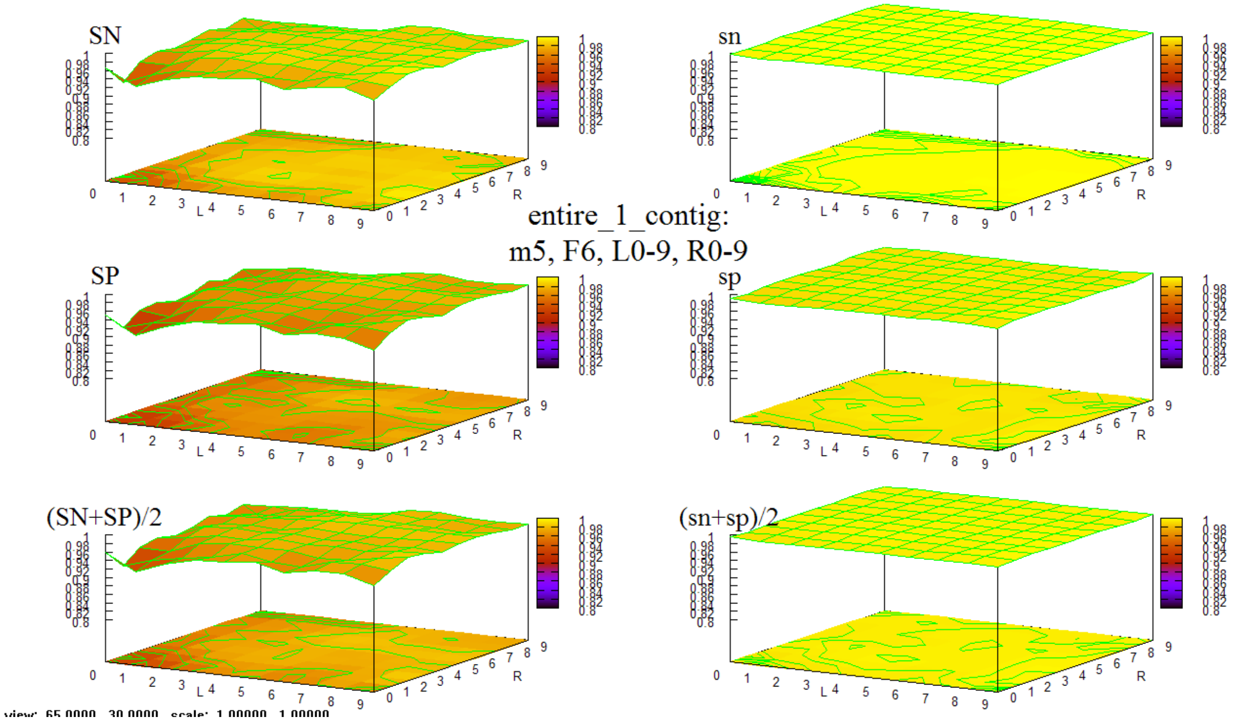


Figure 34 HOHMM performance for data entire_1_contig using m=5, F=6, L=0-9,R=0-9

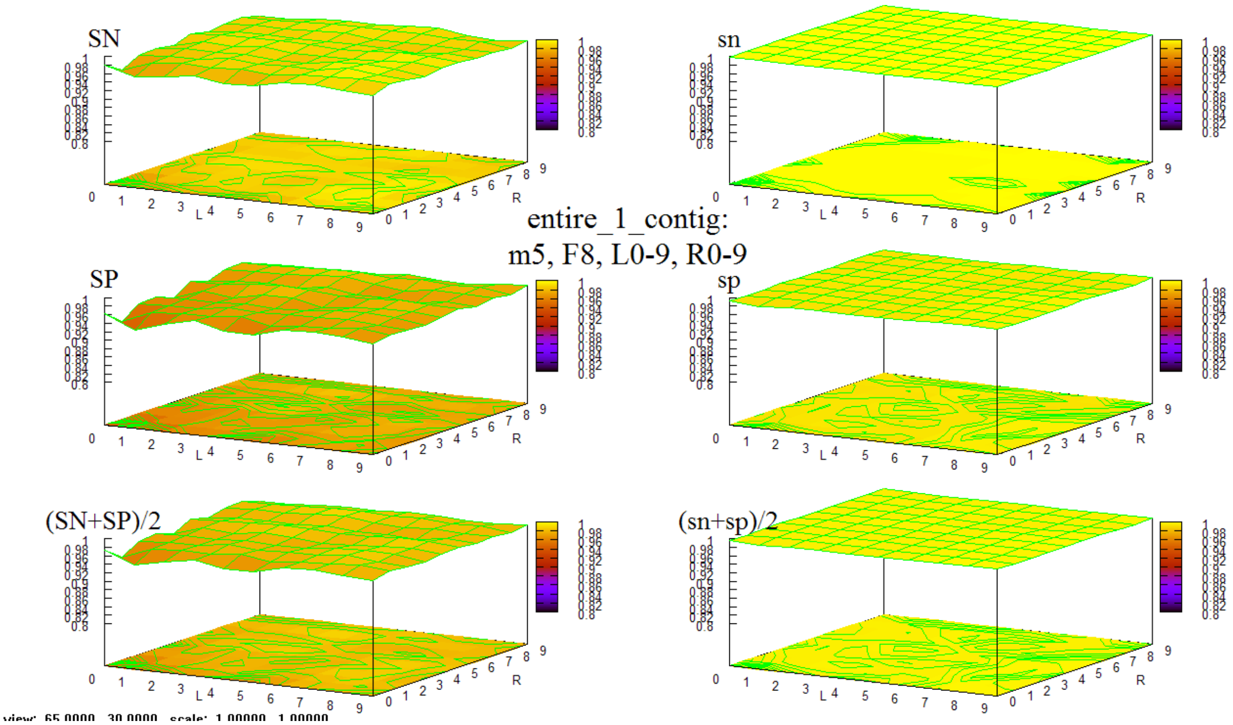


Figure 35 HOHMM performance for data entire_1_contig using m=5, F=8, L=0-9,R=0-9

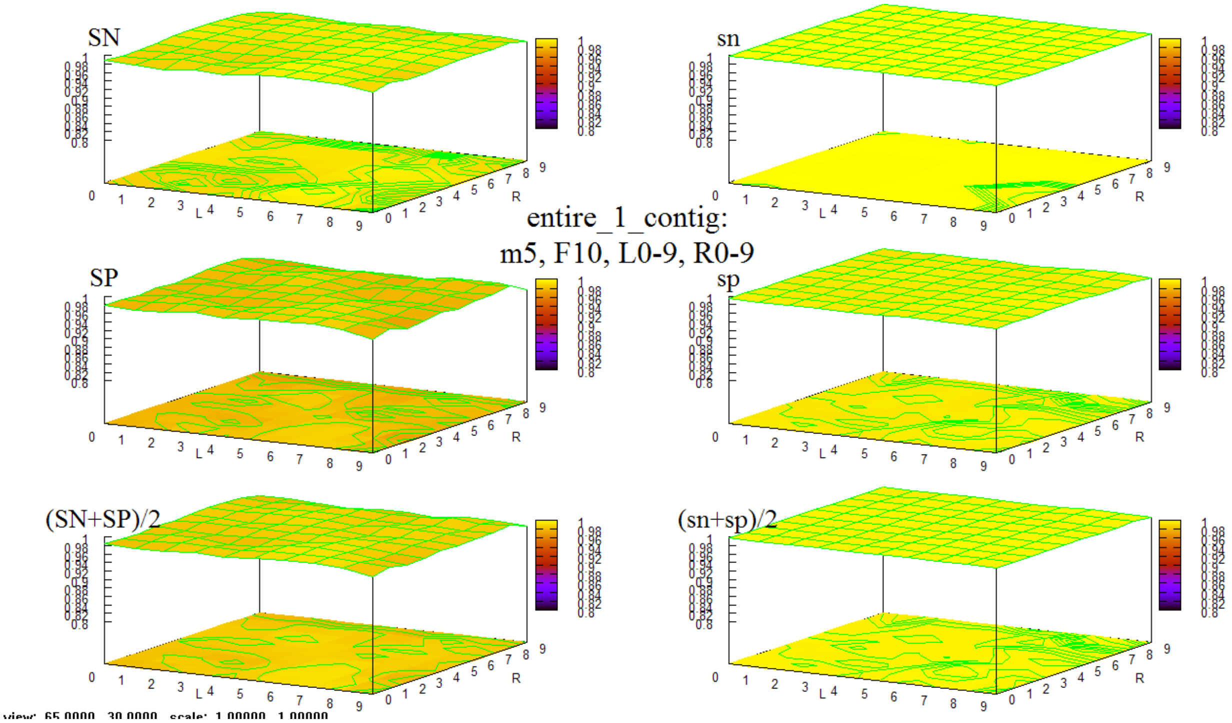


Figure 36 HOHMM performance for data entire_1_contig using m=5, F=10, L=0-9,R=0-9

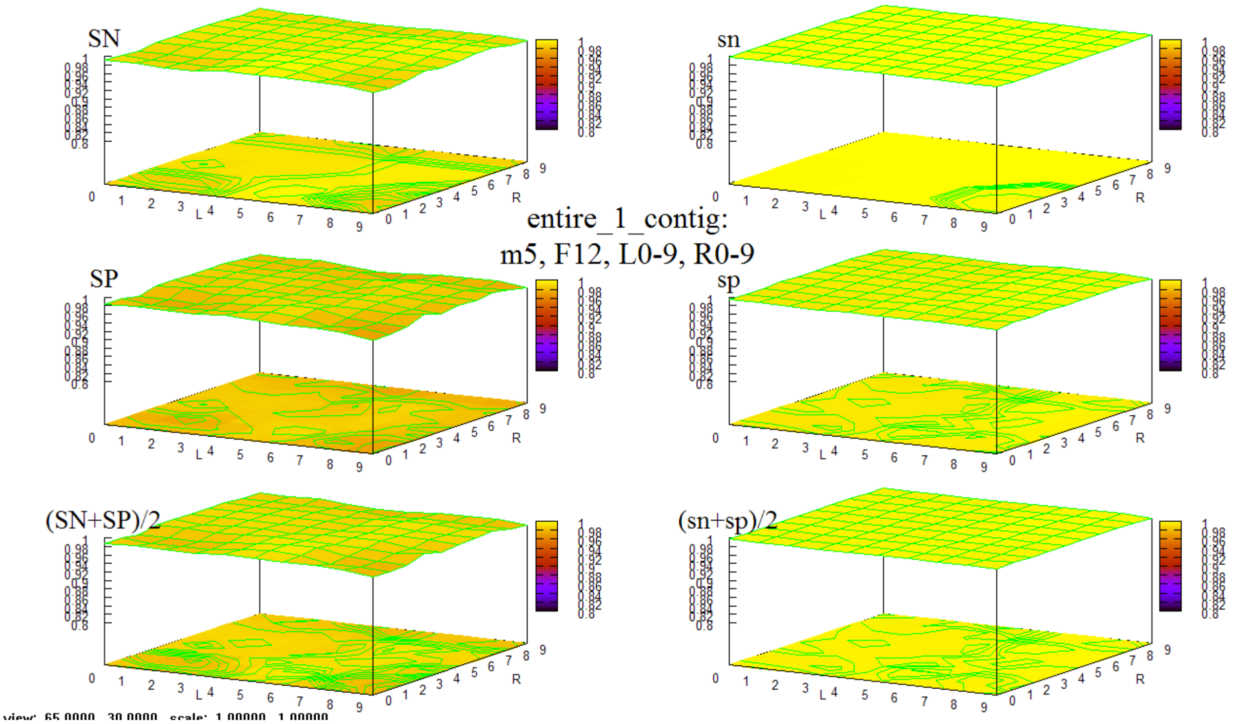


Figure 37 HOHMM performance for data entire_1_contig using m=5, F=12, L=0-9,R=0-9

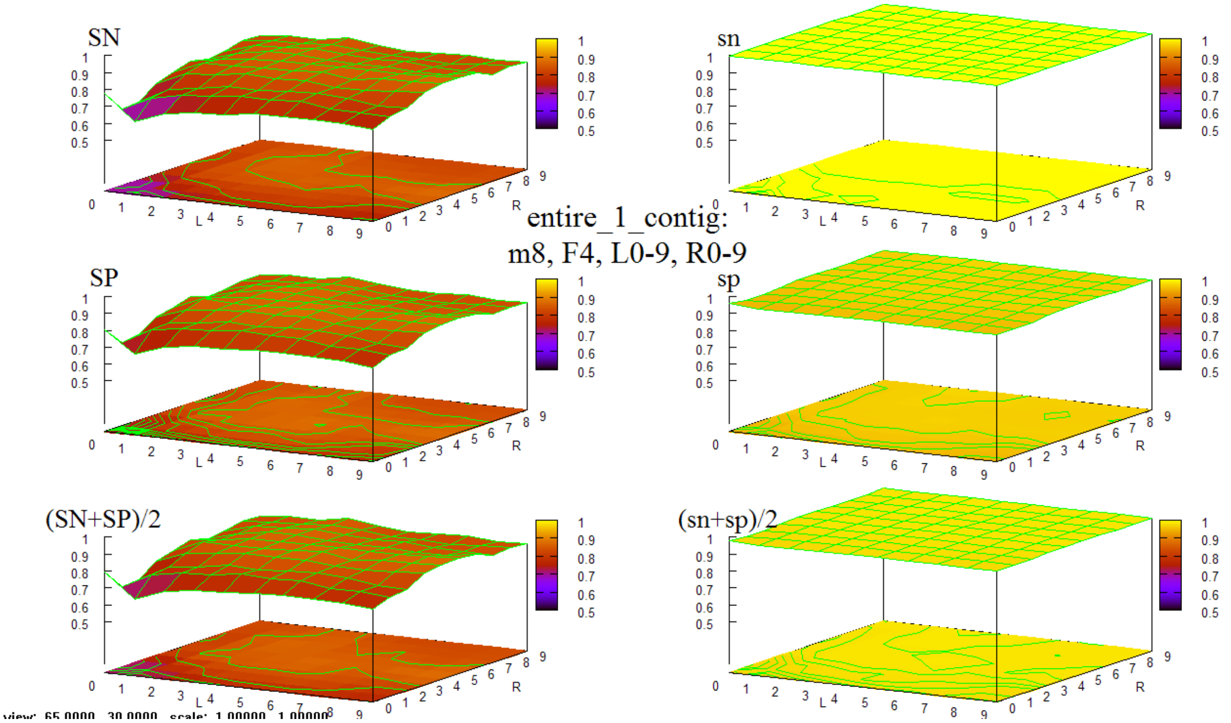


Figure 38 HOHMM performance for data entire_1_contig using m=8, F=4, L=0-9,R=0-9

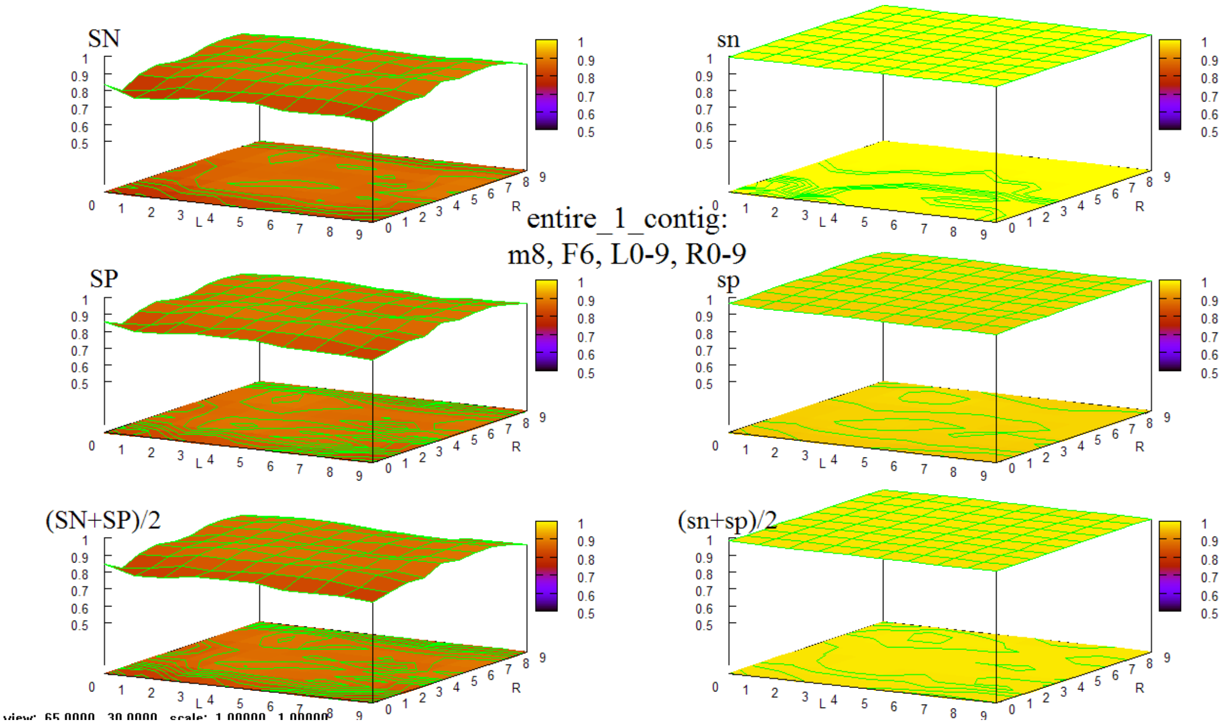


Figure 39 HOHMM performance for data entire_1_contig using m=8, F=6, L=0-9,R=0-9

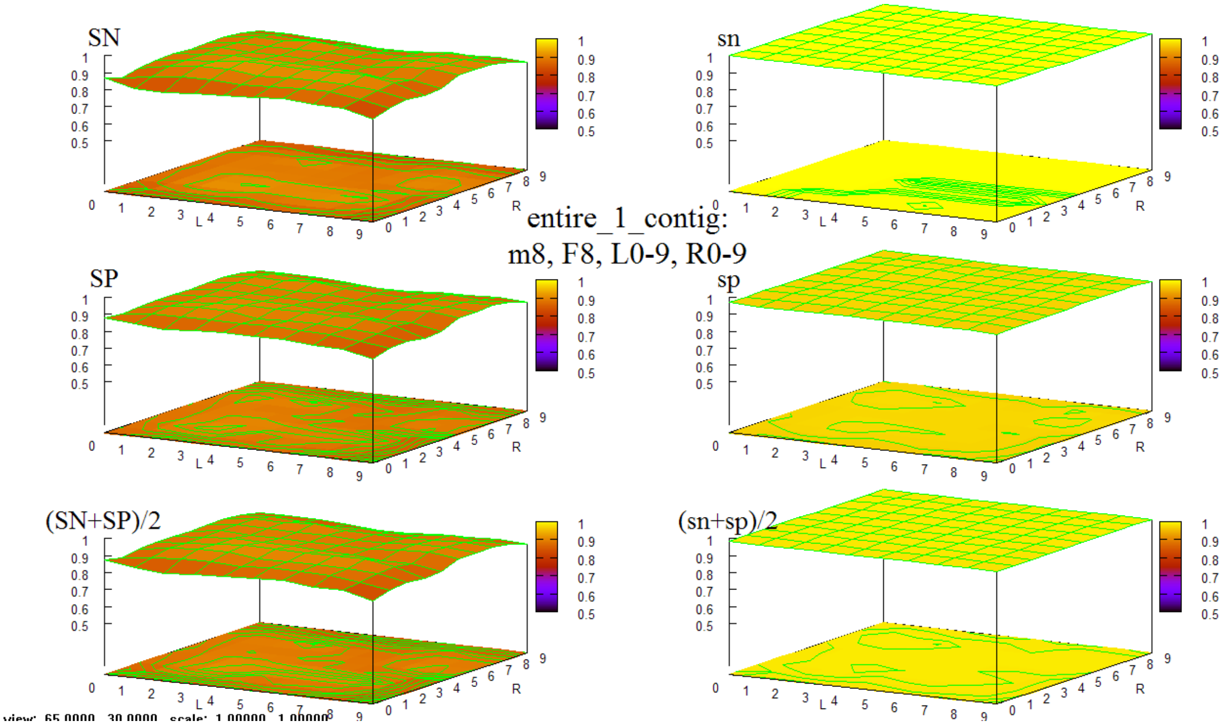


Figure 40 HOHMM performance for data entire_1_contig using m=8, F=8, L=0-9,R=0-9

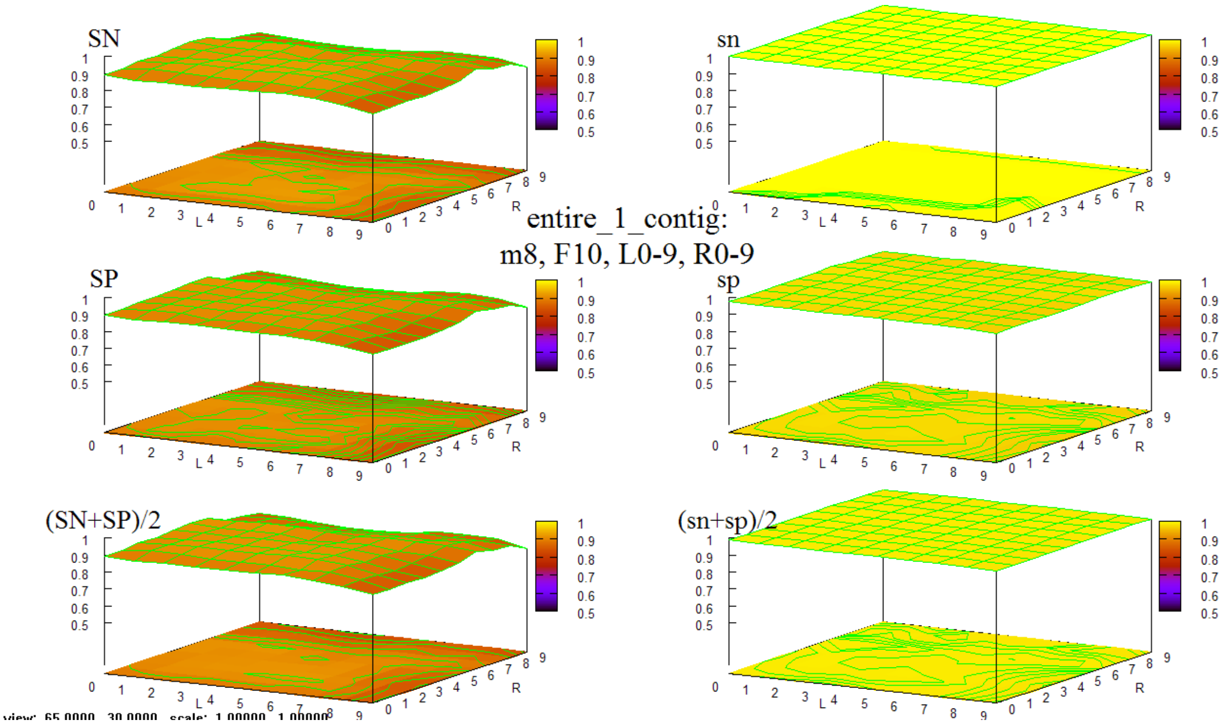


Figure 41 HOHMM performance for data entire_1_contig using m=8, F=10, L=0-9,R=0-9

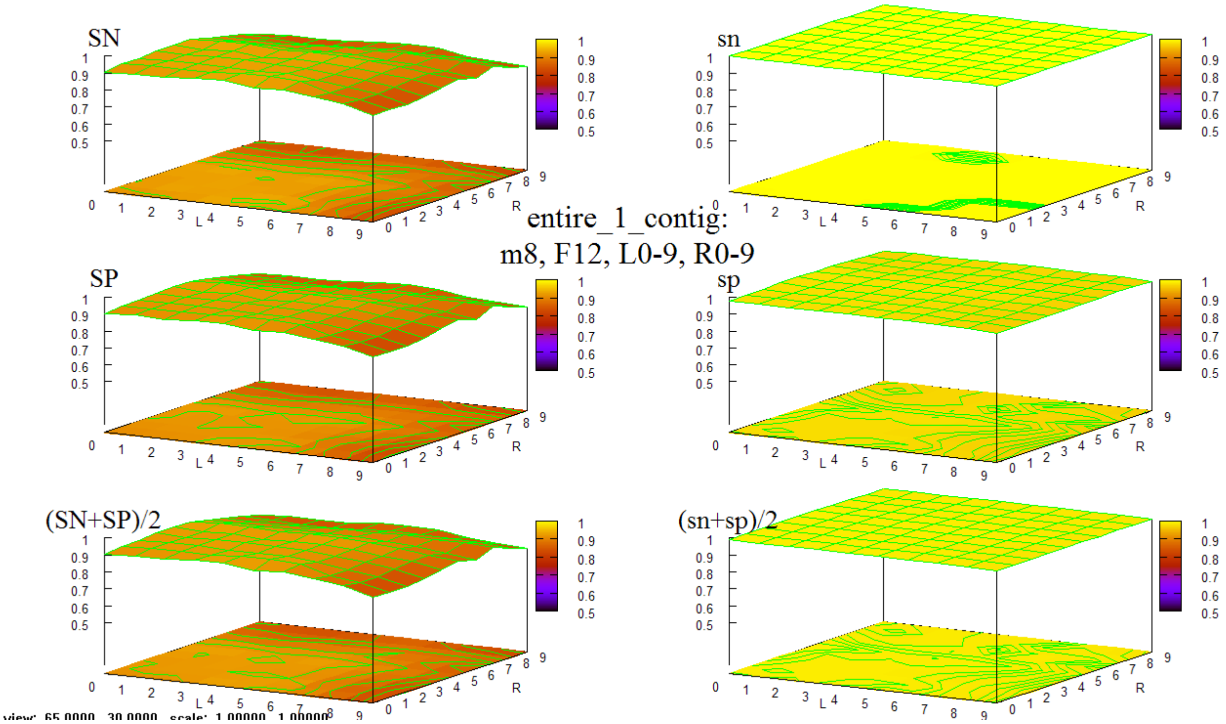


Figure 42 HOHMM performance for data entire_1_contig using m=8, F=12, L=0-9,R=0-9

8.2 Detailed Results for ALLSEQ

The graphs in Figure 43 - Figure 62 depict the detailed results of prediction accuracy of the HOHMM on the benchmark data set ALLSEQ assembled by Burset and Guigo [**17**] using the HOHMM parameters values M (=0, 2, 5, 8), F(=4, 6, 8, 10, 12), L(=0-9) and R(=0-9). In each figure, as in the previous data set entire_1_contig, the smoothness of each plotted surface serves as an indication of the stability of the HOHMM over the entire range of parameter values used.

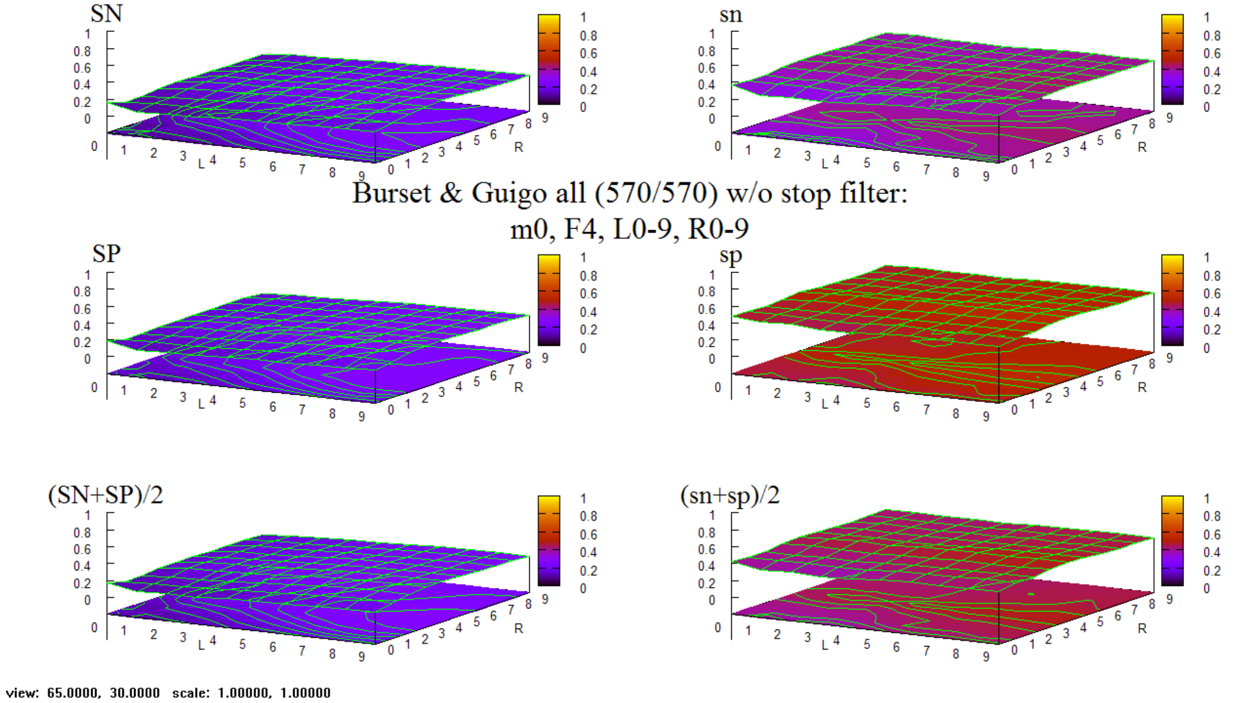


Figure 43 HOHMM performance for B & G data using m=0, F=4, L=0-9,R=0-9

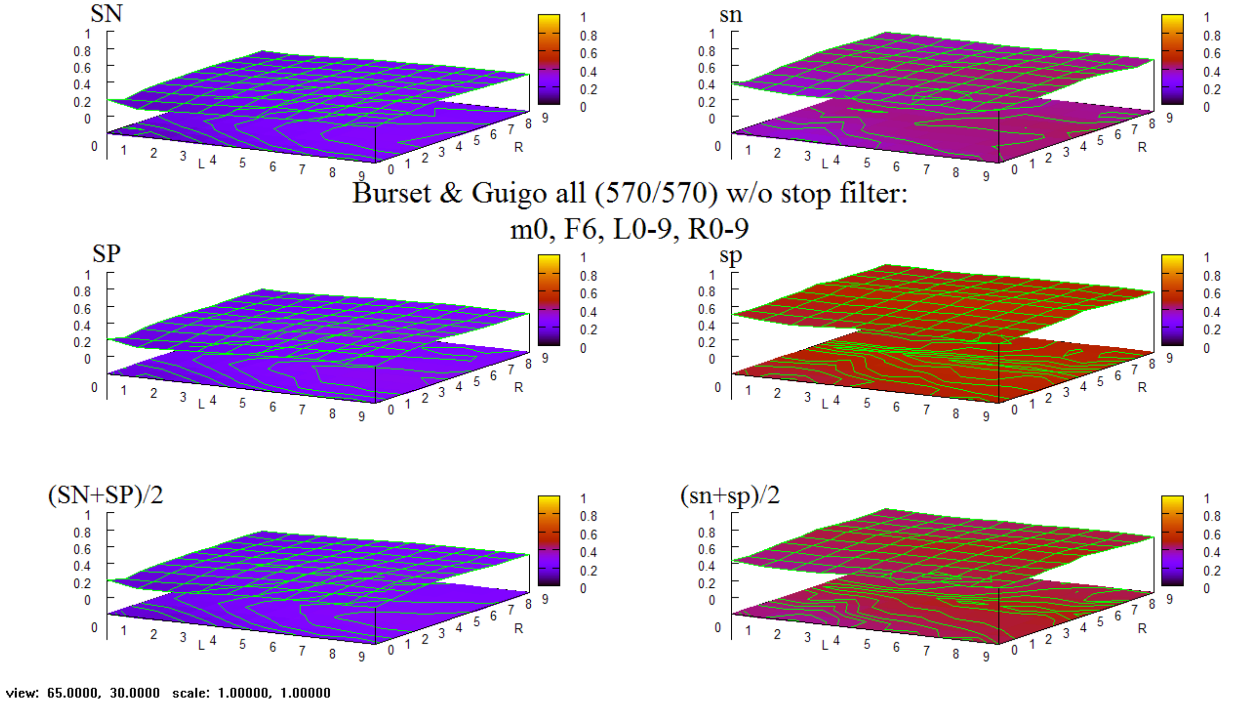


Figure 44 HOHMM performance for B & G data using m=0, F=6, L=0-9,R=0-9

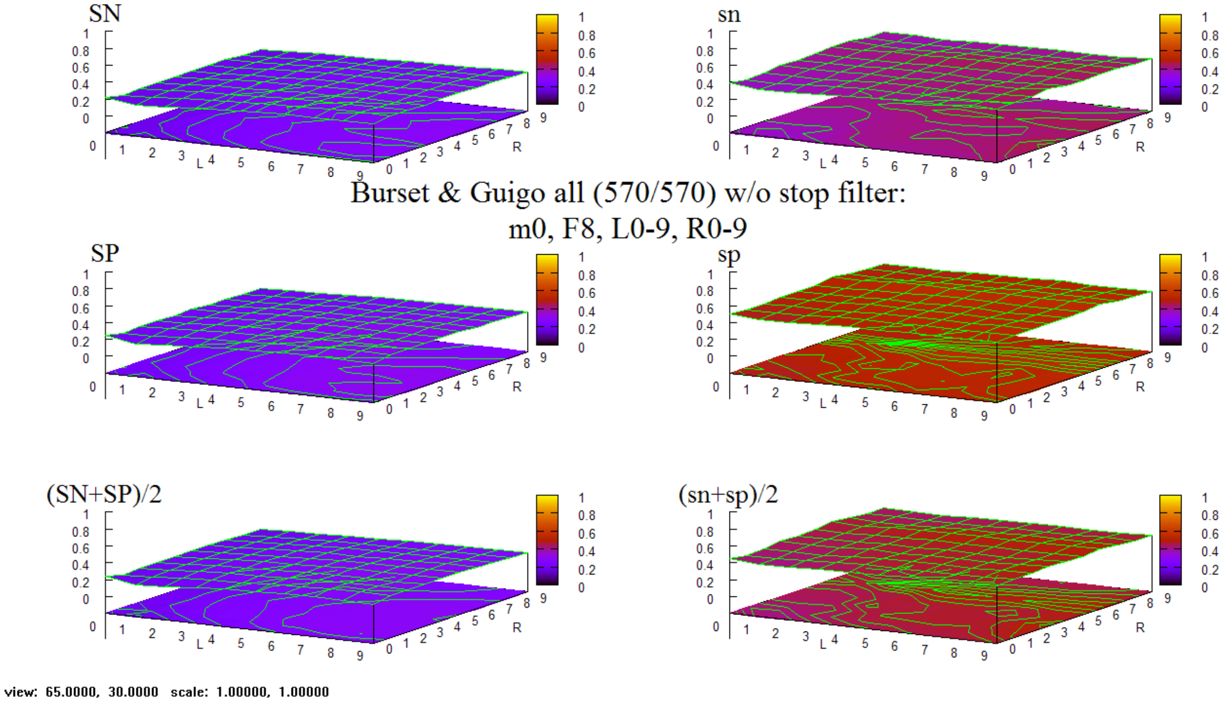


Figure 45 HOHMM performance for B & G data using m=0, F=8, L=0-9,R=0-9

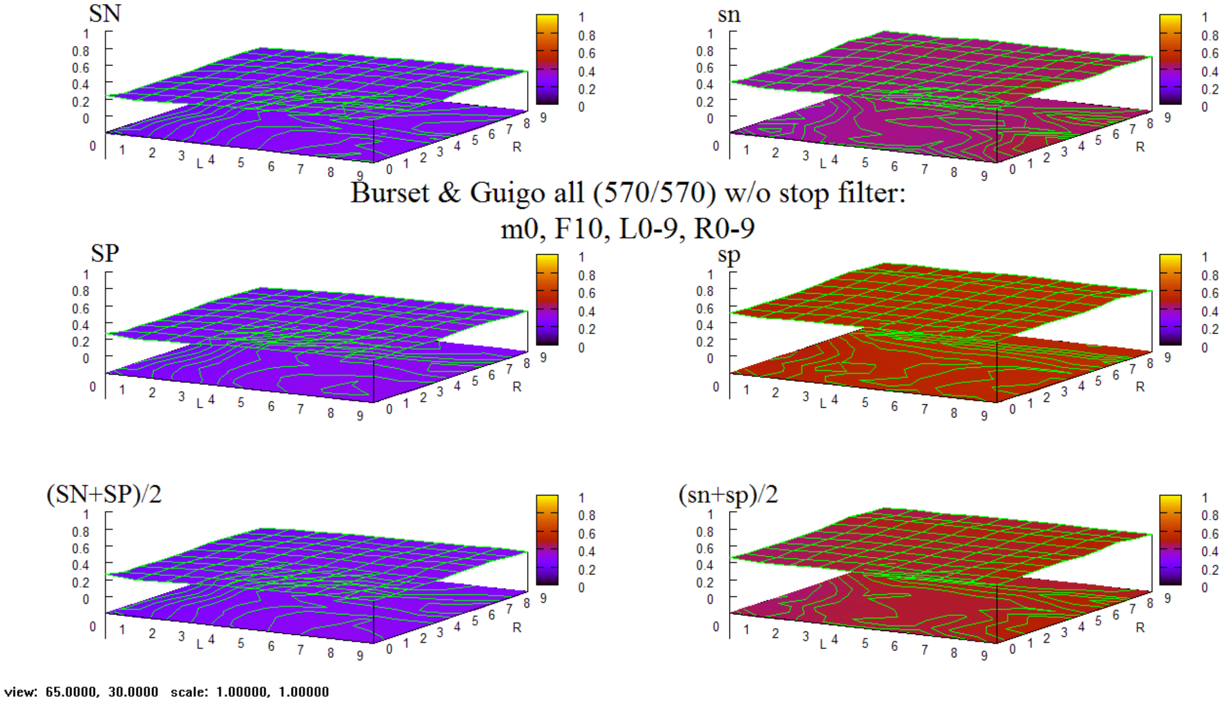


Figure 46 HOHMM performance for B & G data using m=0, F=10, L=0-9,R=0-9

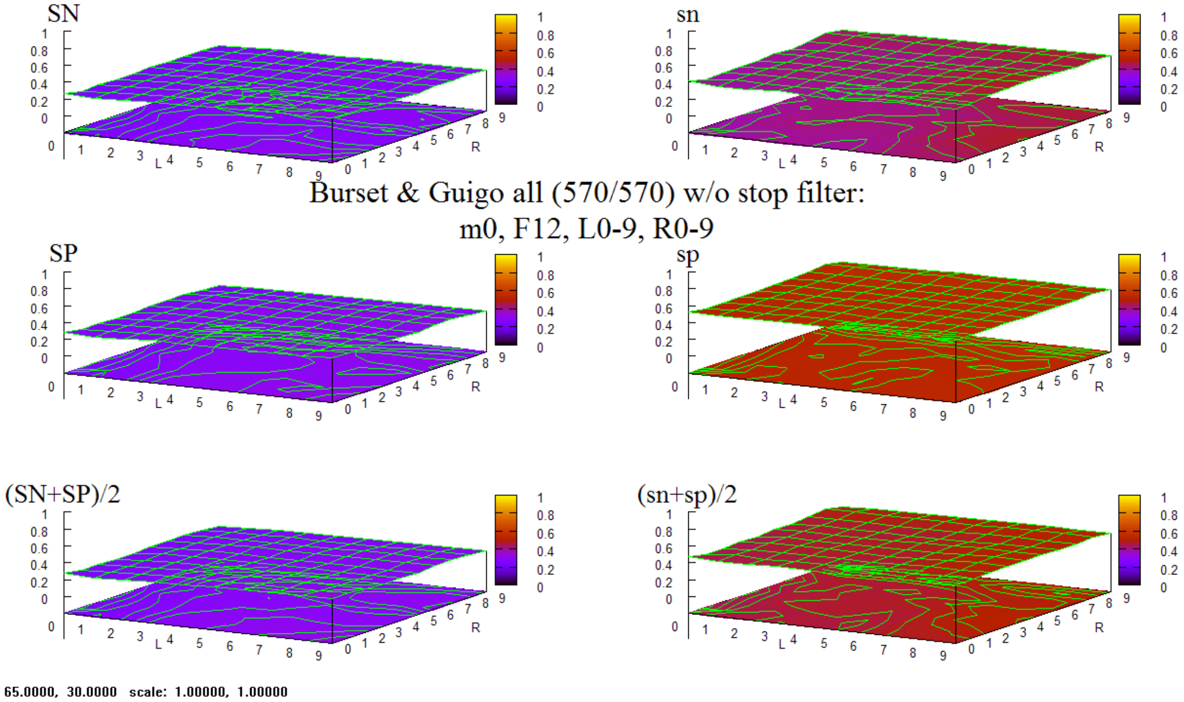


Figure 47 HOHMM performance for B & G data using m=0, F=12, L=0-9,R=0-9

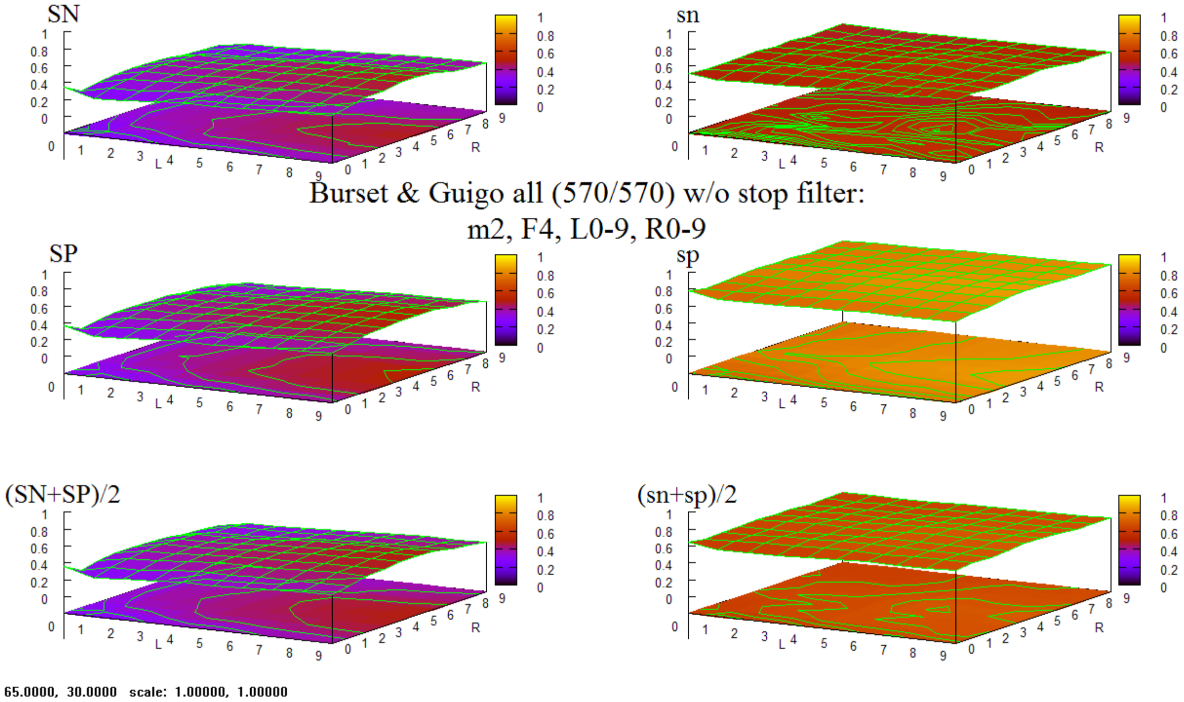


Figure 48 HOHMM performance for B & G data using m=2, F=4, L=0-9,R=0-9

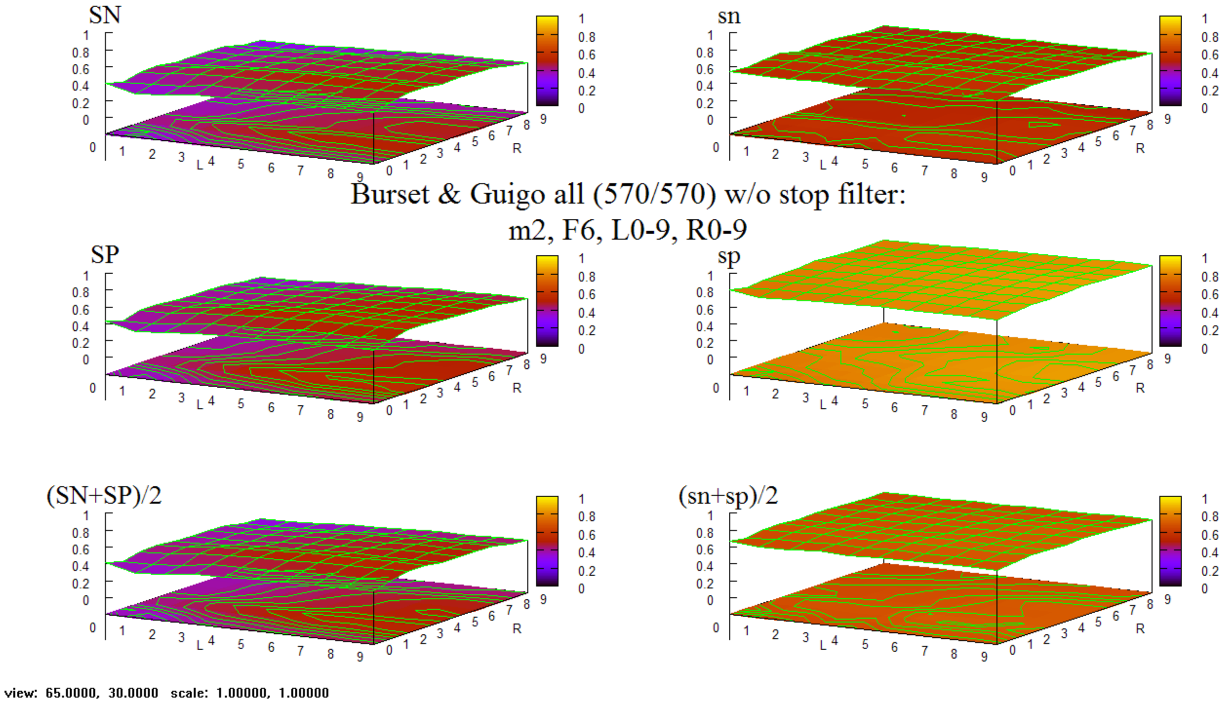


Figure 49 HOHMM performance for B & G data using m=2 , F=6, L=0-9,R=0-9

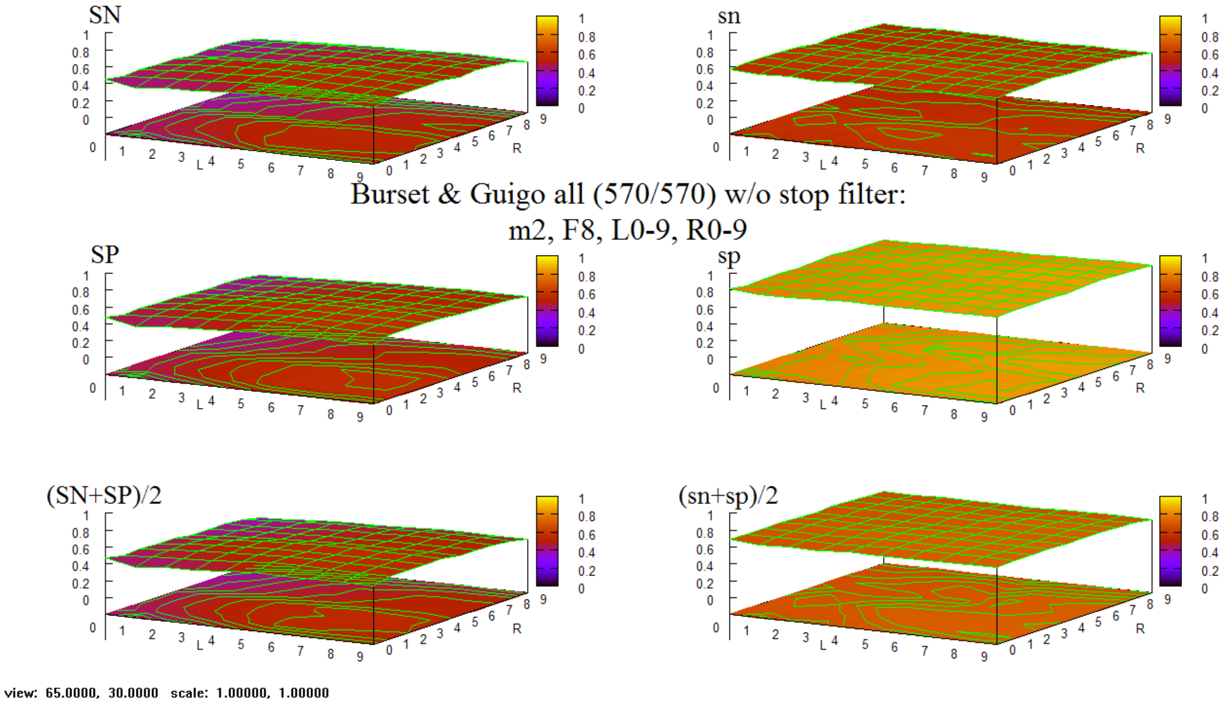


Figure 50 HOHMM performance for B & G data using m=2, F=8, L=0-9,R=0-9

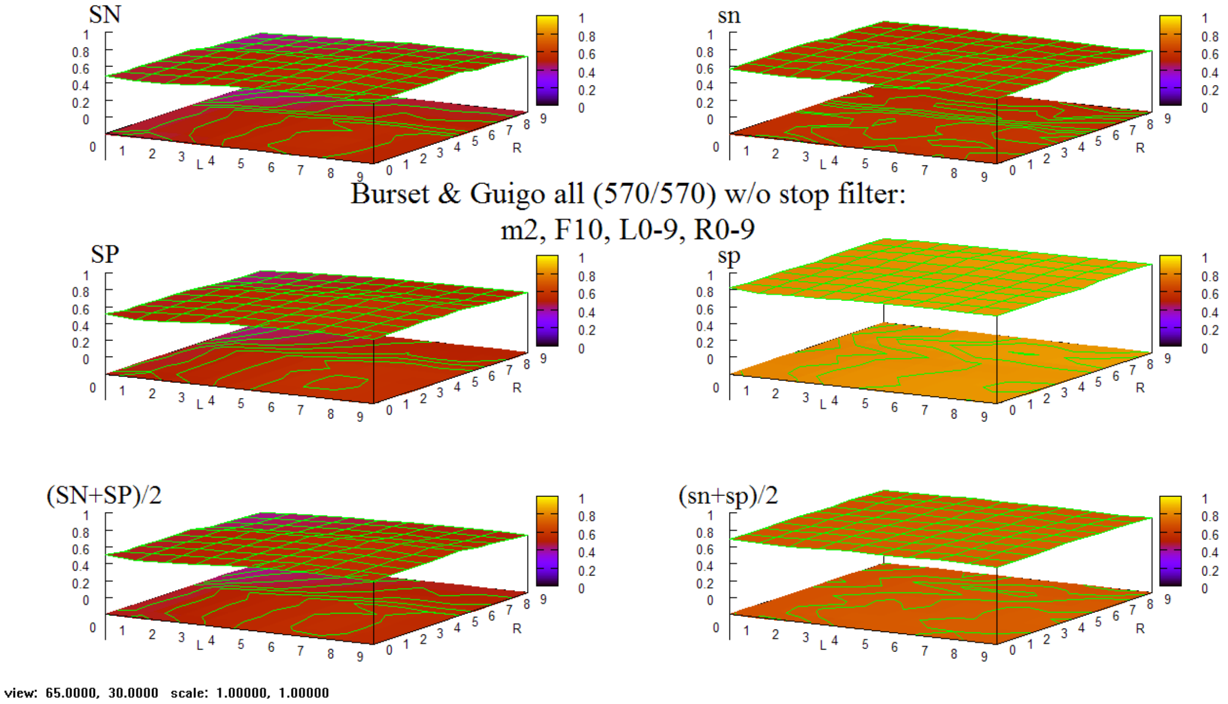


Figure 51 HOHMM performance for B & G data using m=2, F=10, L=0-9,R=0-9

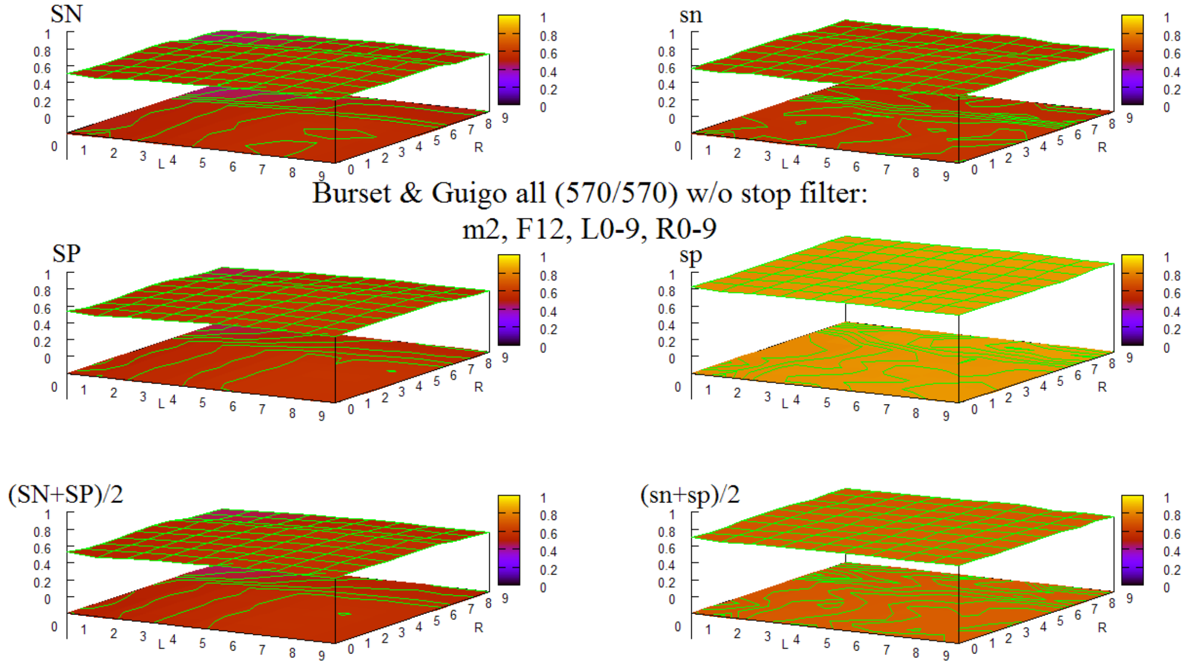


Figure 52 HOHMM performance for B & G data using m=2, F=12, L=0-9,R=0-9

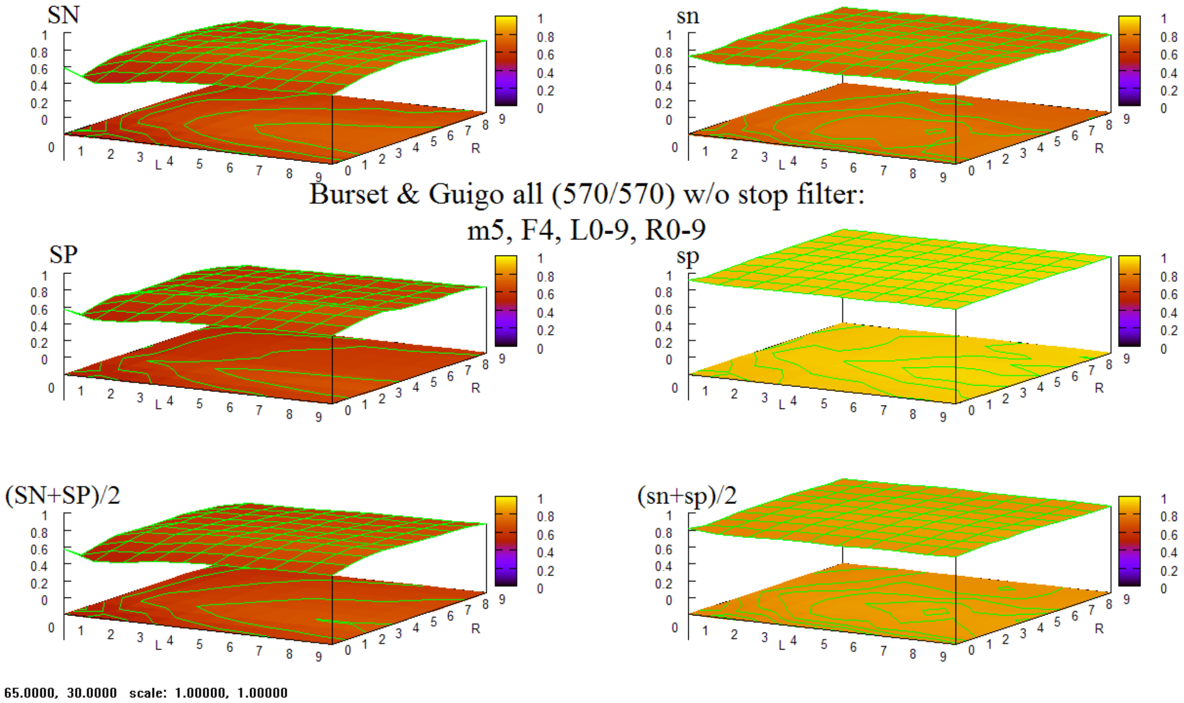


Figure 53 HOHMM performance for B & G data using m=5, F=4, L=0-9,R=0-9

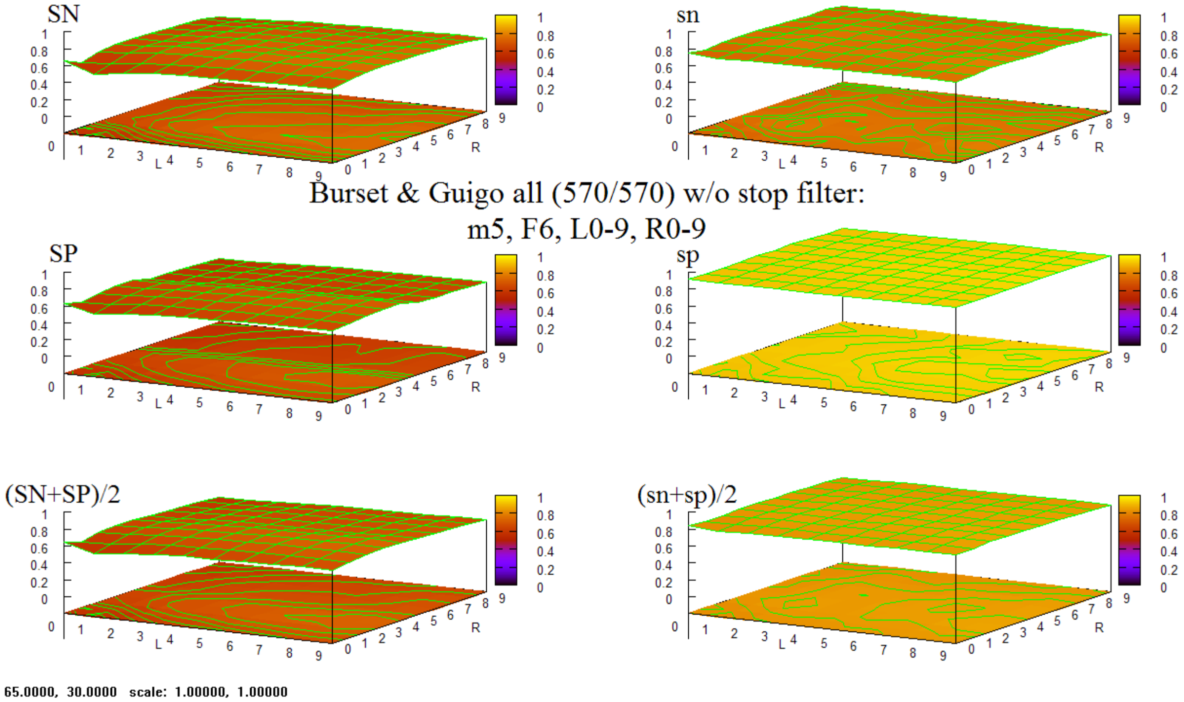


Figure 54 HOHMM performance for B & G data using m=5, F=6, L=0-9,R=0-9

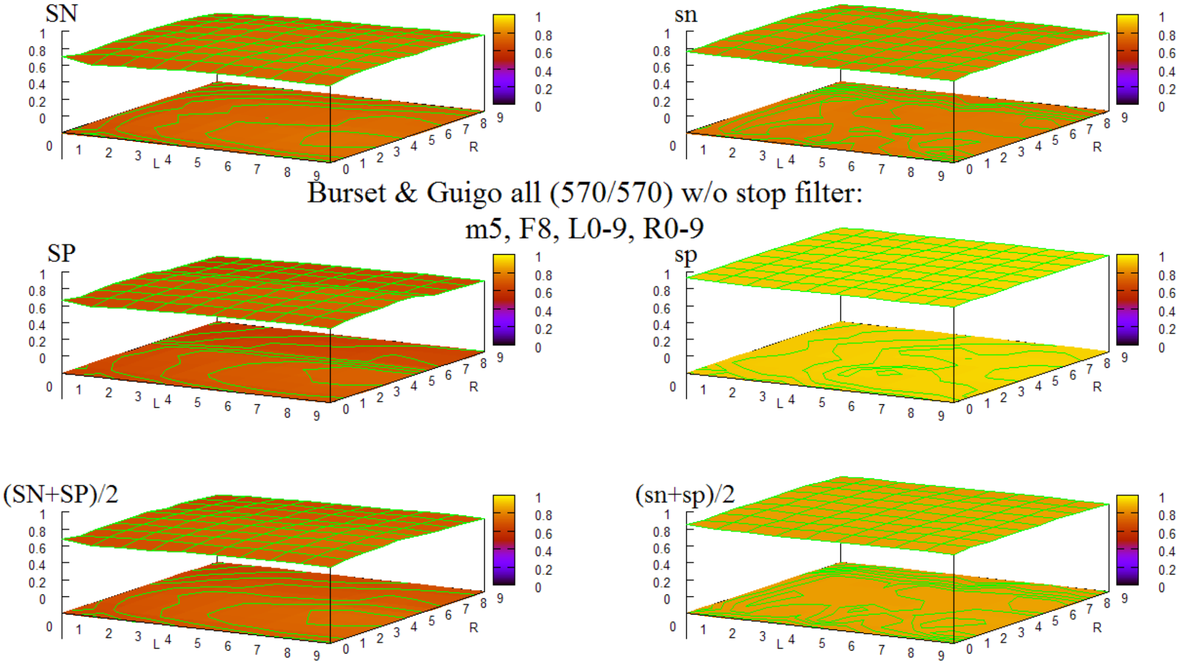


Figure 55 HOHMM performance for B & G data using m=5, F=8, L=0-9,R=0-9

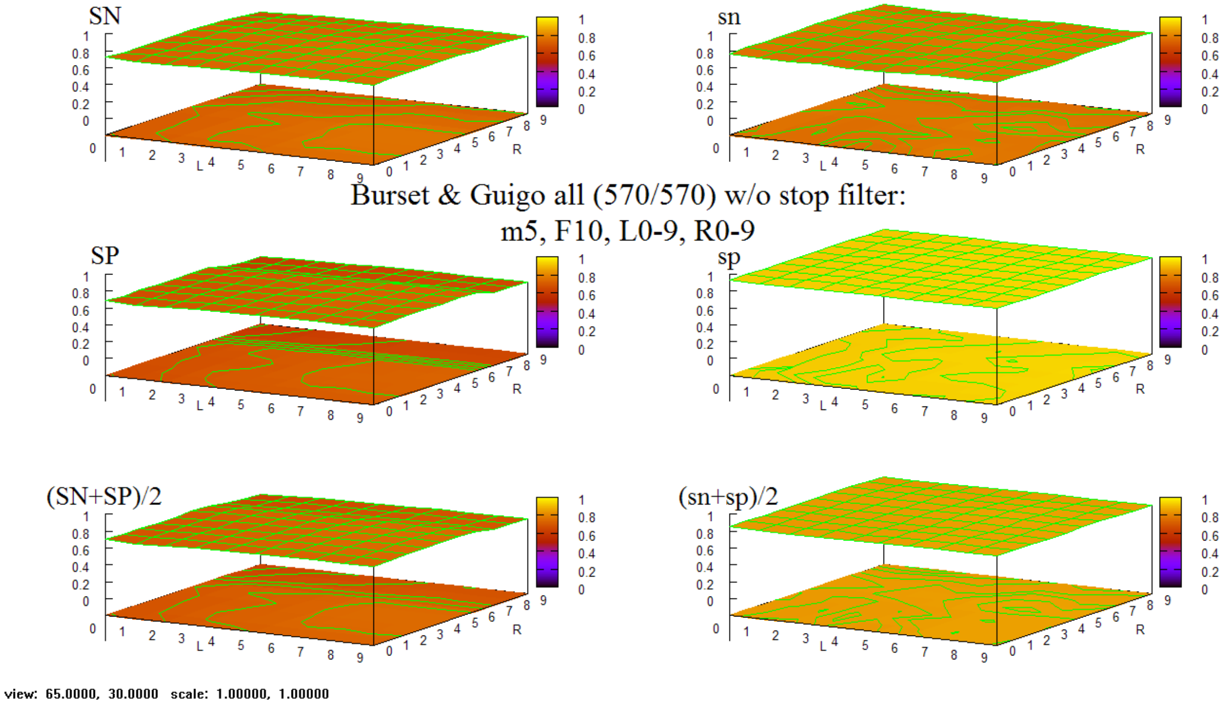


Figure 56 HOHMM performance for B & G data using m=5, F=10, L=0-9,R=0-9

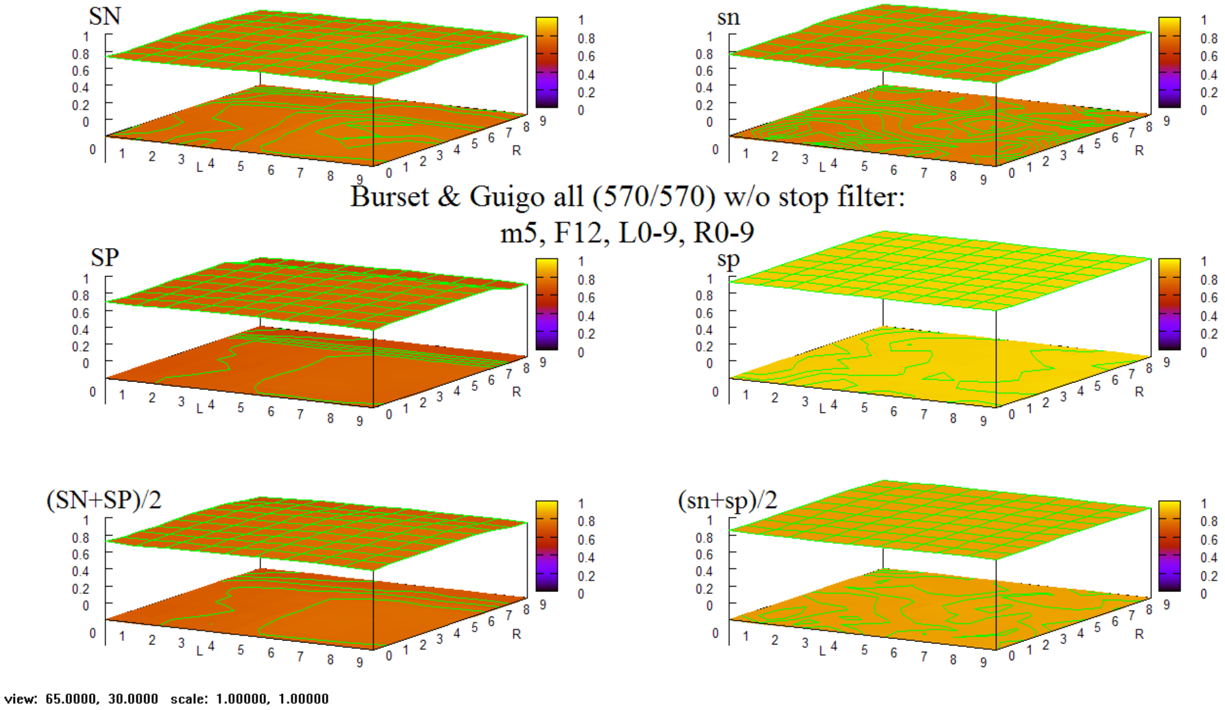


Figure 57 HOHMM performance for B & G data using m=5, F=12, L=0-9,R=0-9

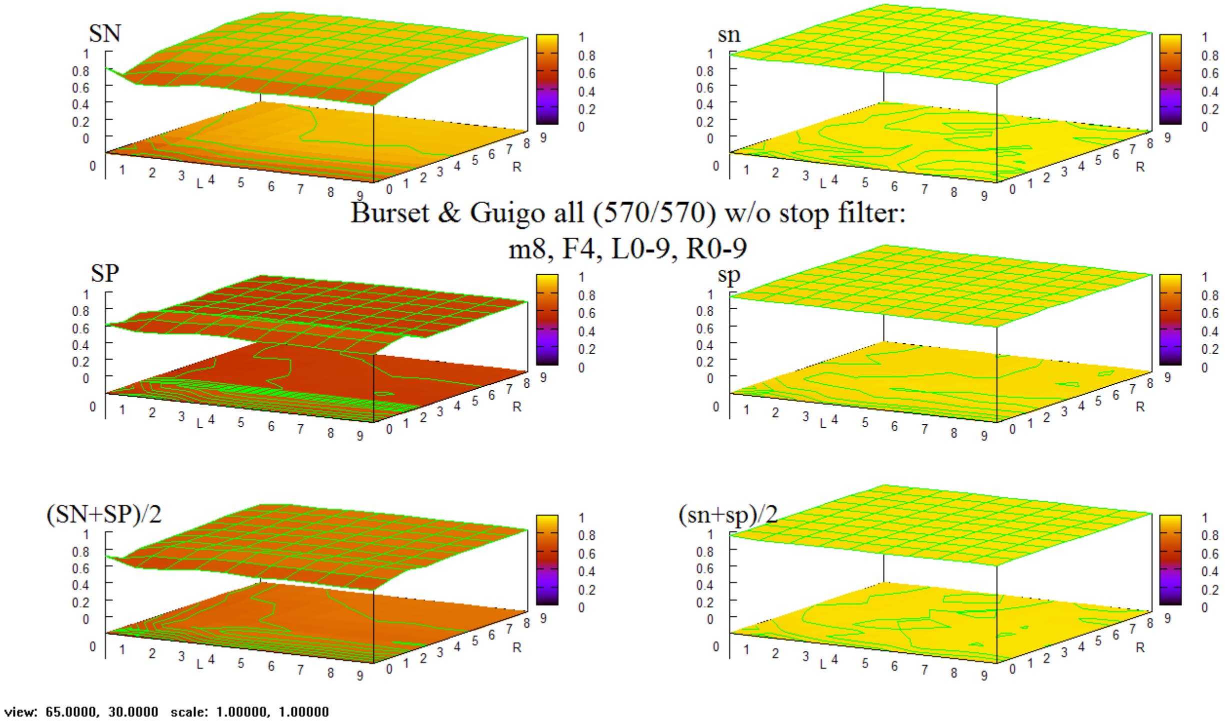


Figure 58 HOHMM performance for B & G data using m=8, F=4, L=0-9,R=0-9

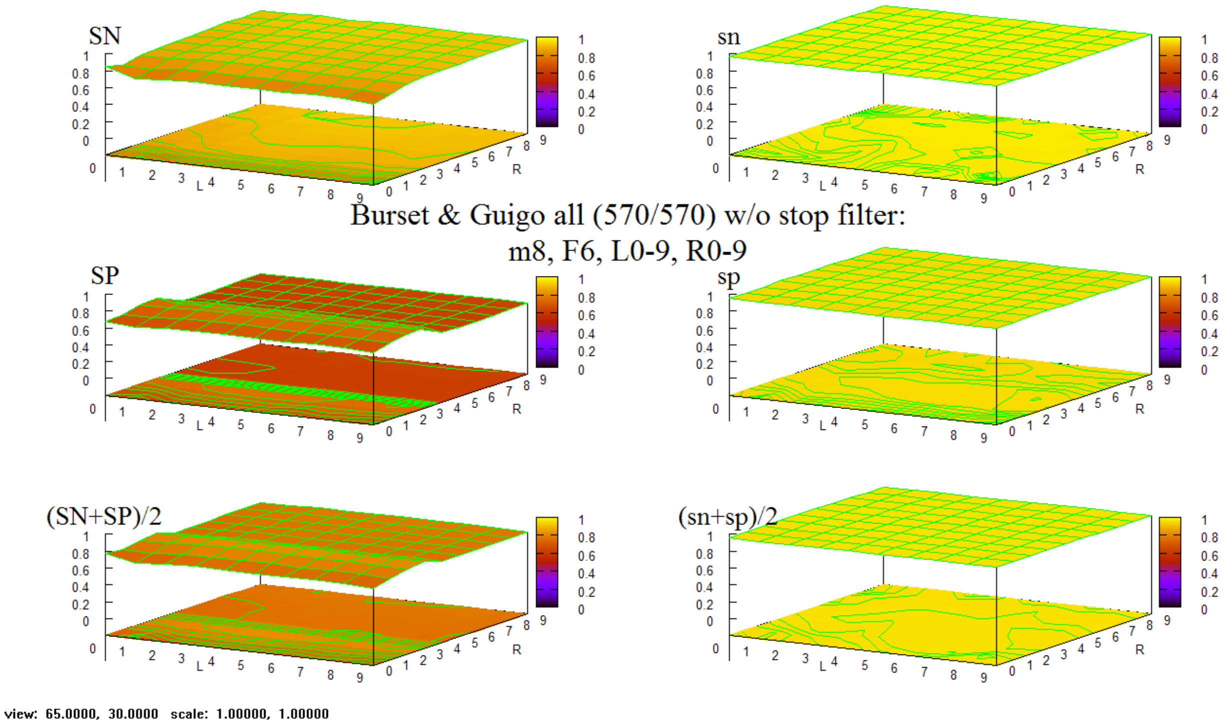


Figure 59 HOHMM performance for B & G data using m=8, F=6, L=0-9,R=0-9

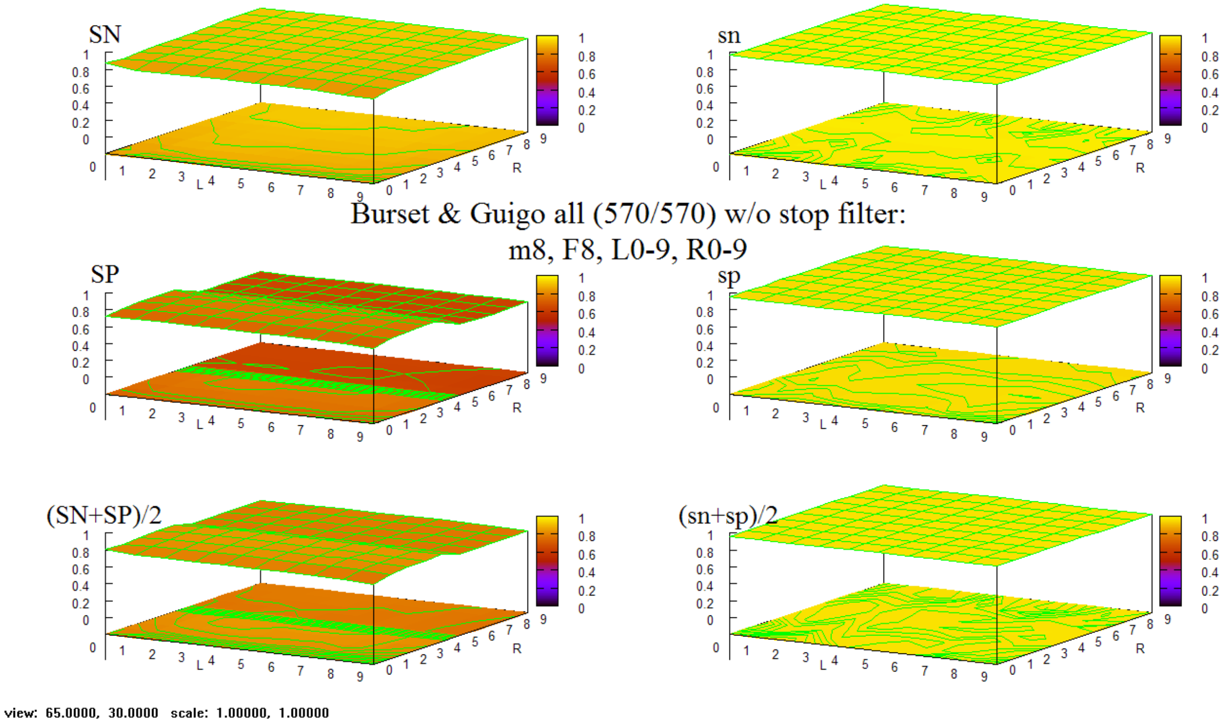


Figure 60 HOHMM performance for B & G data using m=8, F=8, L=0-9,R=0-9

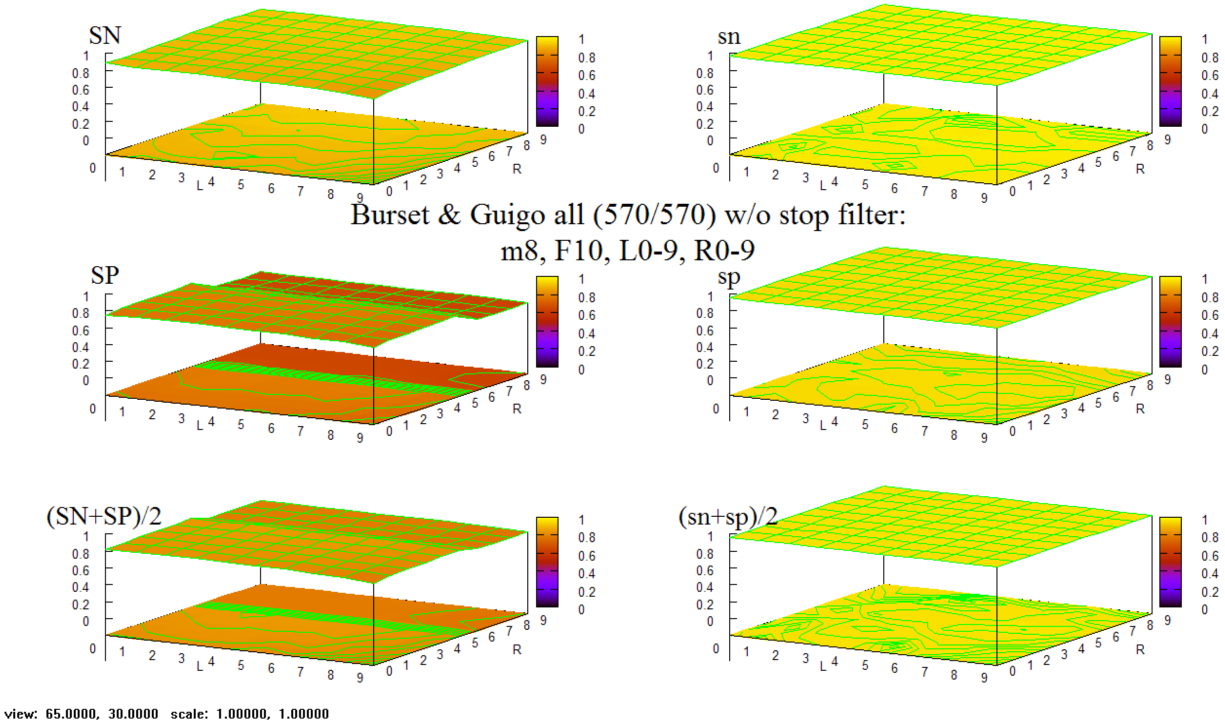


Figure 61 HOHMM performance for B & G data using m=8, F=10, L=0-9,R=0-9

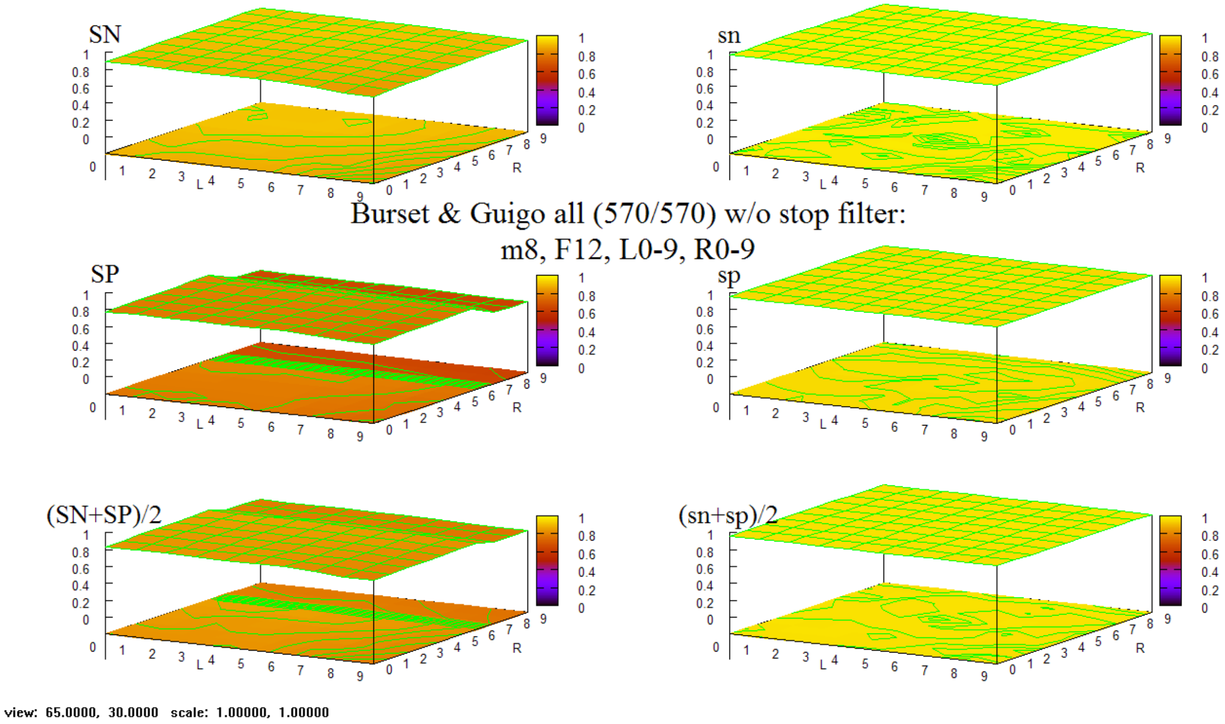


Figure 62 HOHMM performance for B & G data using m=8, F=12, L=0-9,R=0-9

8.3 Detailed Results for Chromosomes I-V of *C. elegans*

The following sequence of plots in Figure 63 - Figure 102 depicts the results of 5-fold cross-validation analysis of prediction accuracy of the HOHMM on Chromosomes I-V of *C. elegans* using the parameter values M (=0, 2, 5, 8), F(=2, 4, 6, 8, 10, 12, 14, 16, 18, 20), L(=0-9) and R(=0-9). Note that no iterative, in-frame stop filtering was performed. Here also, as in the previous data sets entire_1_contig and ALLSEQ, in each figure, the smoothness of each plotted surface serves as an indication of the stability of the HOHMM over the entire range of parameter values used.

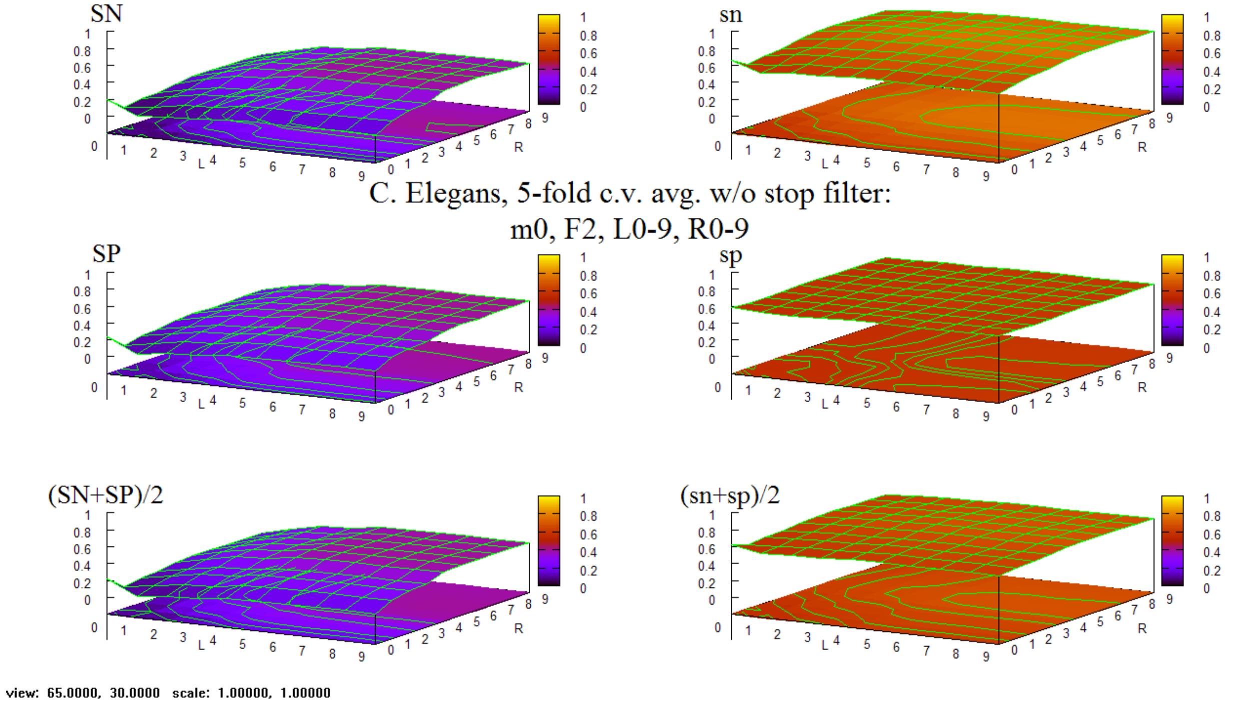


Figure 63 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=2, L=0-9,R=0-9

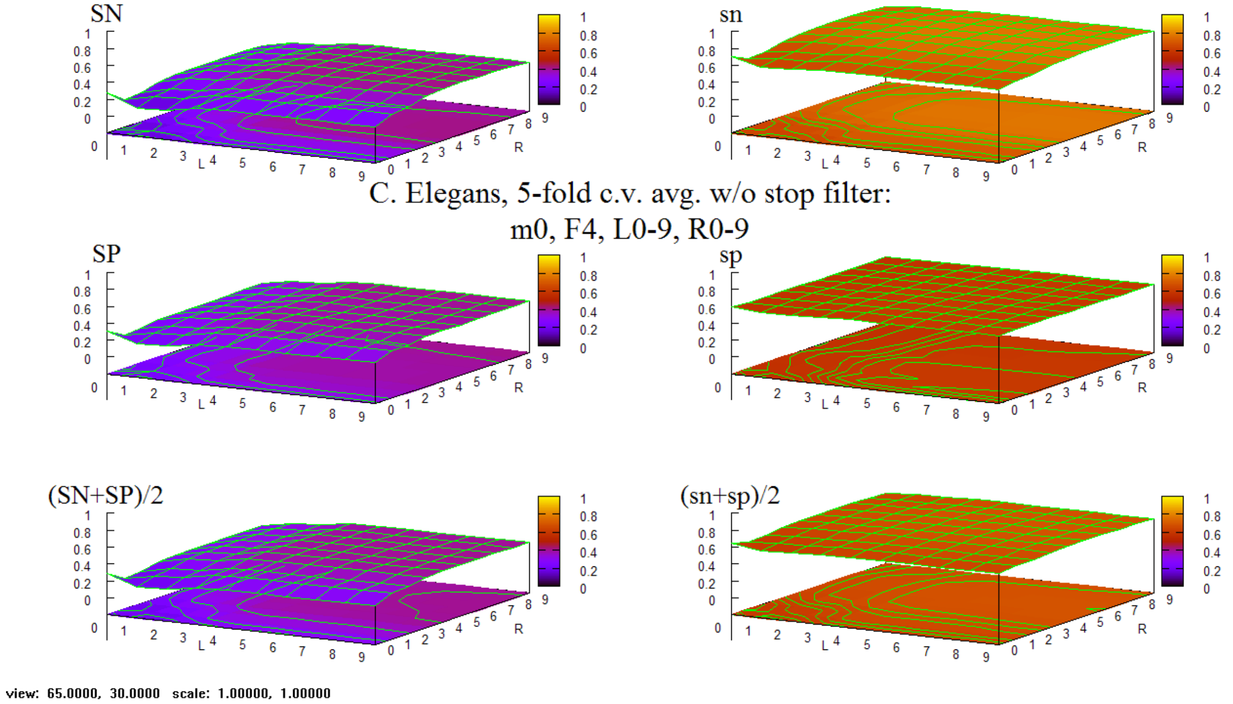


Figure 64 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=4, L=0-9,R=0-9

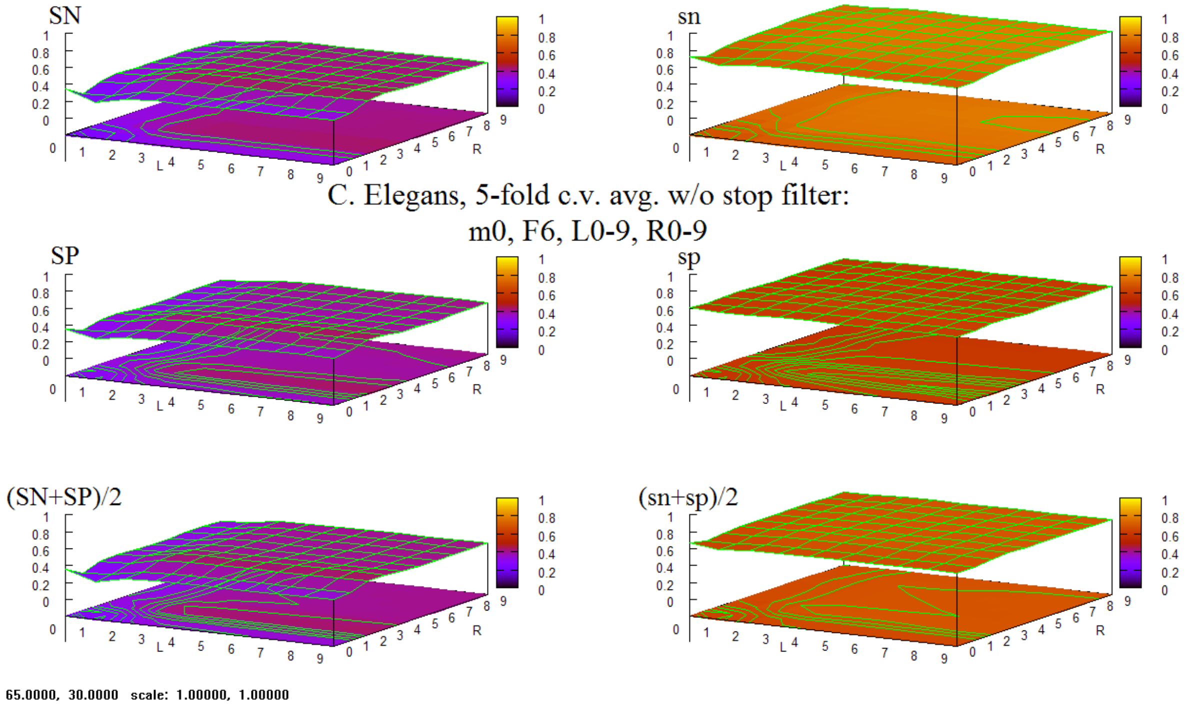


Figure 65 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=6, L=0-9,R=0-9

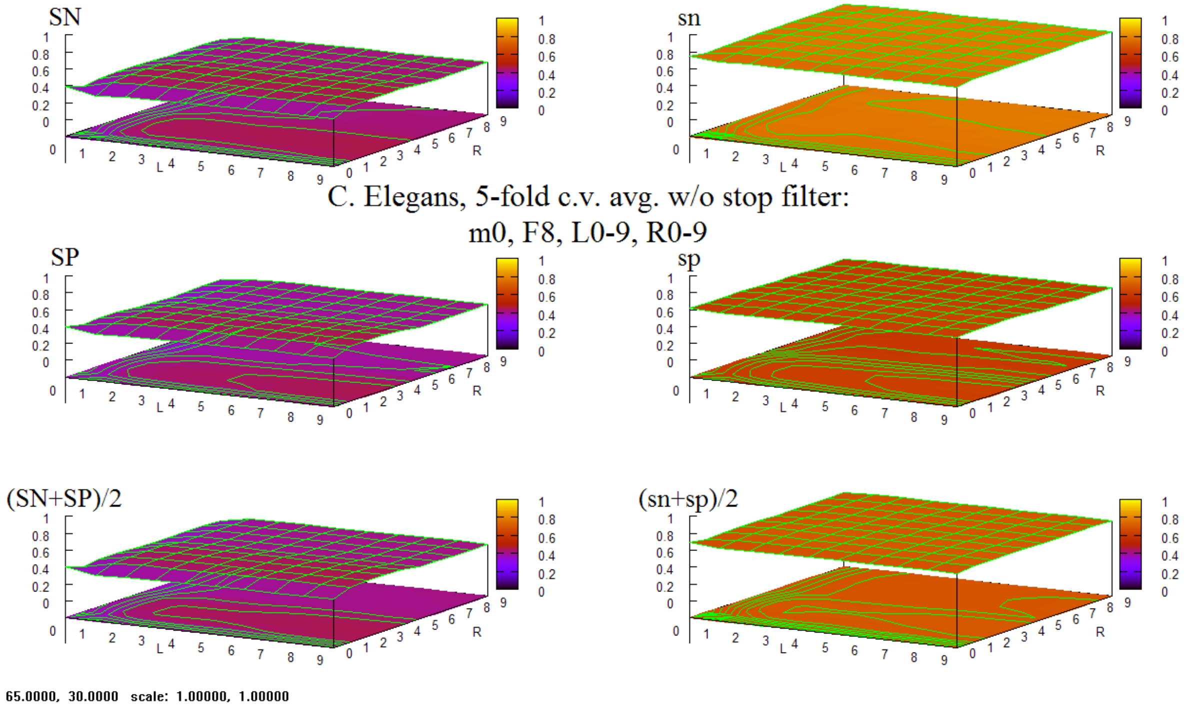


Figure 66 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=8, L=0-9,R=0-9

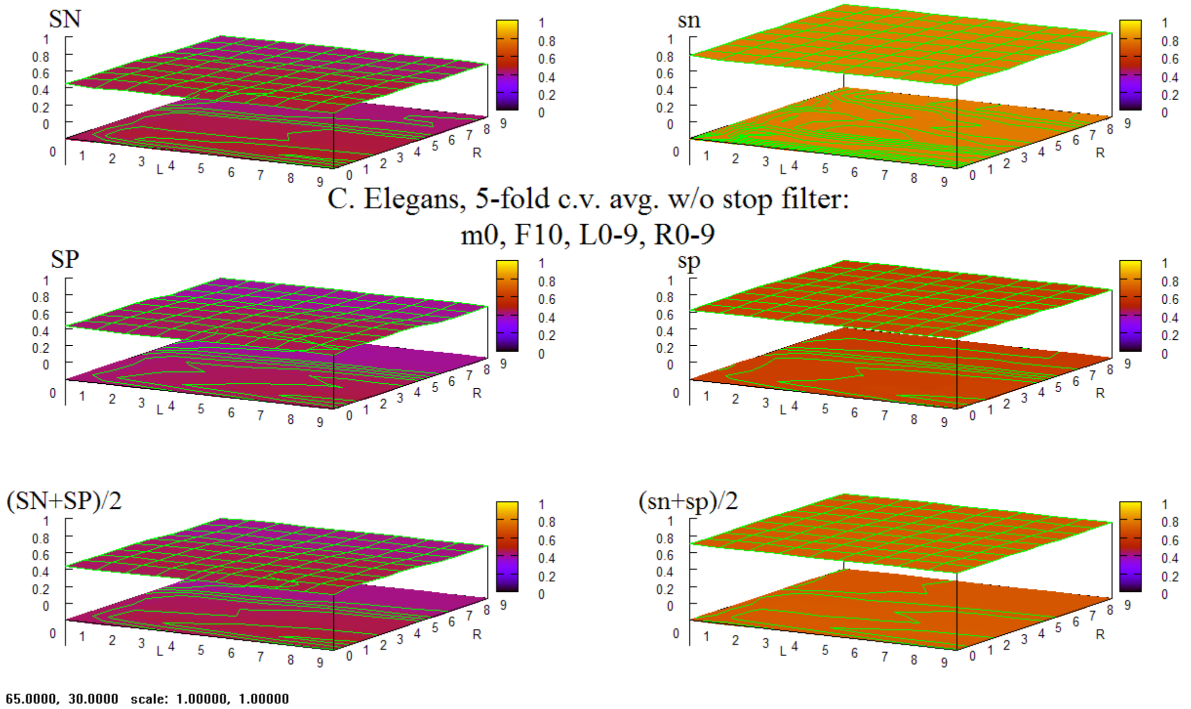


Figure 67 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=10, L=0-9,R=0-9

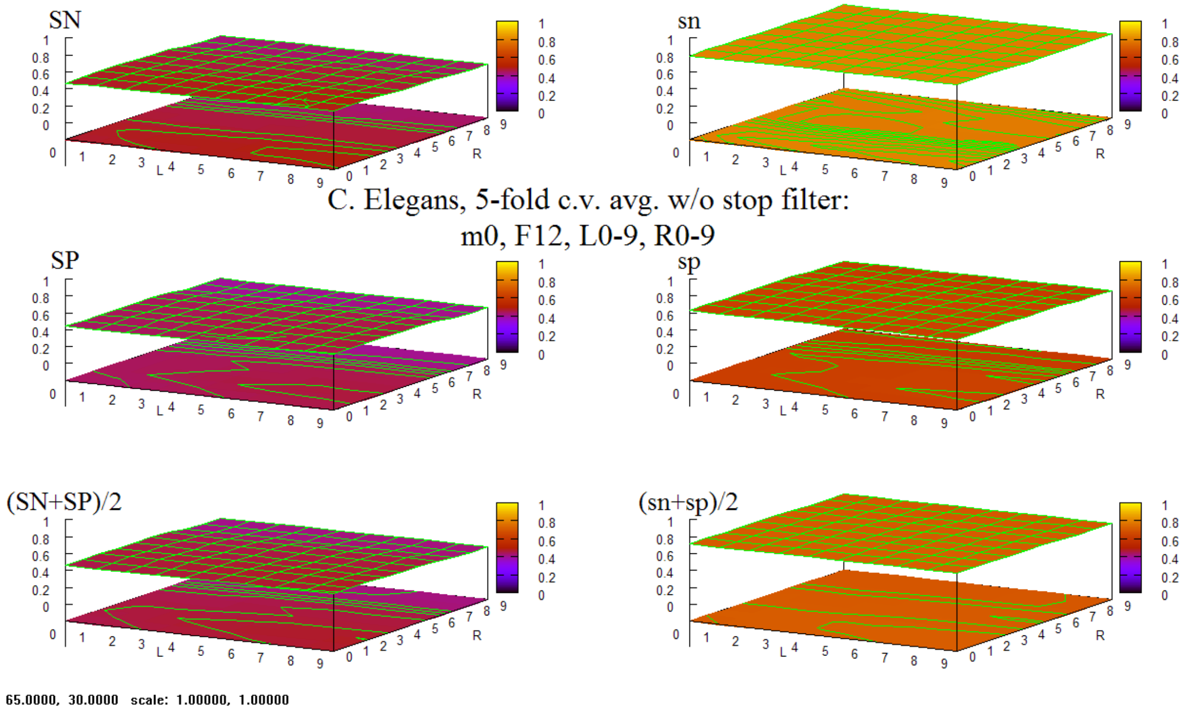


Figure 68 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=12, L=0-9,R=0-9

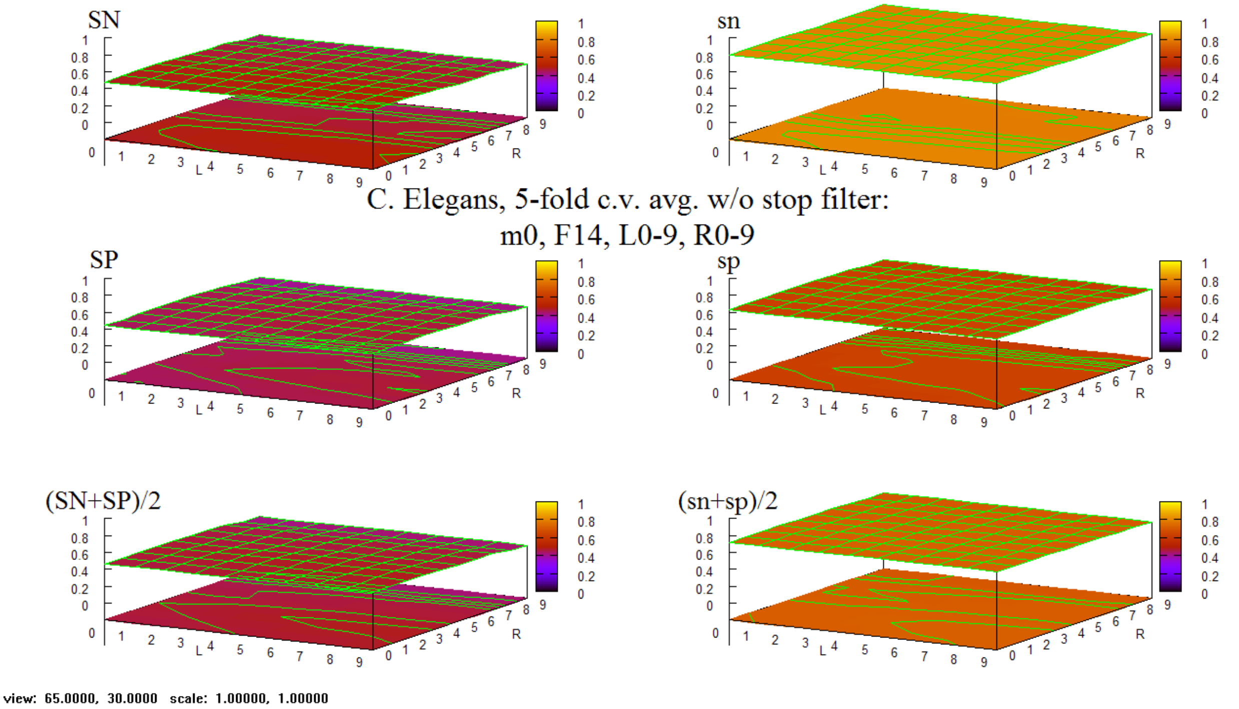


Figure 69 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=14, L=0-9,R=0-9

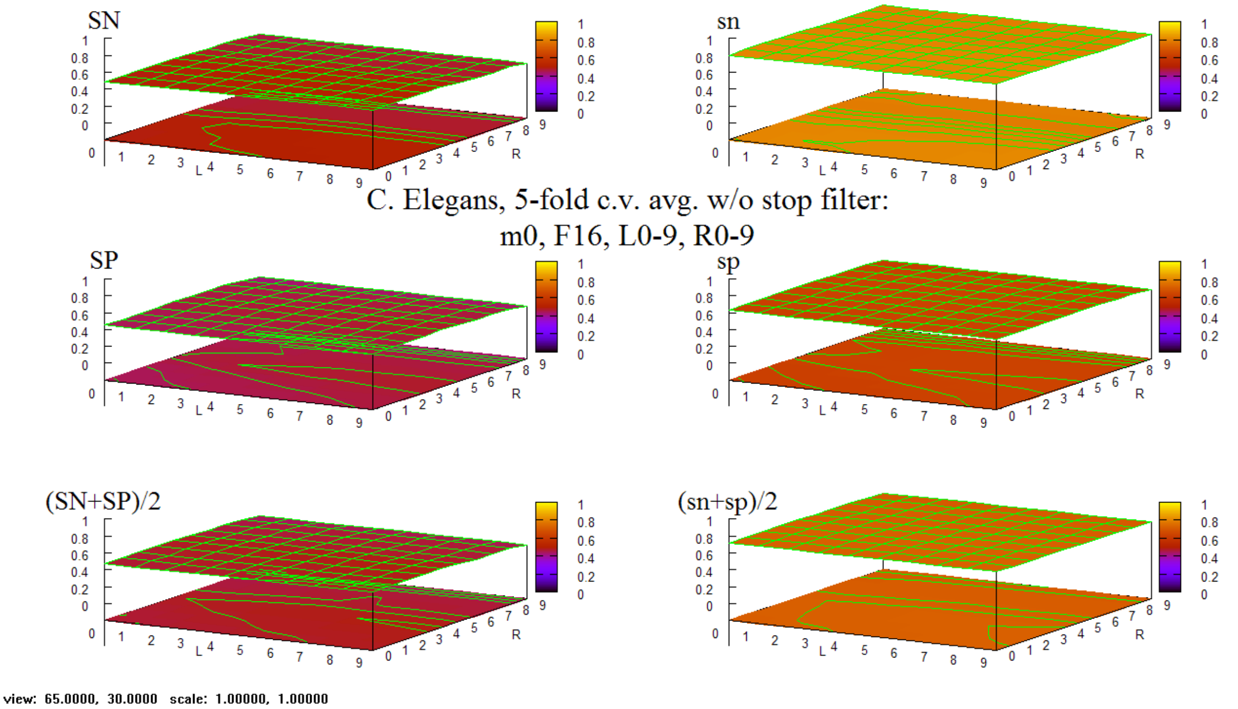


Figure 70 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=16, L=0-9,R=0-9

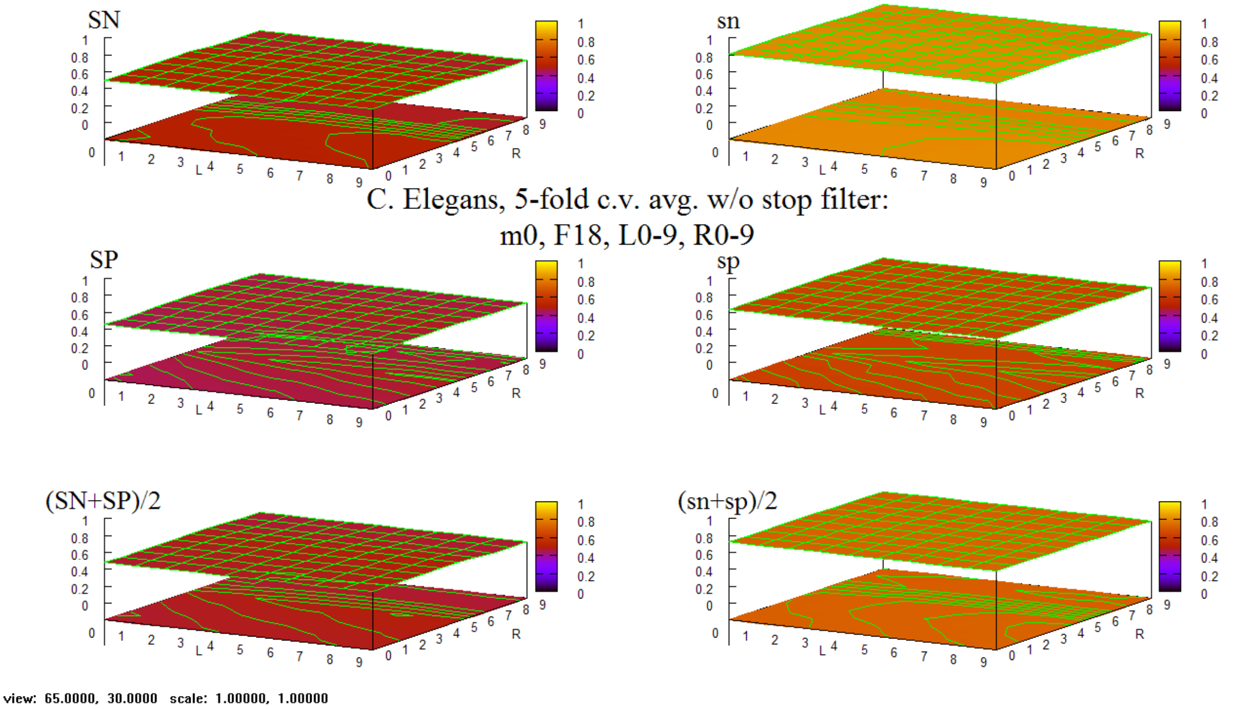


Figure 71 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=18, L=0-9,R=0-9

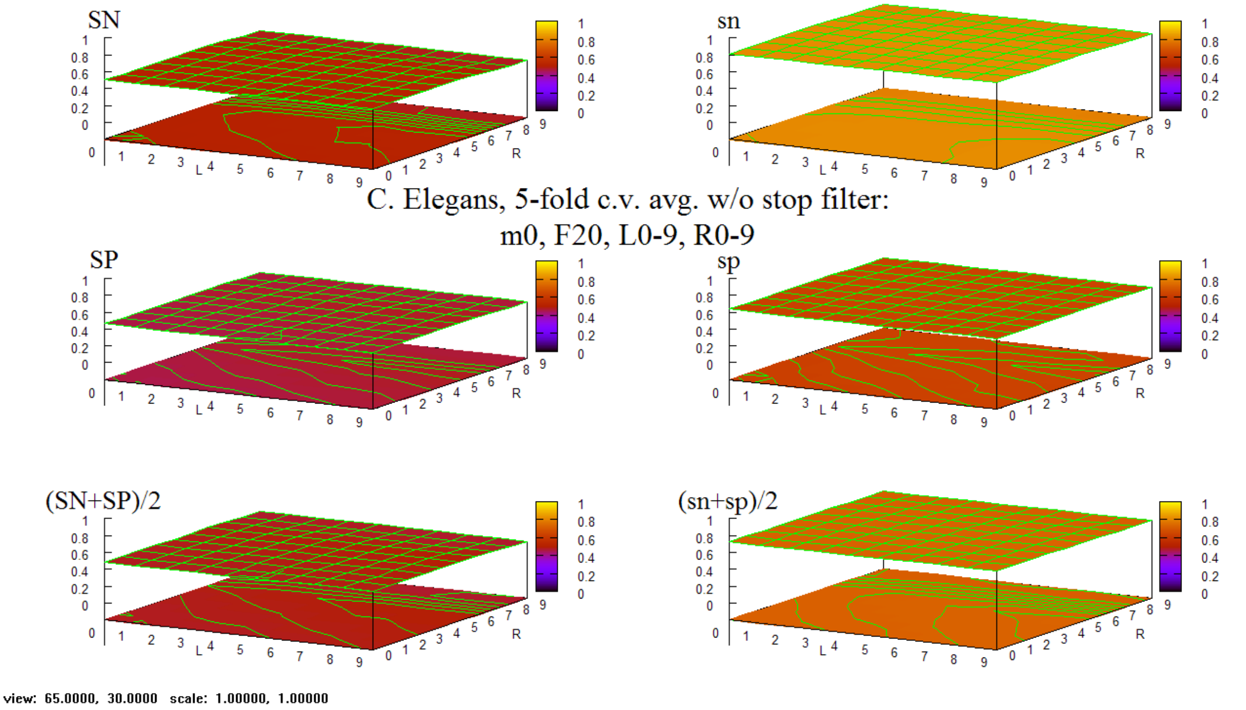


Figure 72 HOHMM performance using C.E. I-V (alt. splicing deleted) m=0, F=20, L=0-9,R=0-9

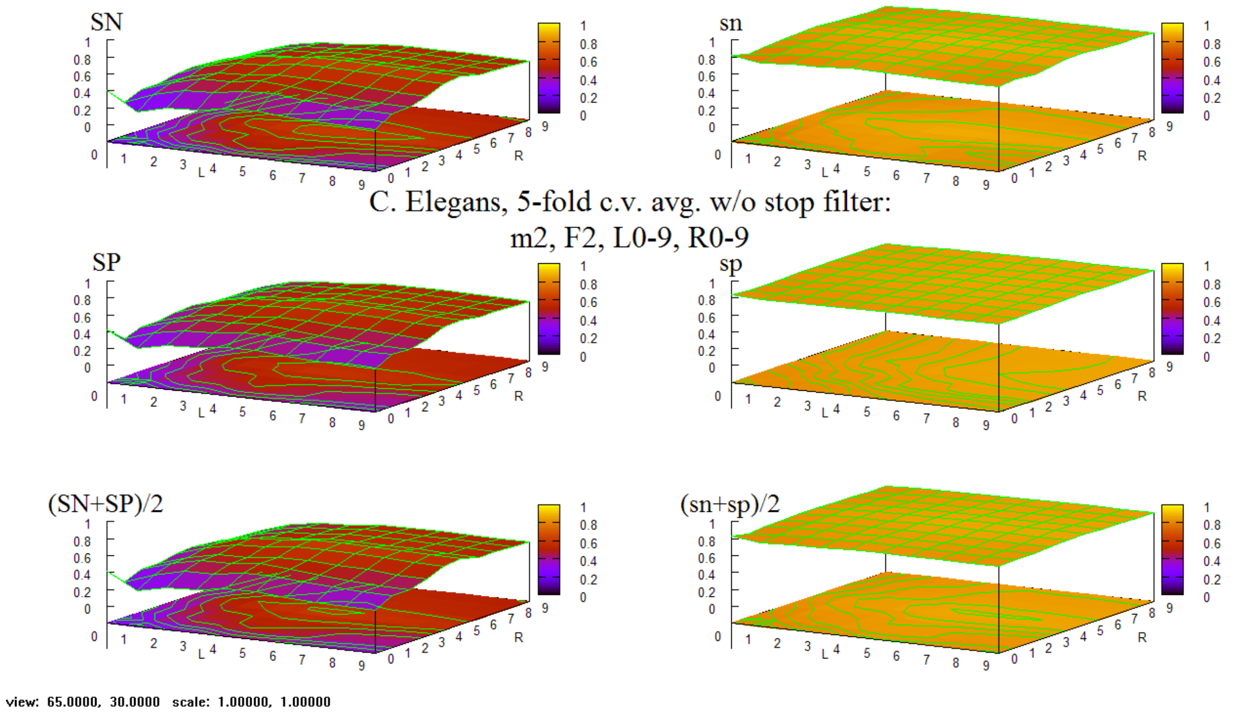


Figure 73 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=2, L=0-9,R=0-9

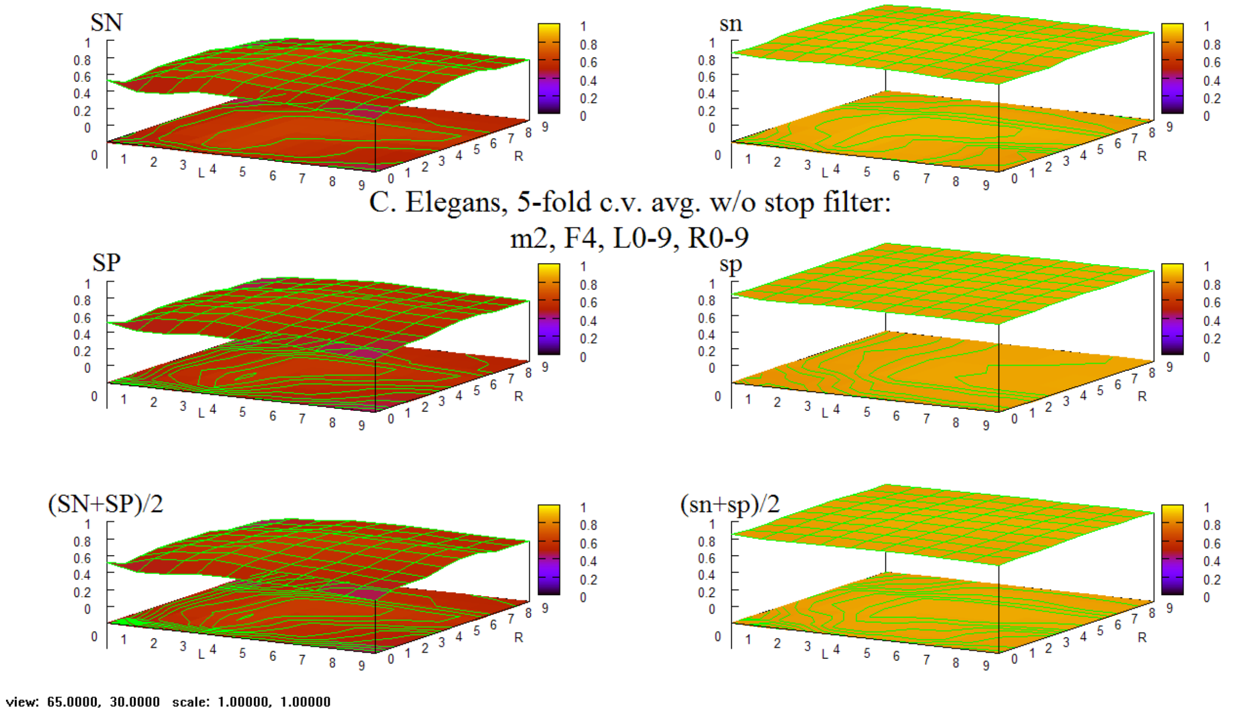


Figure 74 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=4, L=0-9,R=0-9

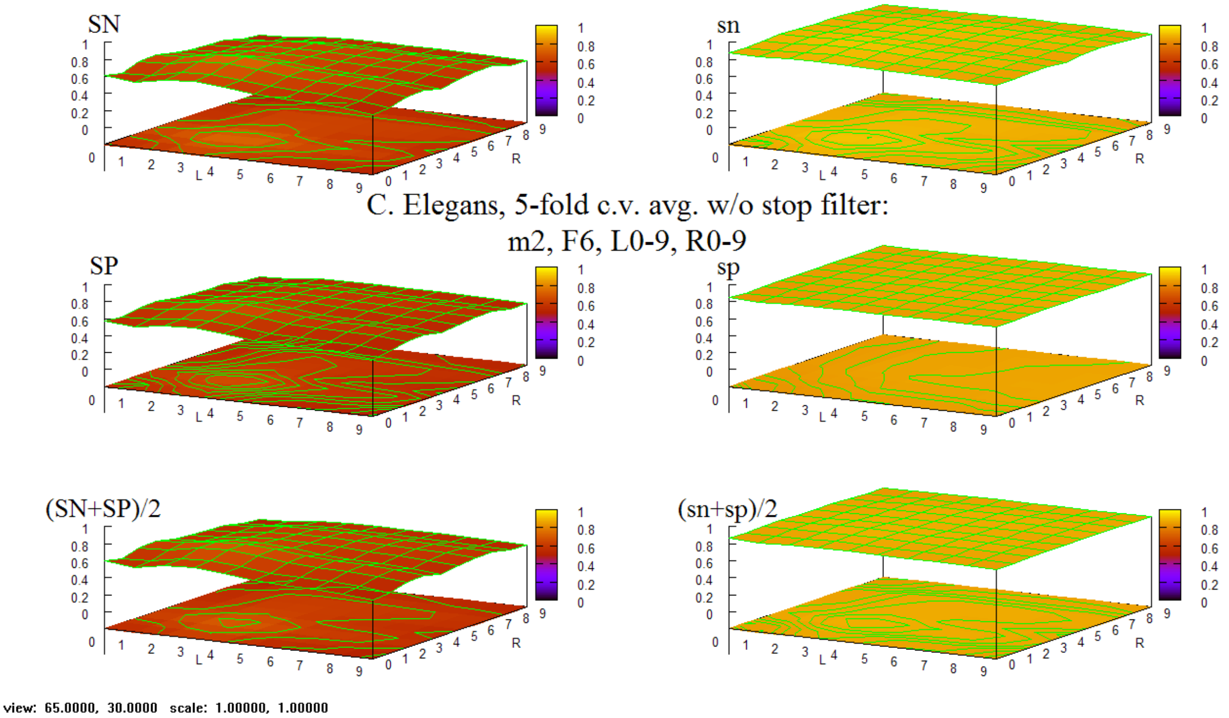


Figure 75 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=6, L=0-9,R=0-9

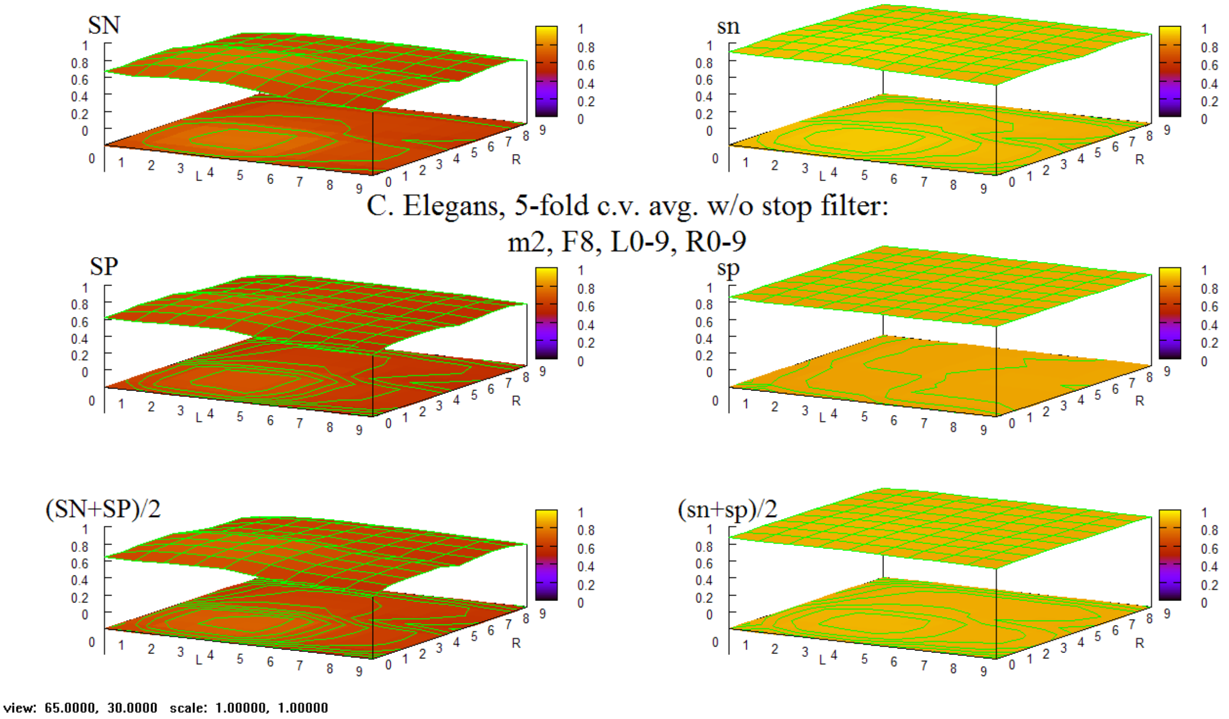


Figure 76 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=8, L=0-9,R=0-9

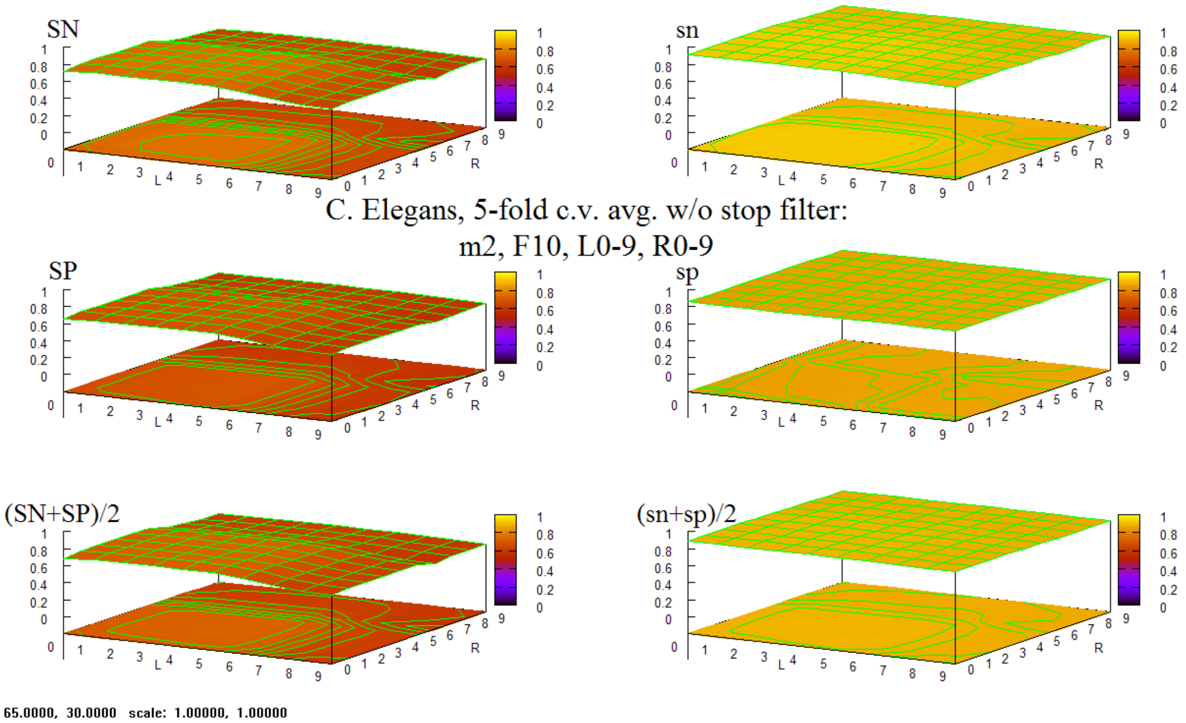


Figure 77 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=10, L=0-9,R=0-9

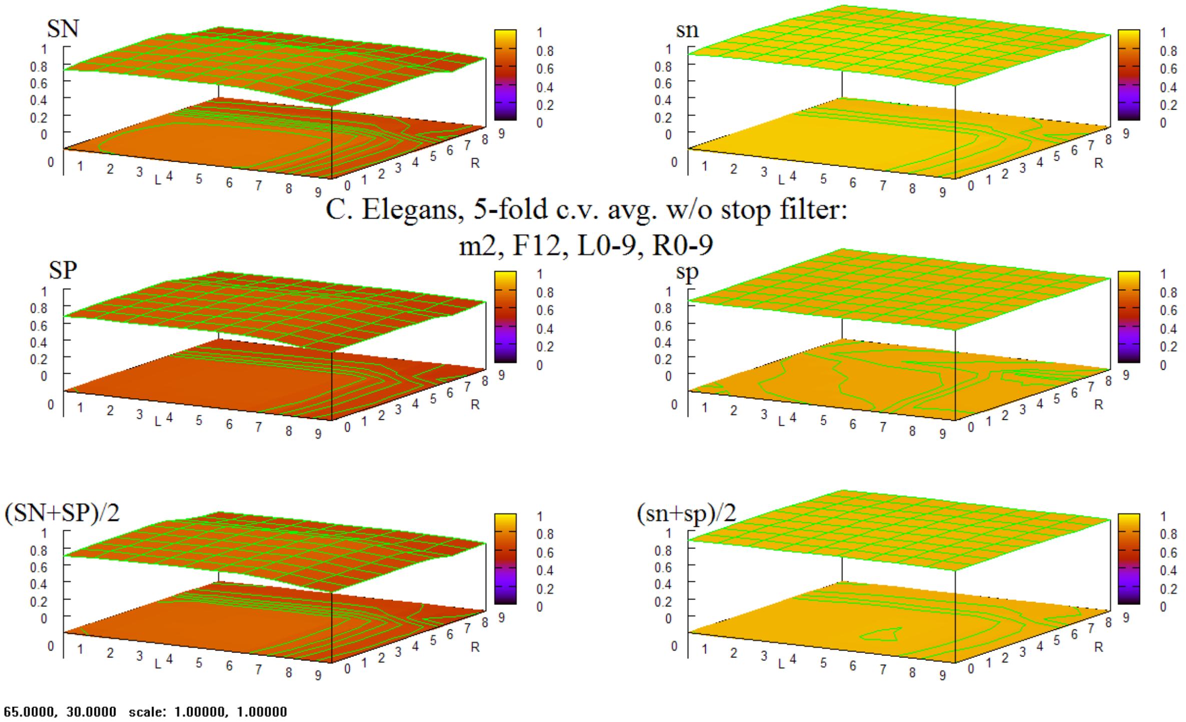


Figure 78 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=12, L=0-9,R=0-9

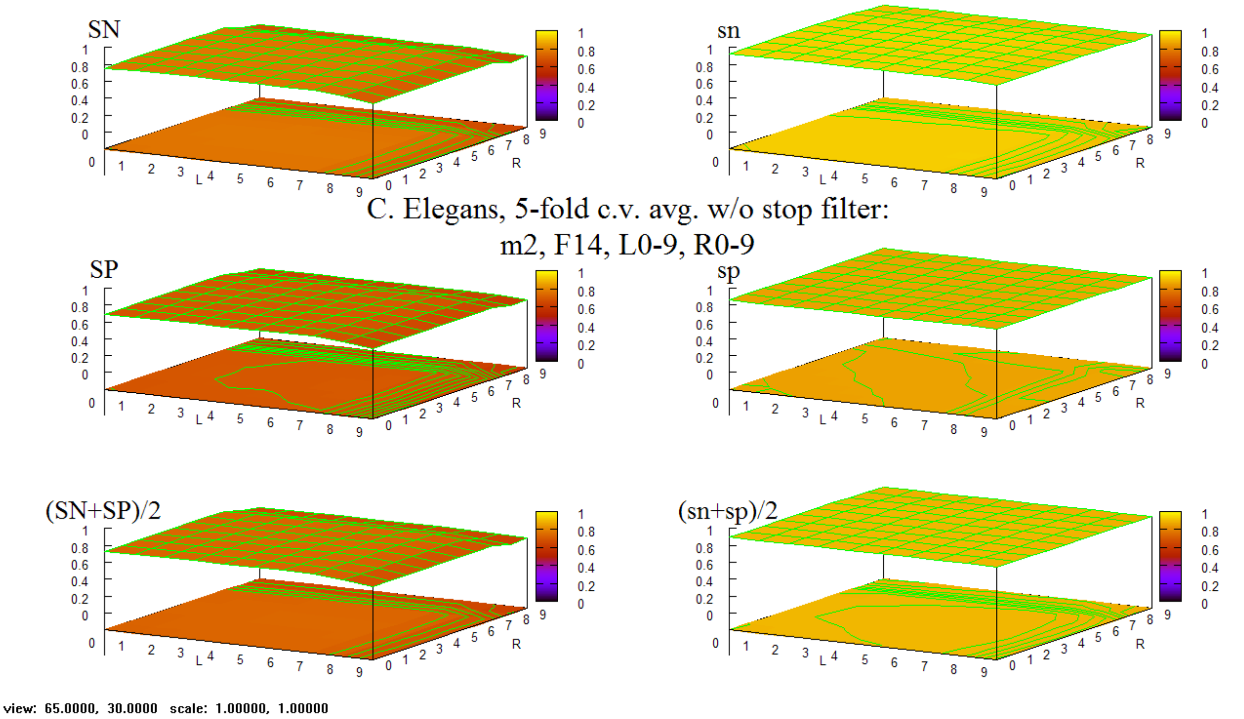


Figure 79 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=14, L=0-9,R=0-9

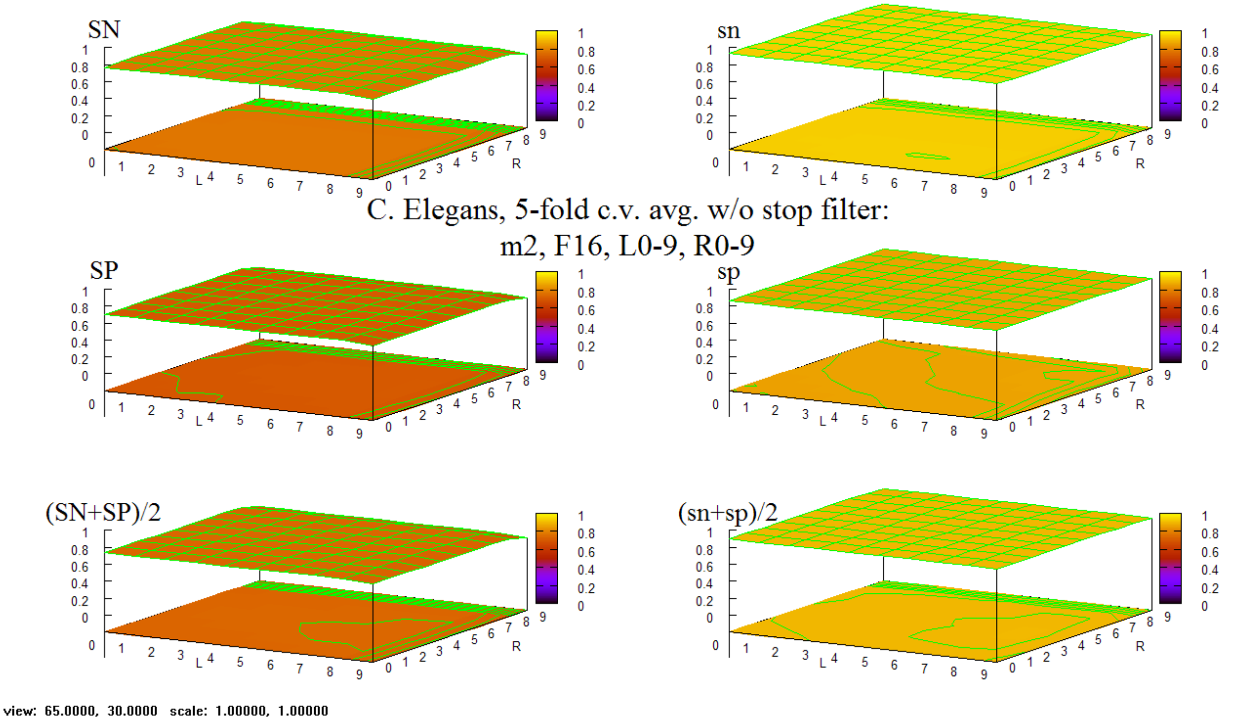


Figure 80 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=16, L=0-9,R=0-9

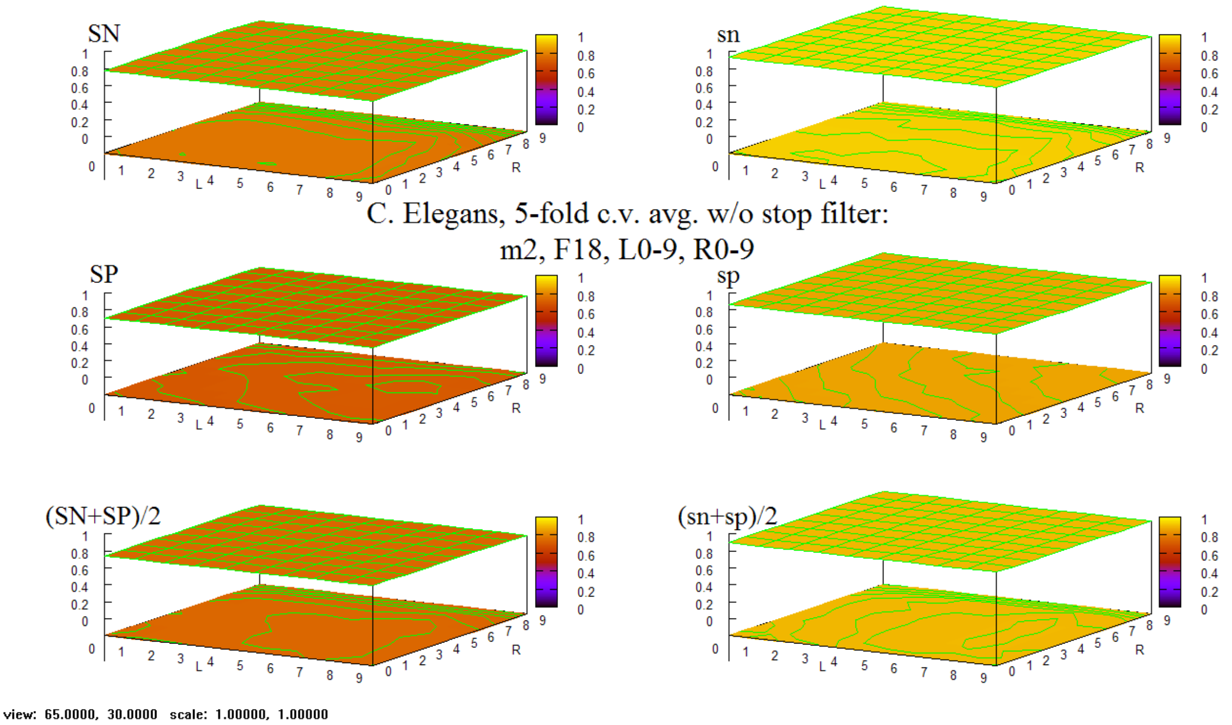


Figure 81 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=18, L=0-9,R=0-9

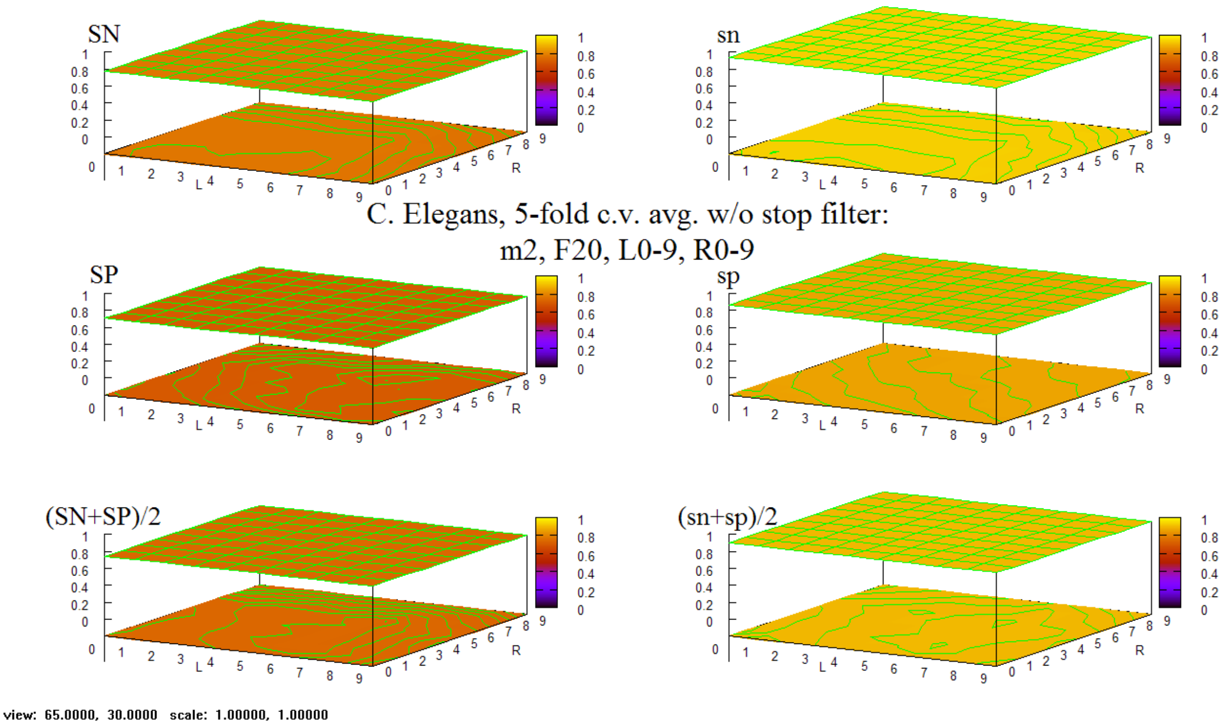


Figure 82 HOHMM performance using C.E. I-V (alt. splicing deleted) m=2, F=20, L=0-9,R=0-9

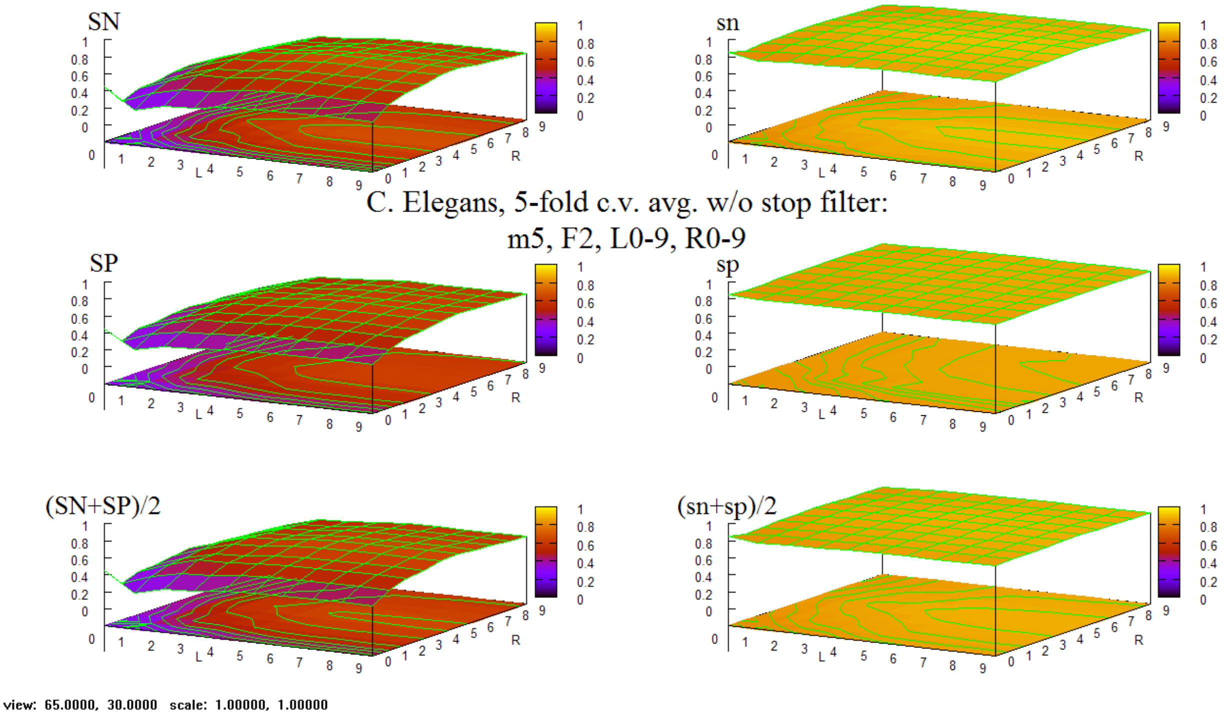


Figure 83 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=2, L=0-9,R=0-9

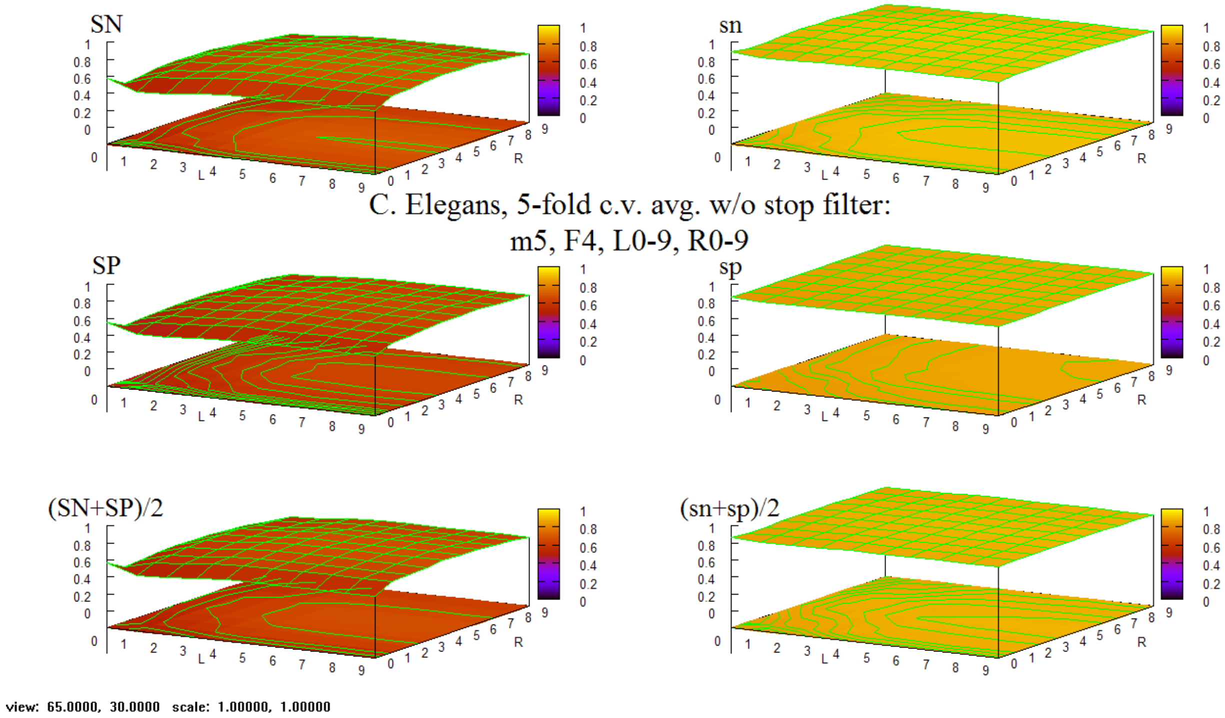


Figure 84 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=4, L=0-9,R=0-9

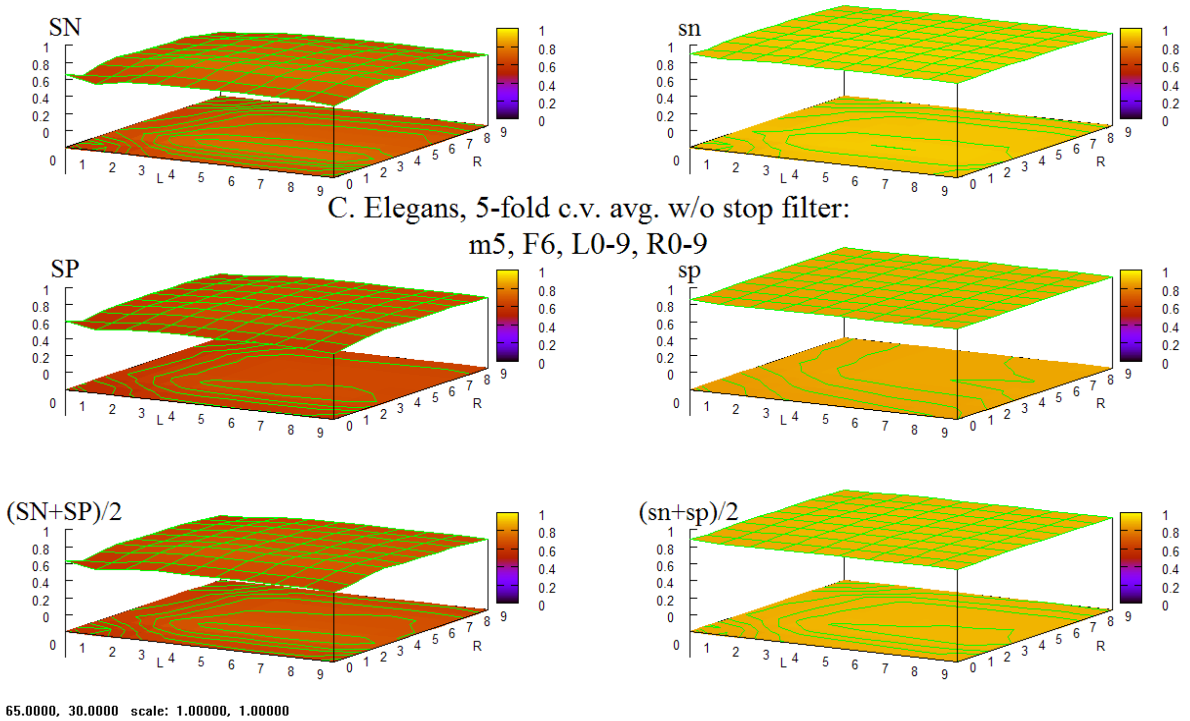


Figure 85 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=6, L=0-9,R=0-9

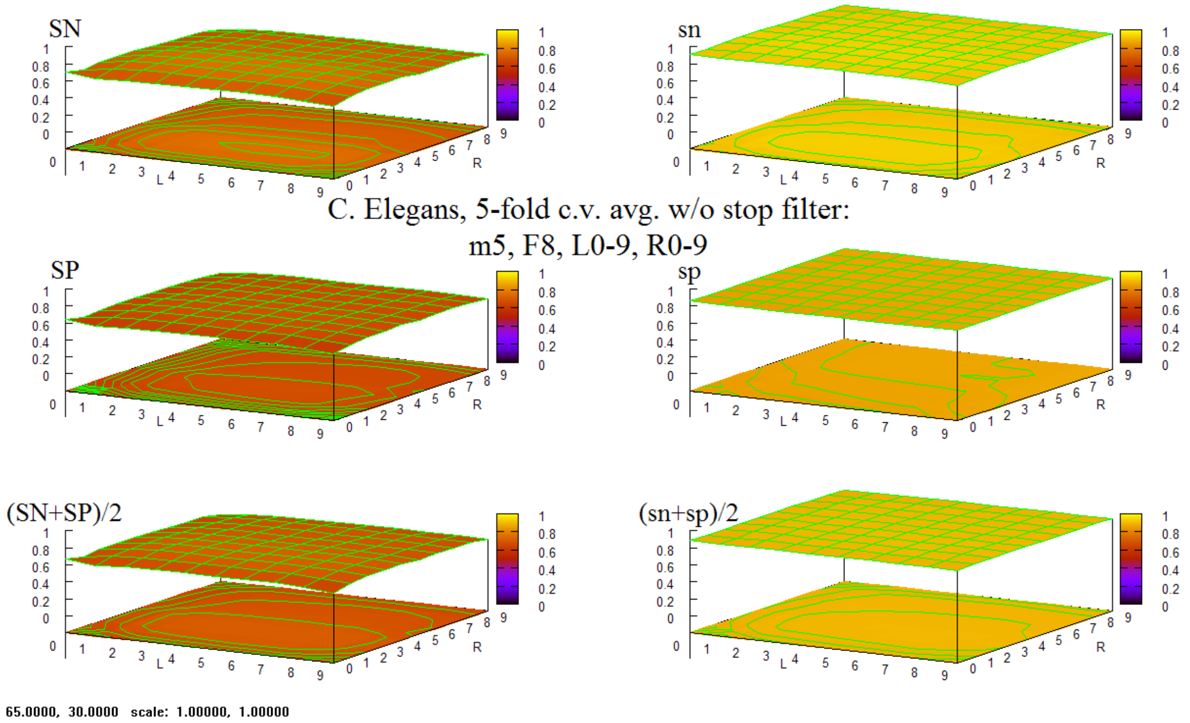


Figure 86 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=8, L=0-9,R=0-9

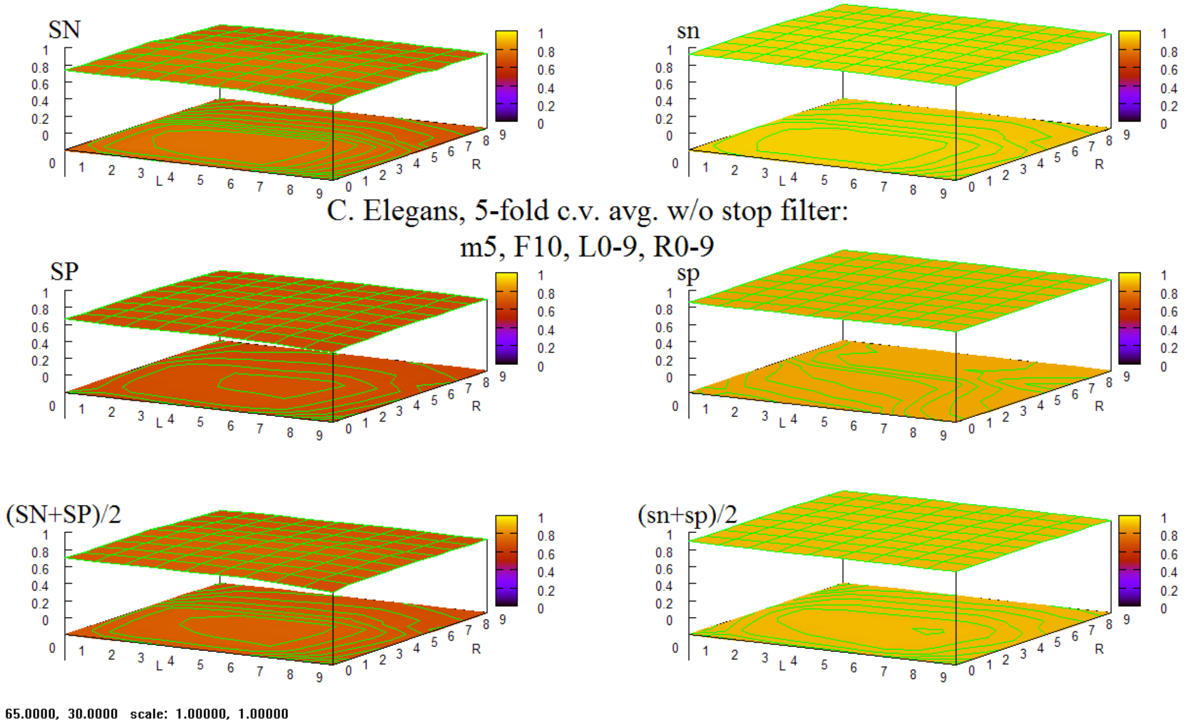


Figure 87 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=10, L=0-9,R=0-9

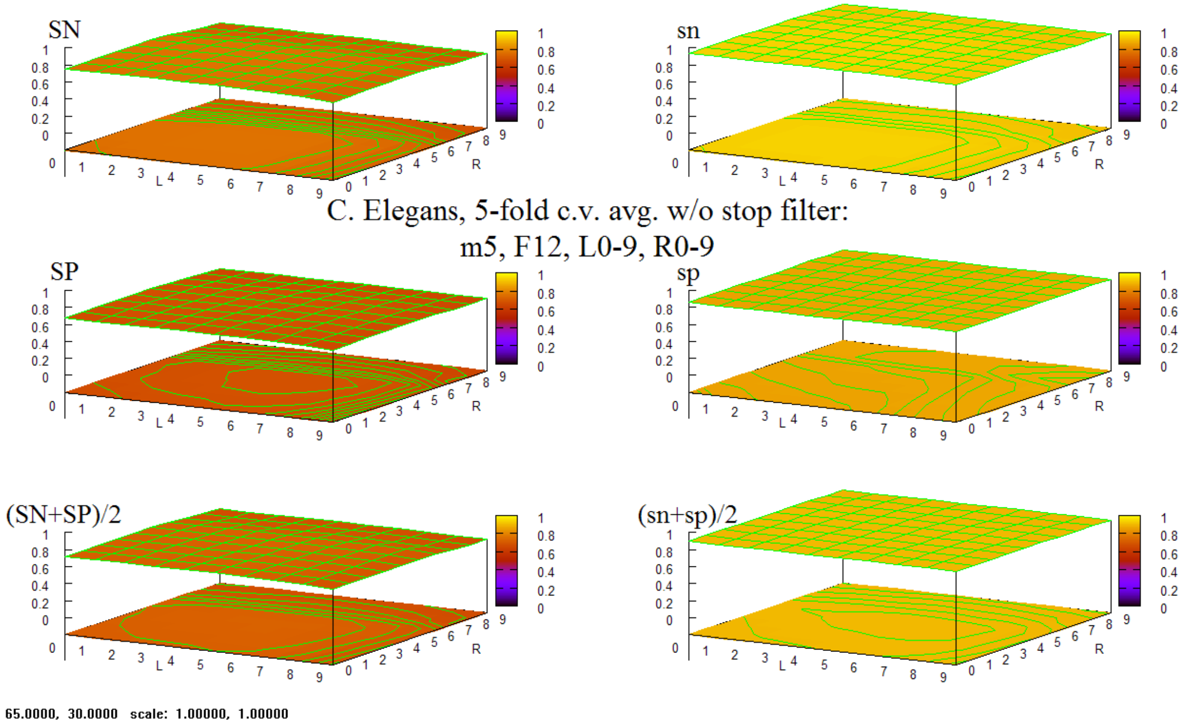


Figure 88 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=12, L=0-9,R=0-9

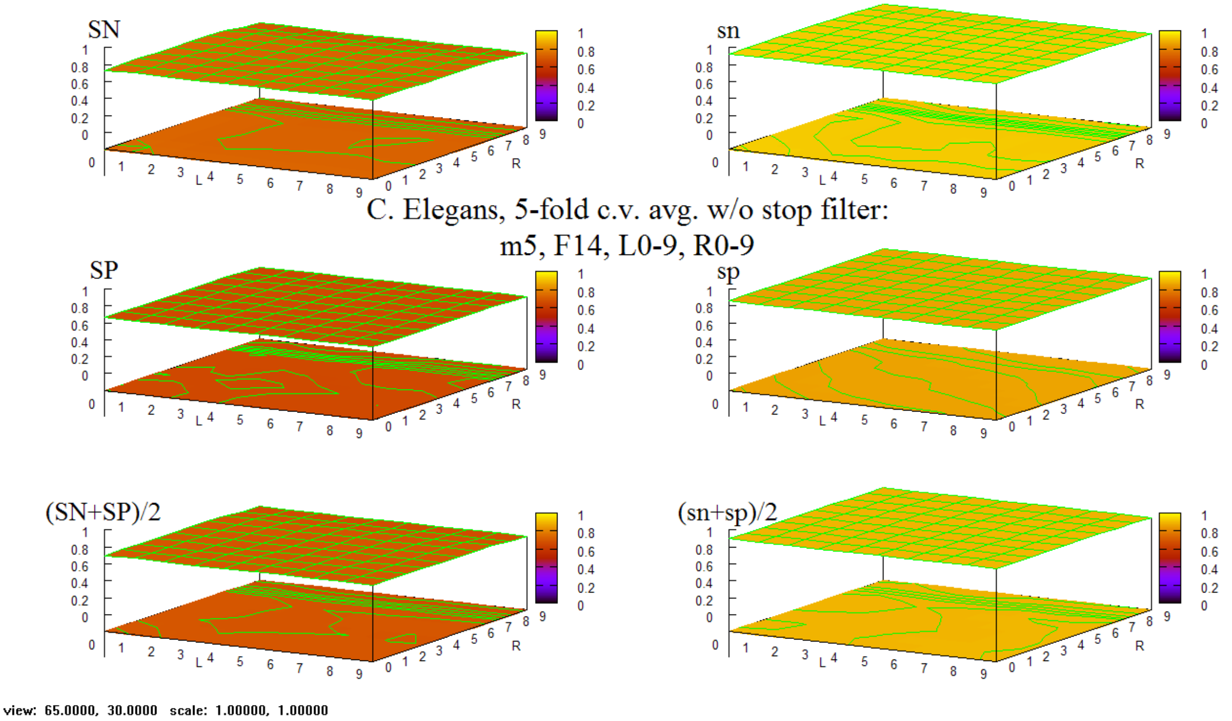


Figure 89 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=14, L=0-9,R=0-9

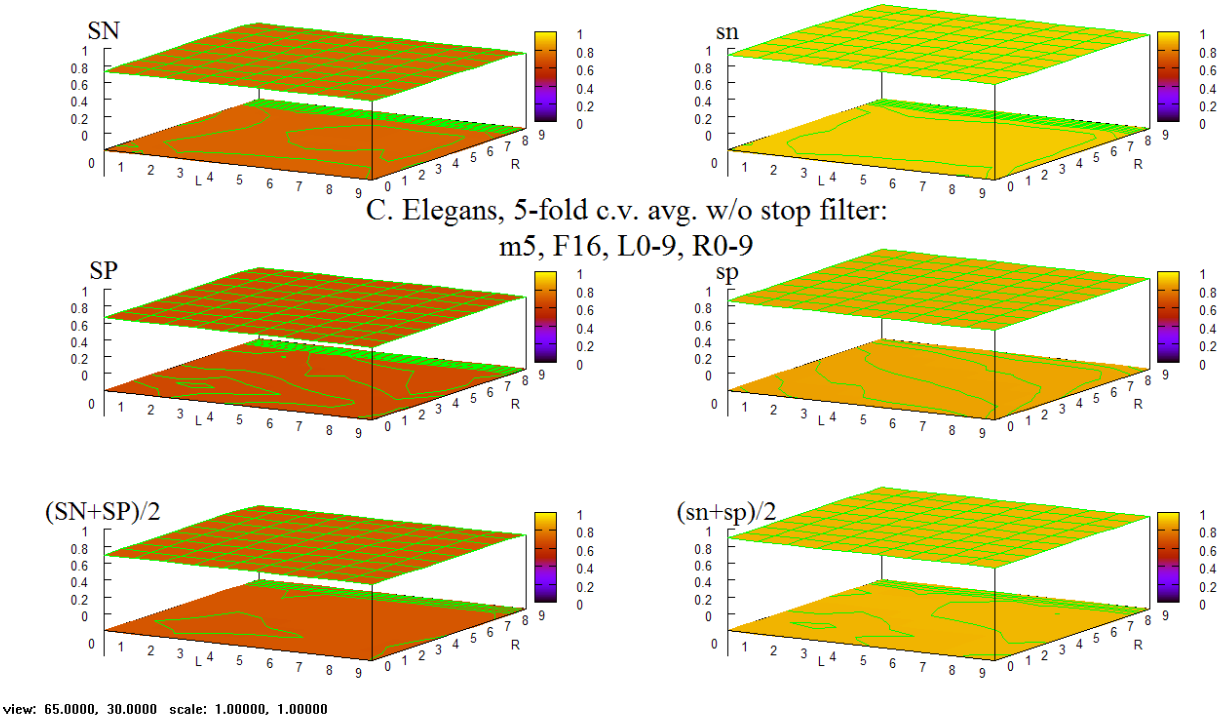


Figure 90 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=16, L=0-9,R=0-9

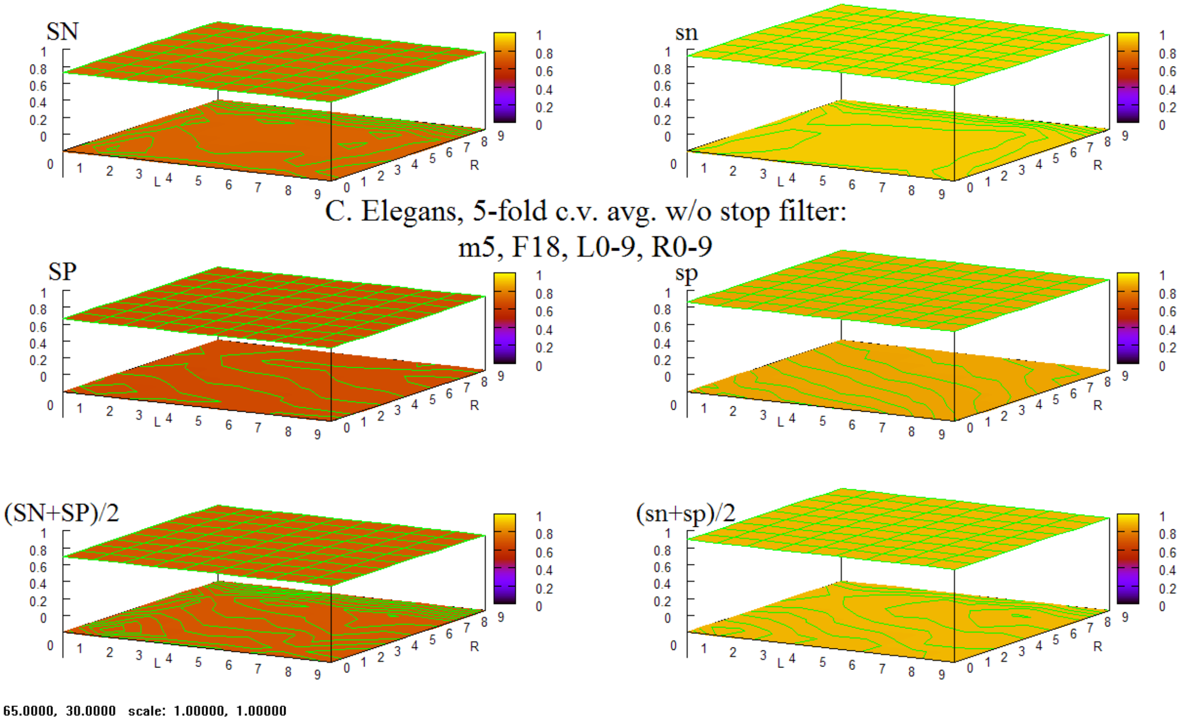


Figure 91 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=18, L=0-9,R=0-9

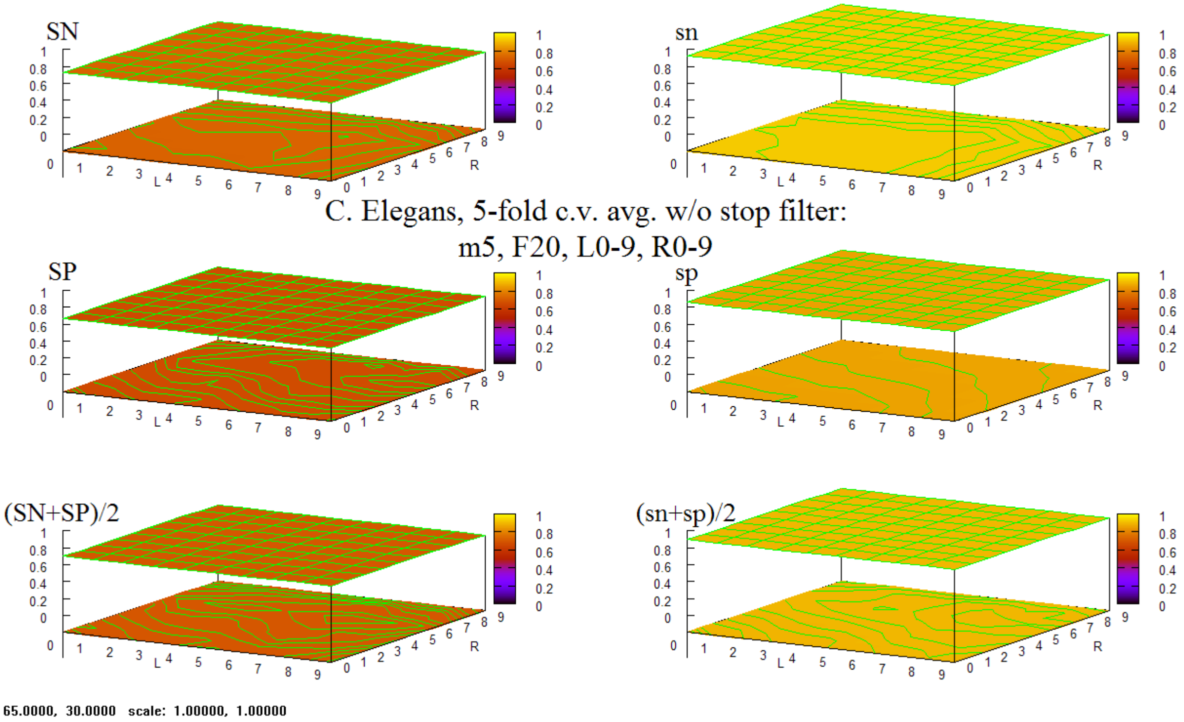


Figure 92 HOHMM performance using C.E. I-V (alt. splicing deleted) m=5, F=20, L=0-9,R=0-9

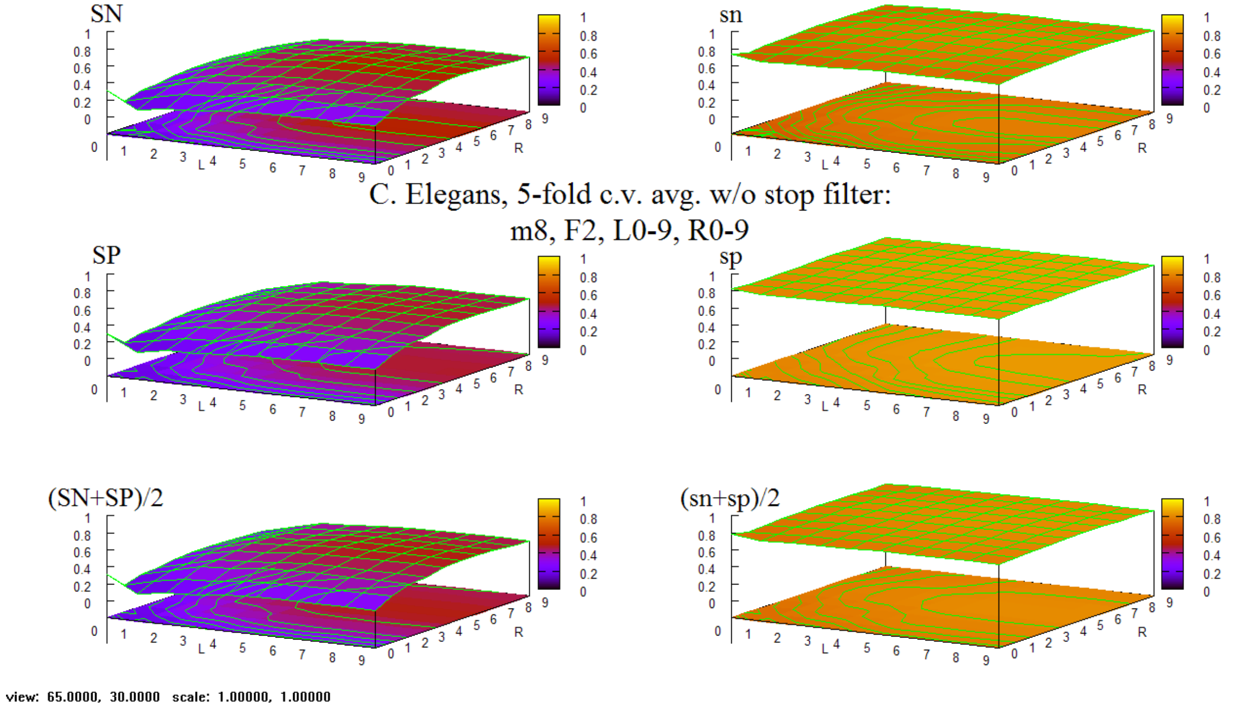


Figure 93 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=2, L=0-9,R=0-9

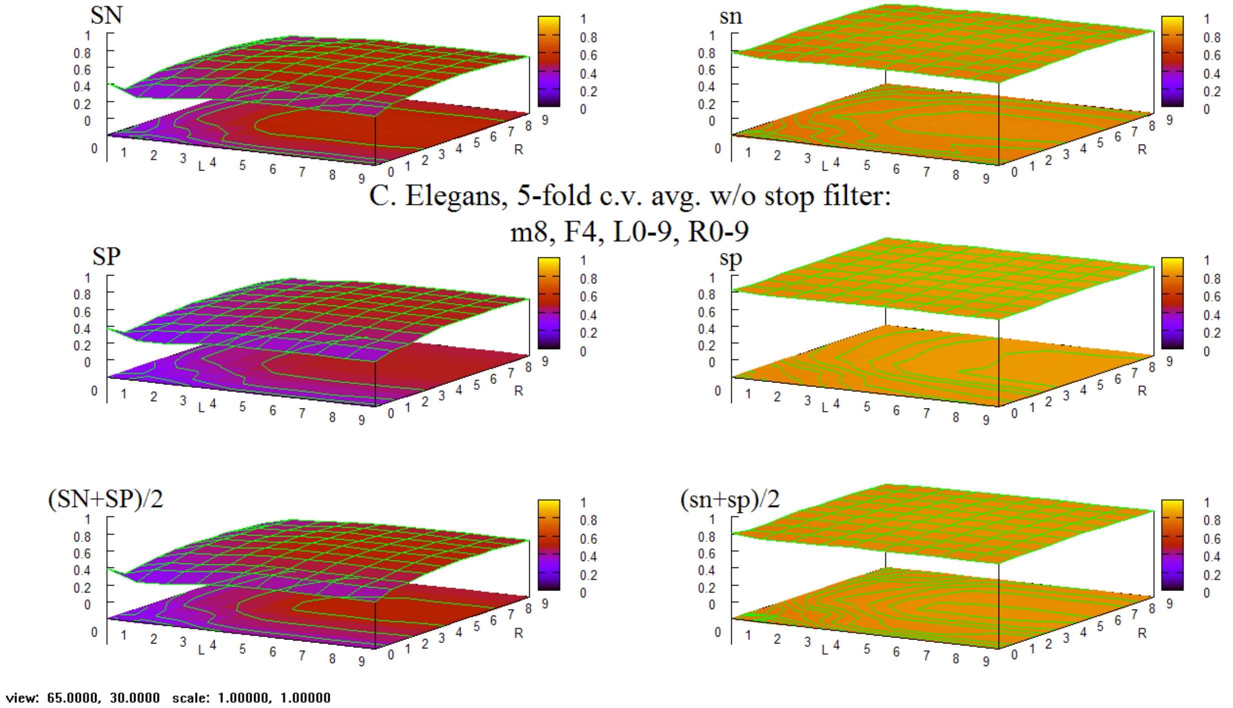


Figure 94 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=4, L=0-9,R=0-9

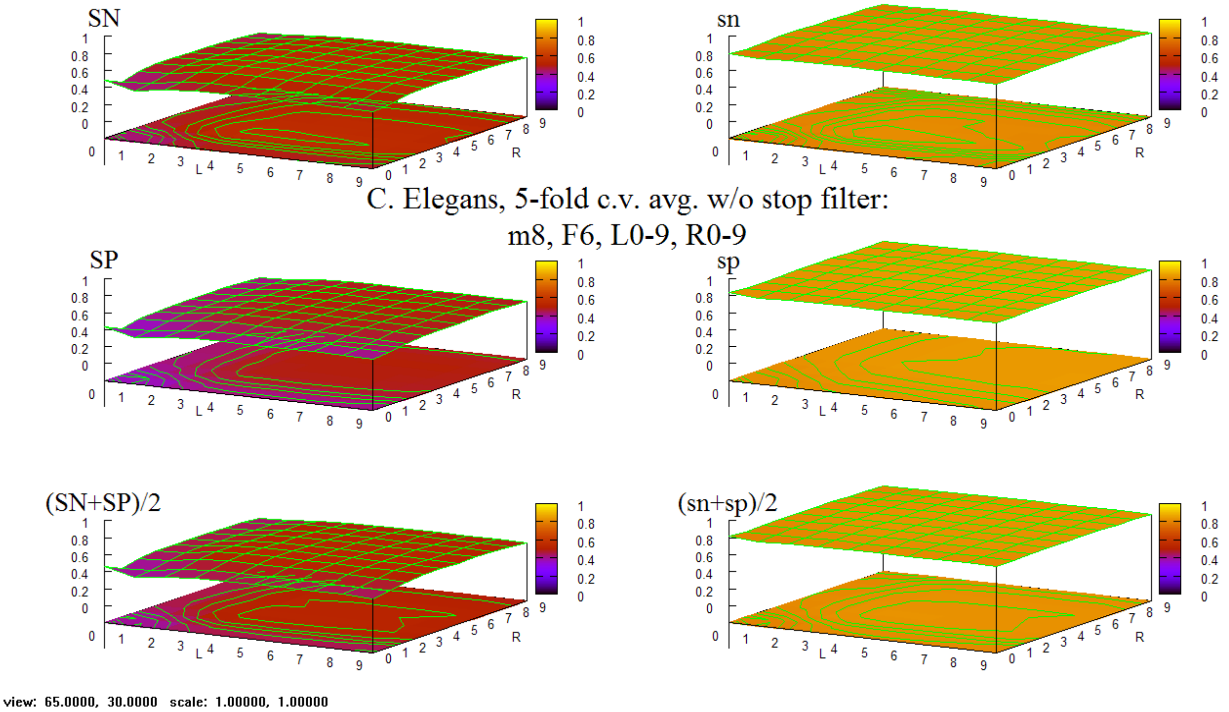


Figure 95 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=6, L=0-9,R=0-9

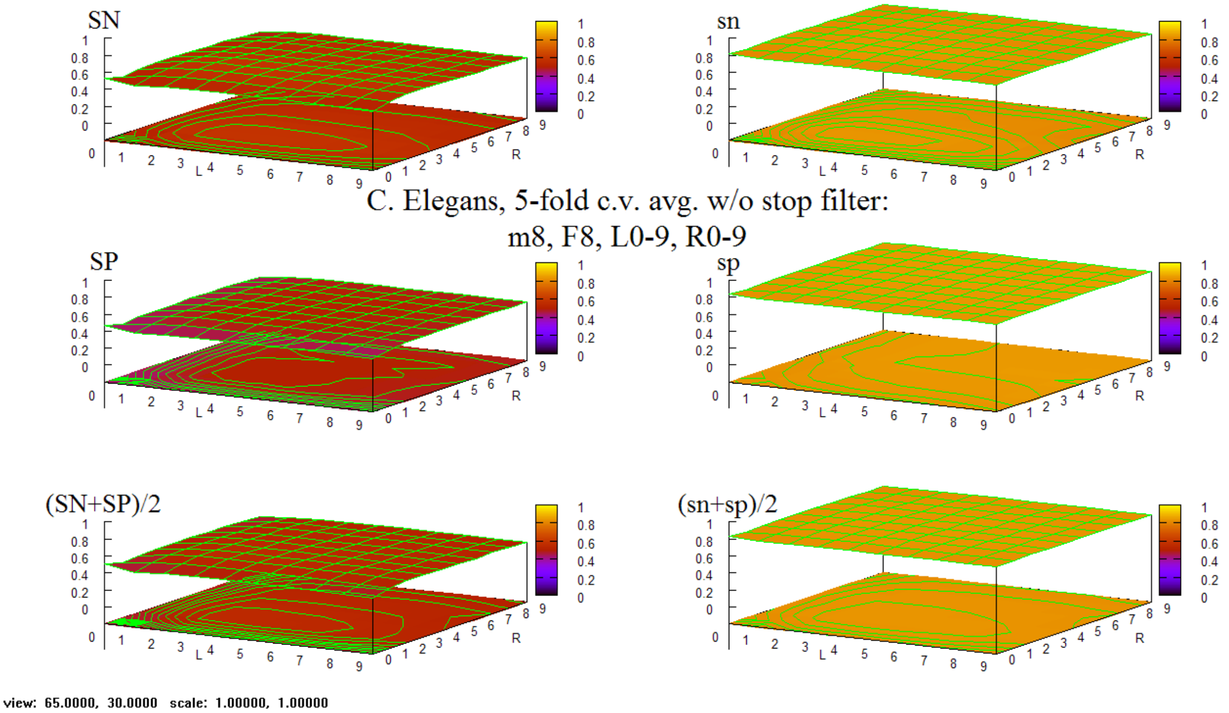


Figure 96 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=8, L=0-9,R=0-9

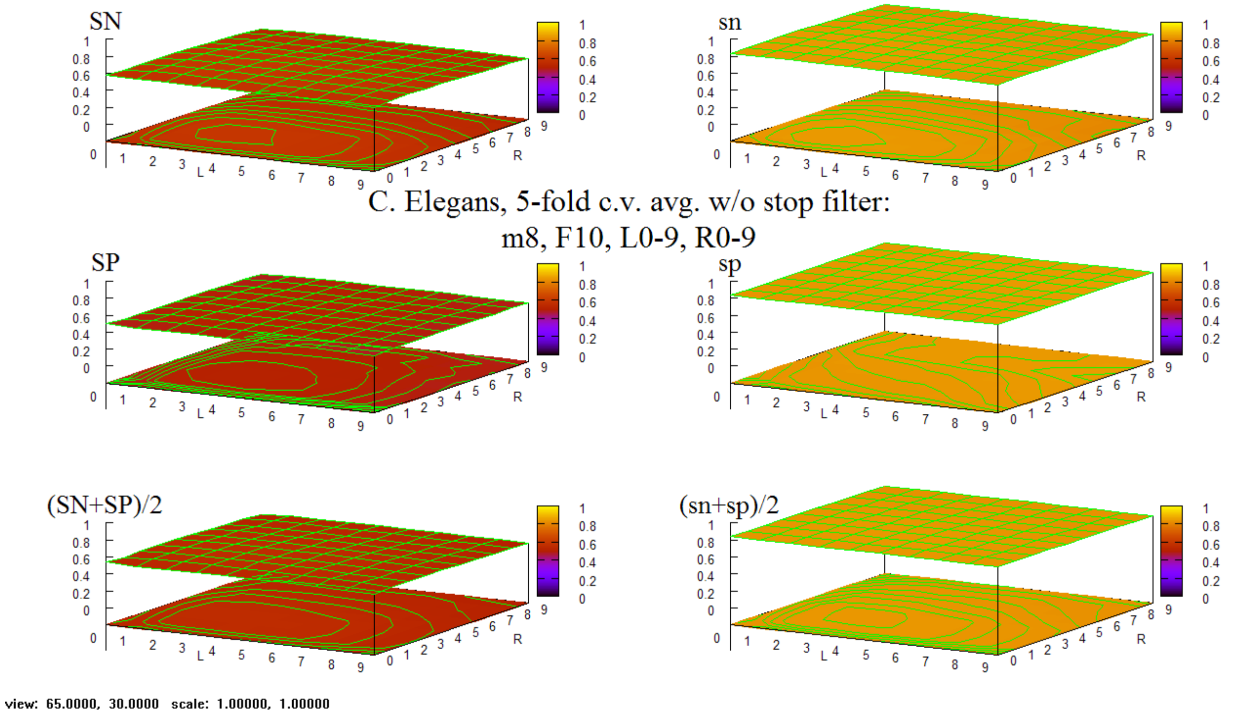


Figure 97 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=10, L=0-9,R=0-9

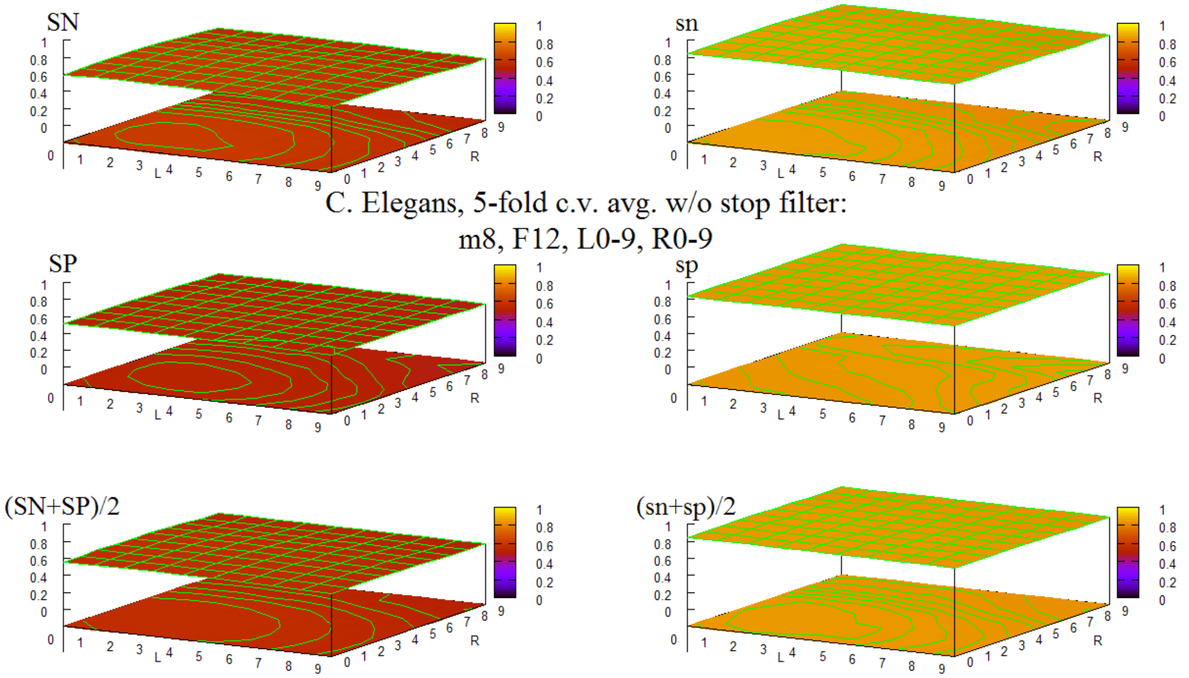


Figure 98 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=12, L=0-9,R=0-9

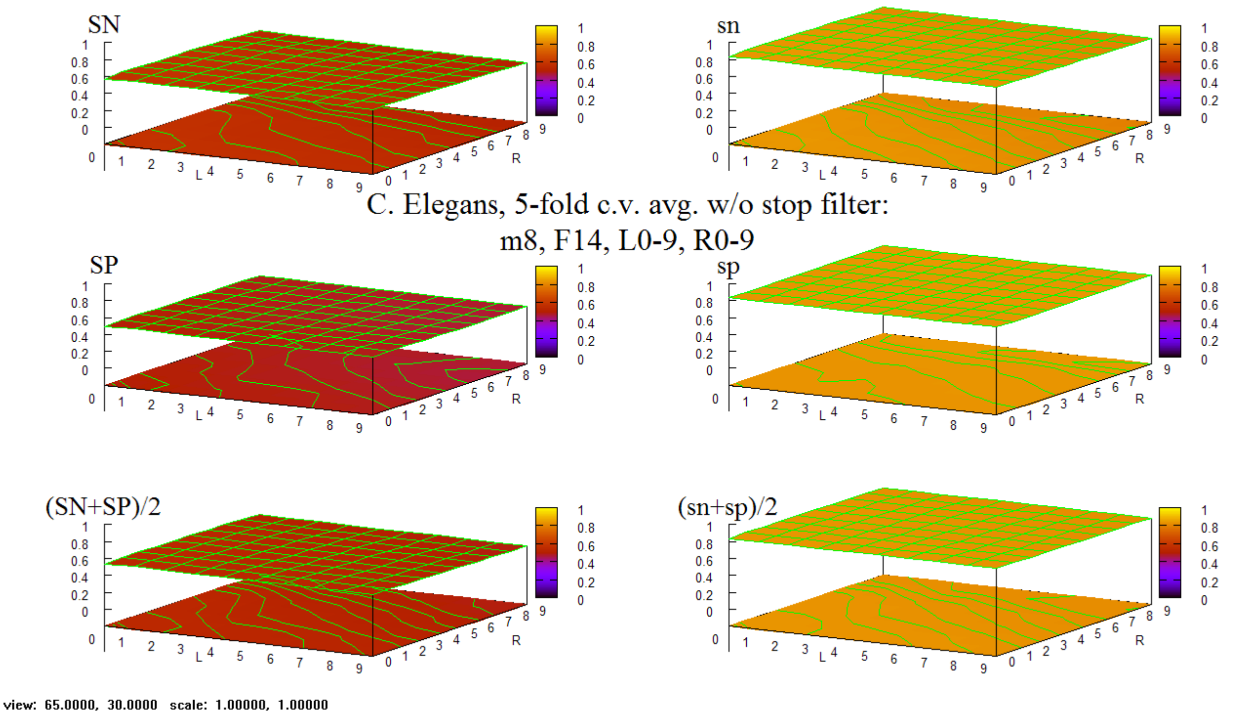


Figure 99 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=14, L=0-9,R=0-9

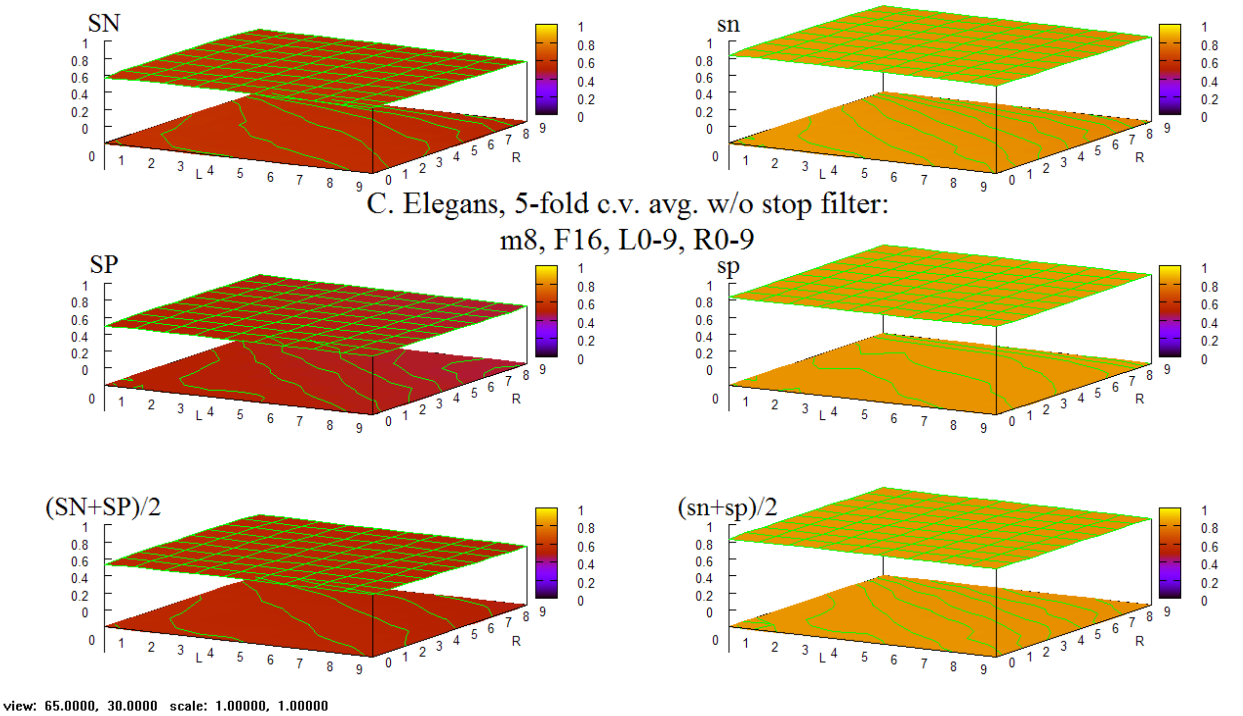


Figure 100 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=16, L=0-9,R=0-9

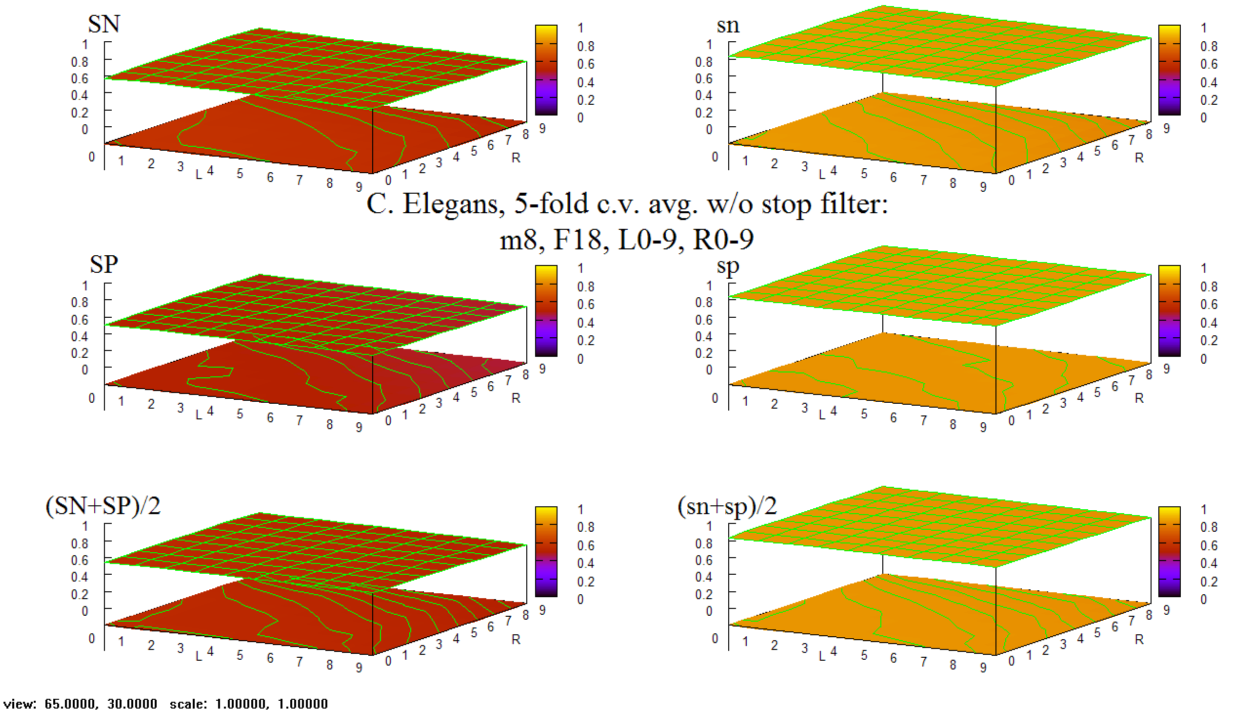


Figure 101 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=18, L=0-9,R=0-9

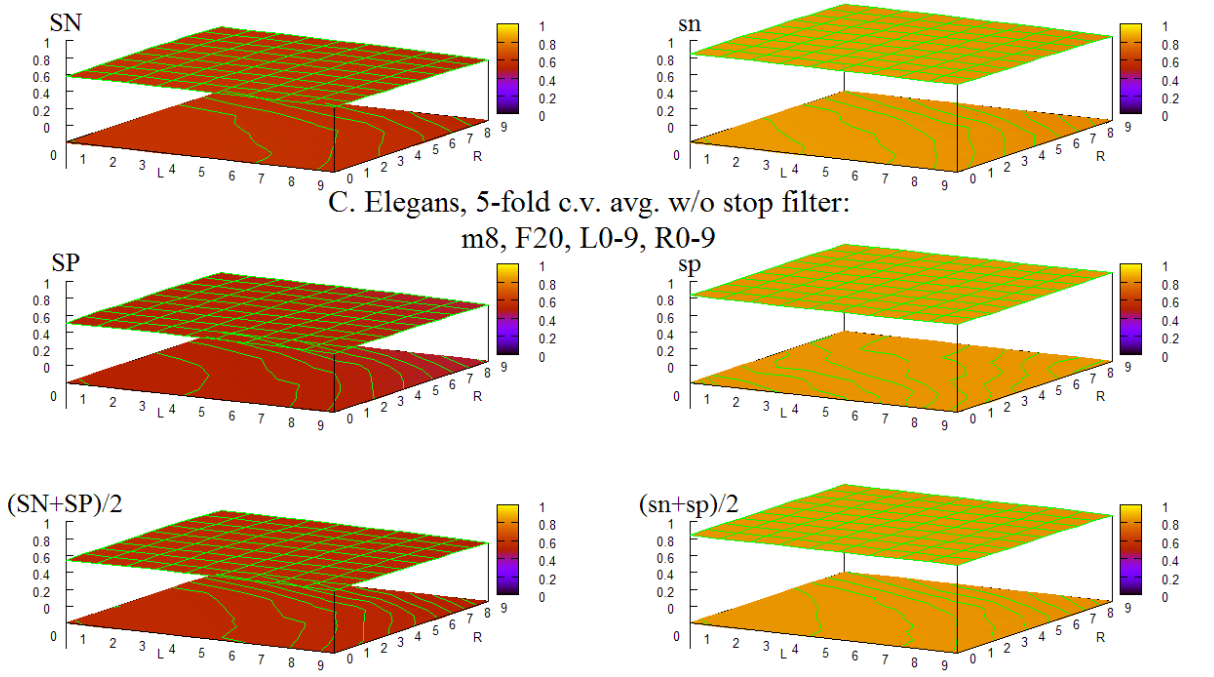


Figure 102 HOHMM performance using C.E. I-V (alt. splicing deleted) m=8, F=20, L=0-9,R=0-9

8.4 Detailed Results of Relative Entropy for Chromosomes I-V of *C. elegans*

In Figure 103 - Figure 118 below, we show the results of the computations of relative entropy for the reduced data set of Chromosomes I-V of *C. elegans* for a sampling window of size 40. In general, the periodicity of length 3 observed among the curves is an indication of the relative strengths of dependency within base strings of sizes 1, 3, 6, and 9 as such between the respective pairs of corresponding eij and xx probability models, for example, the model for je_0 corresponding to that of jj.

Each pair of figures (with 2 different plotting scales per pair) represent the relative entropy values for one of the 8 different eij-dimer types, je_0, e_0i_0, e_1i_1, e_2i_2, i_0e_1, i_1e_2, i_2e_0, and e_2j. The first graph in each pair of graphs uses the scaling factor $\log_{10}(4^{M+1})$, where M is the base Markov order with the chosen values 0, 2, 5, and 8. This value is the log of the maximum absolute entropy for a distribution of base strings of length M+1.

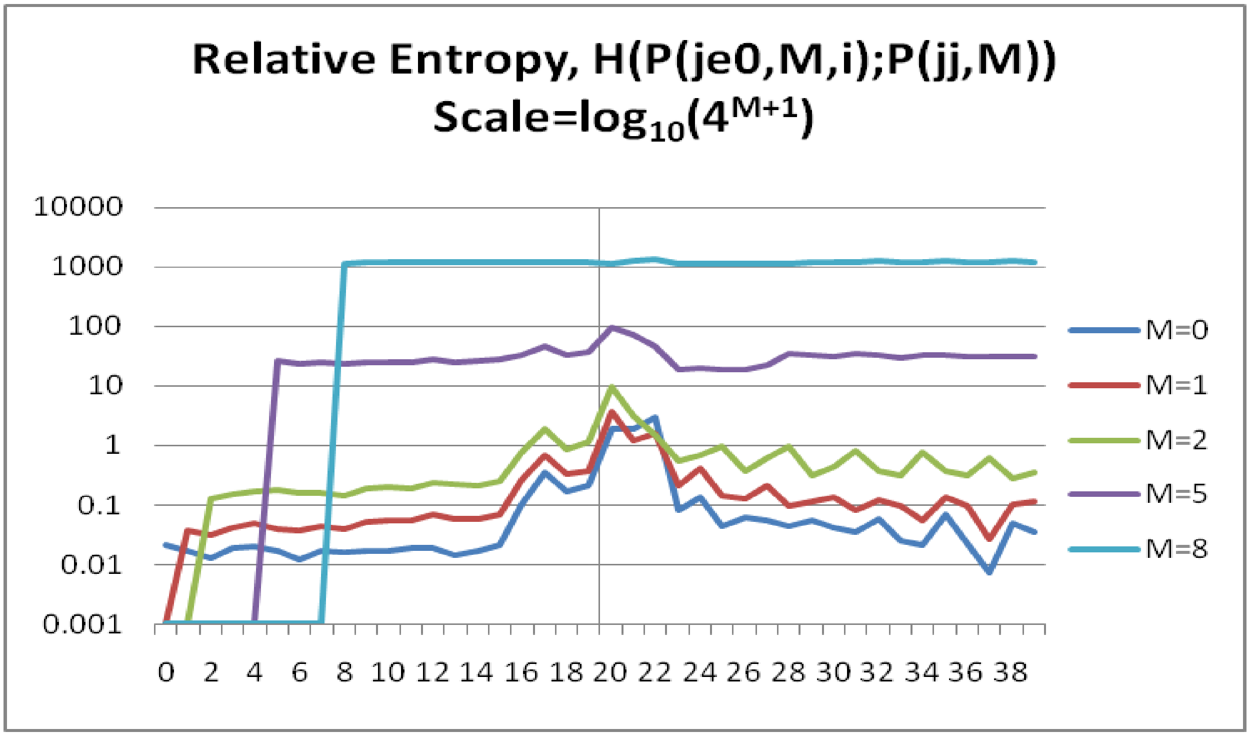


Figure 103 Relative entropy, H(P(je_0, M, i), P(jj, M)), for reduced C.E., I-V with log scaling.

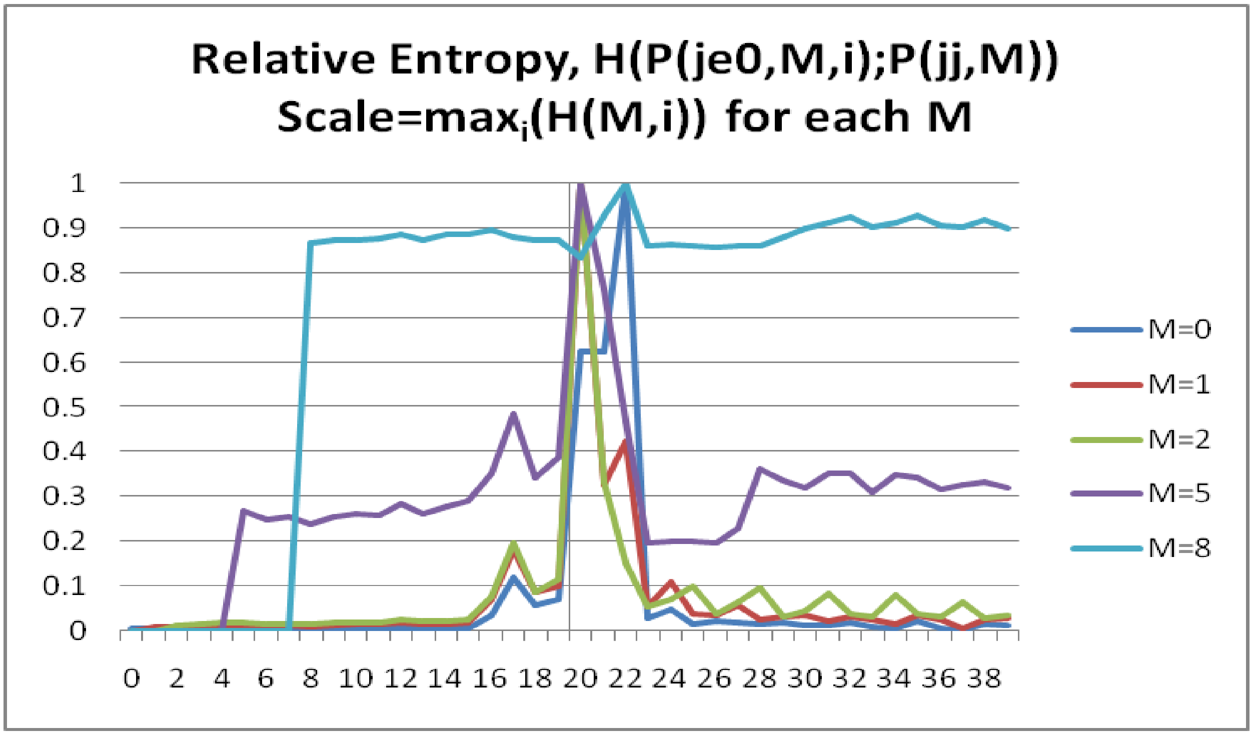


Figure 104 Relative entropy, H(P(je_0, M, i), P(jj, M)), for reduced C.E., I-V with max scaling.

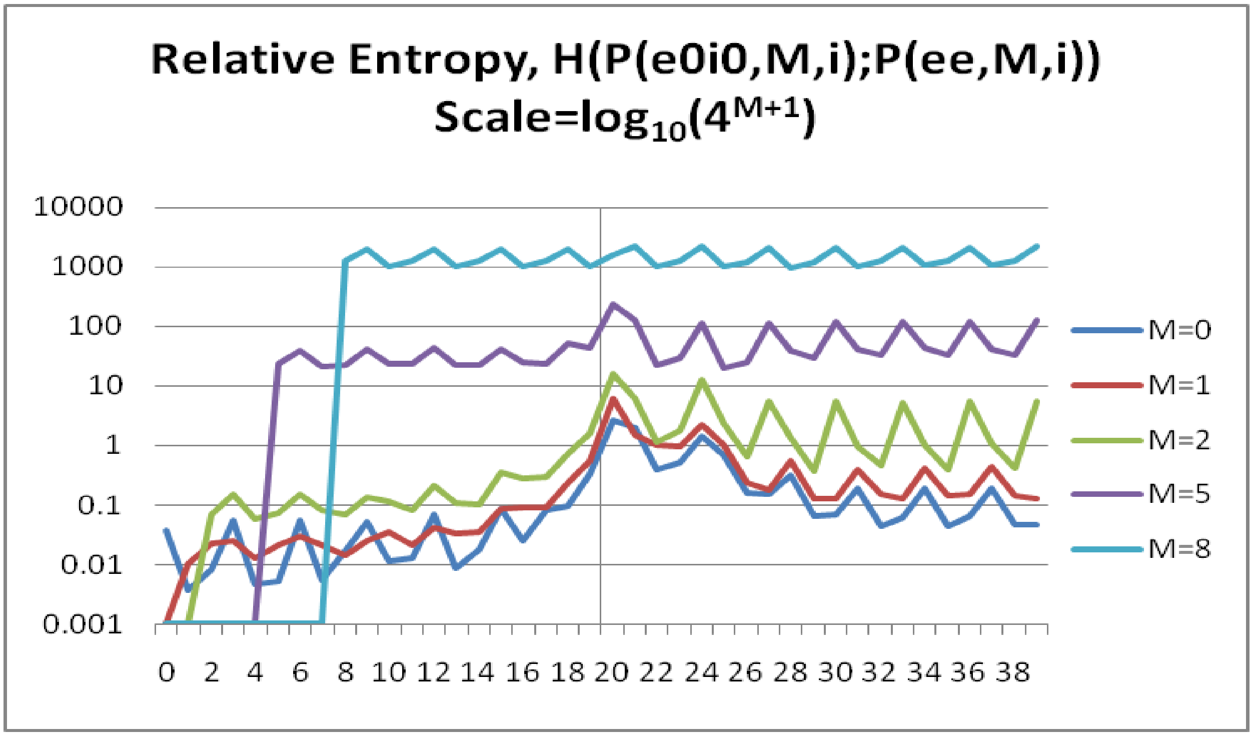


Figure 105 Relative entropy, H(P($e_0 i_0$, M, i), P(ee, M,i)), for reduced C.E., I-V with log scaling.

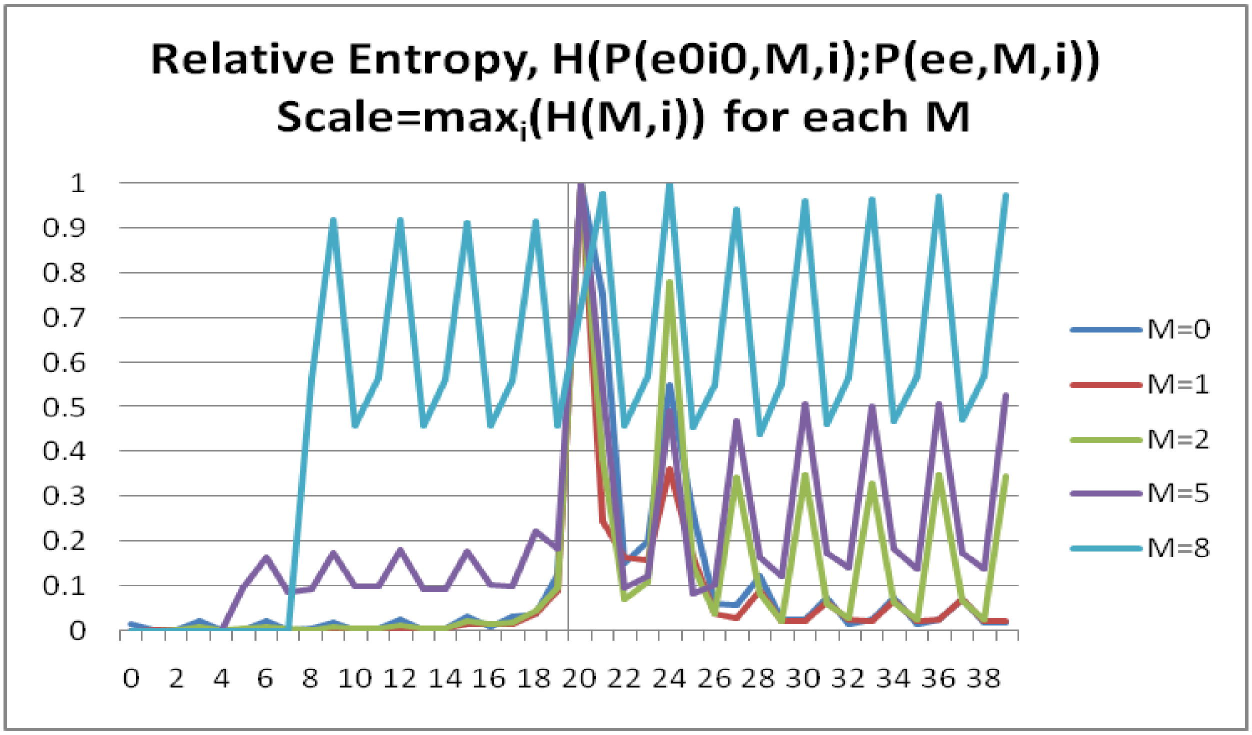


Figure 106 Relative entropy, H(P($e_0 i_0$, M, i), P(ee, M,i)), for reduced C.E., I-V with max scaling.

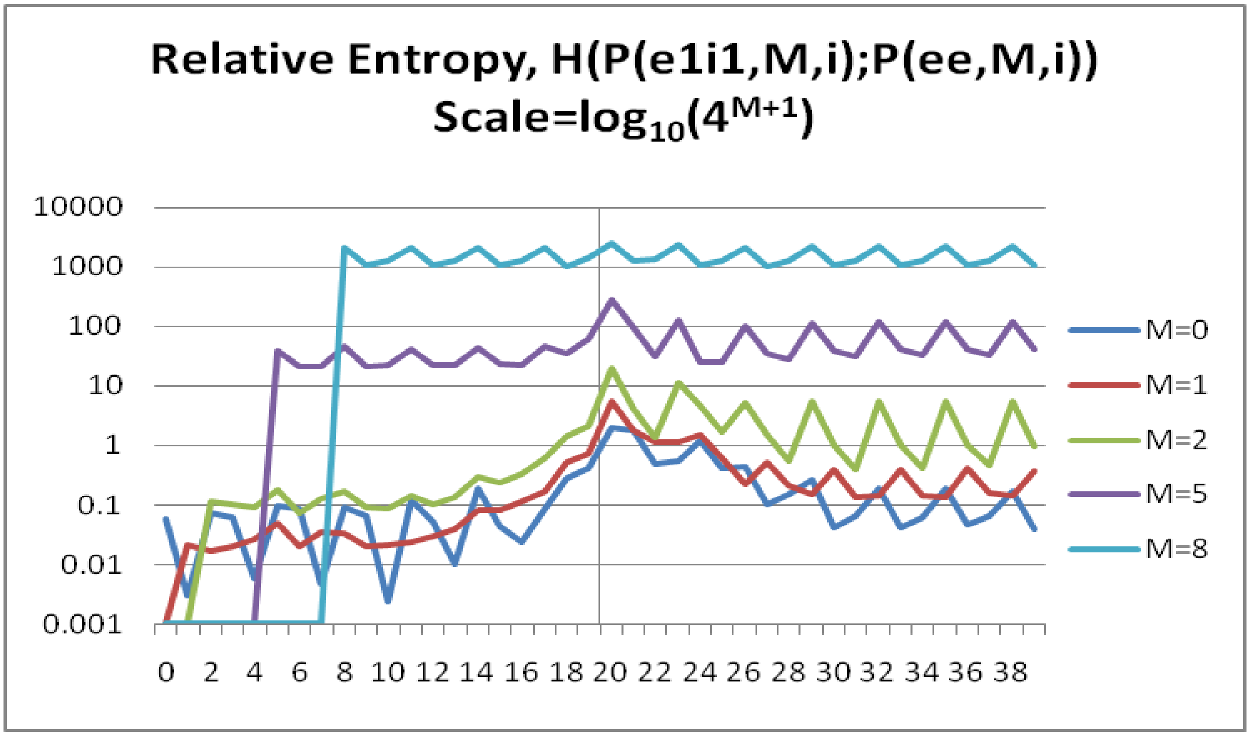


Figure 107 Relative entropy, H(P(e$_1$i$_1$, M, i), P(ee, M,i)), for reduced C.E., I-V with log scaling.

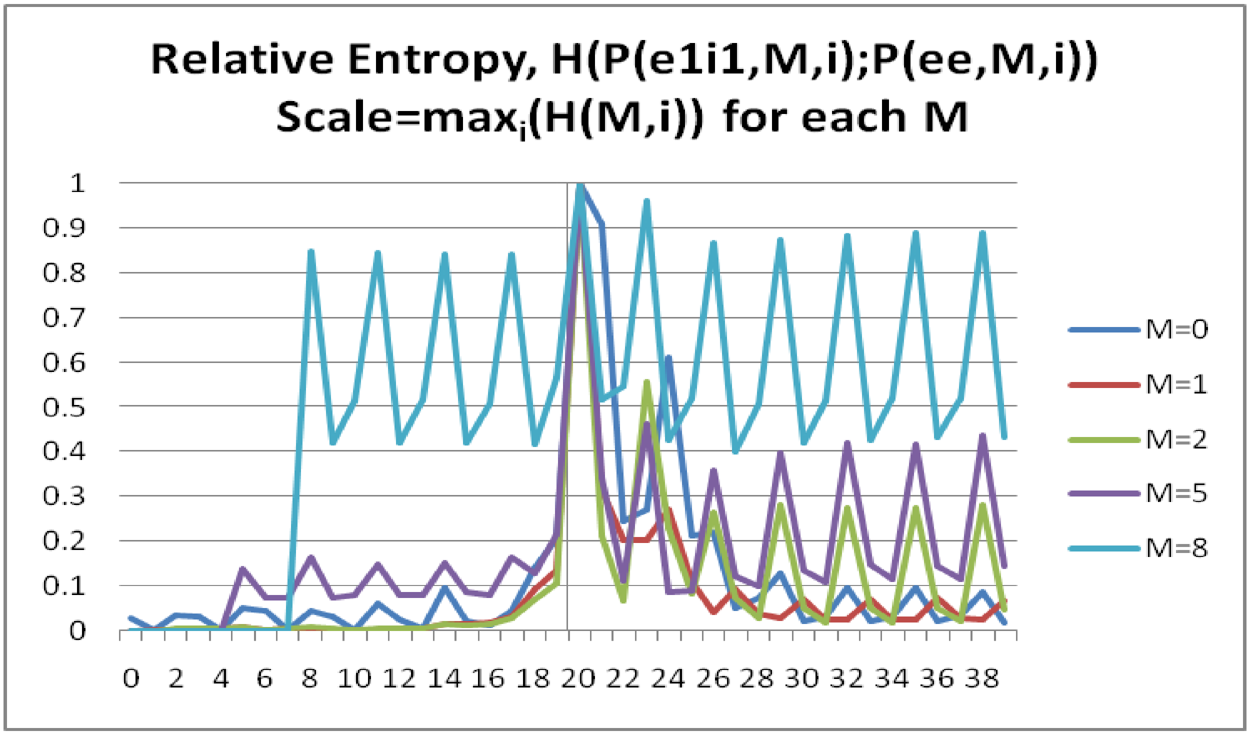


Figure 108 Relative entropy, H(P(e$_1$i$_1$, M, i), P(ee, M,i)), for reduced C.E., I-V with max scaling.

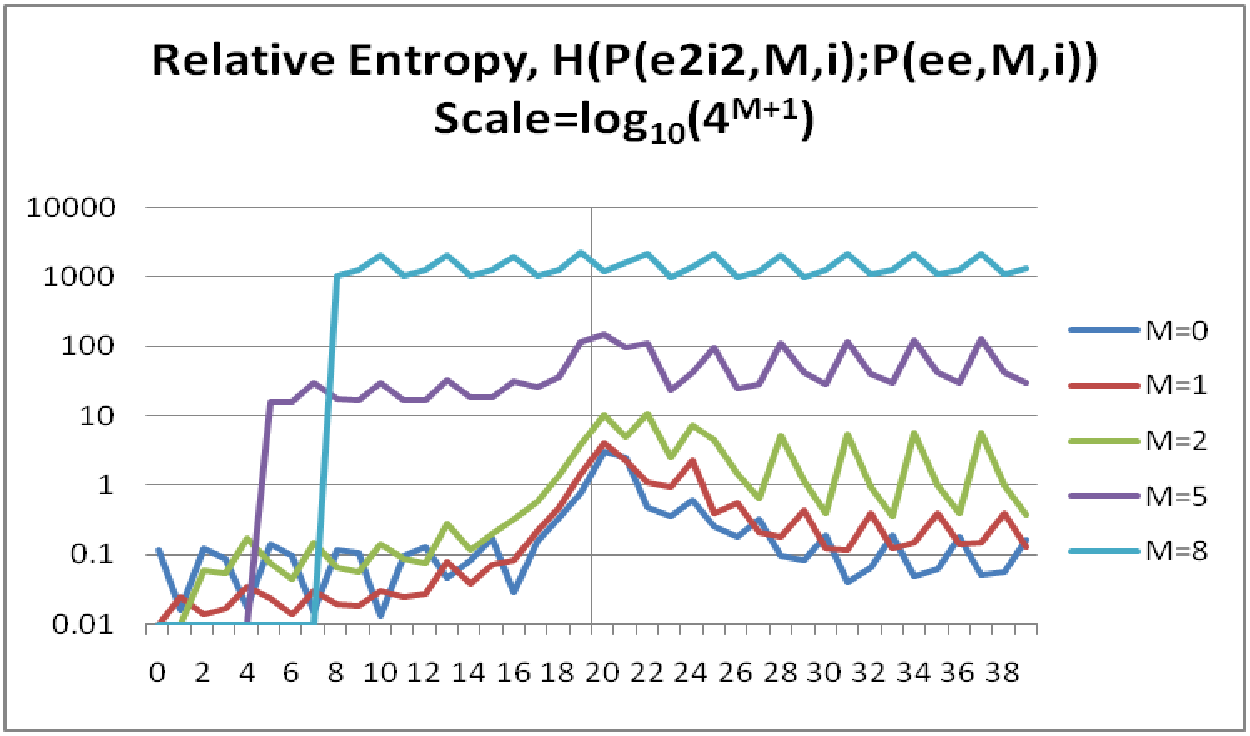


Figure 109 Relative entropy, H(P(e₂i₂, M, i), P(ee, M,i)), for reduced C.E., I-V with log scaling.

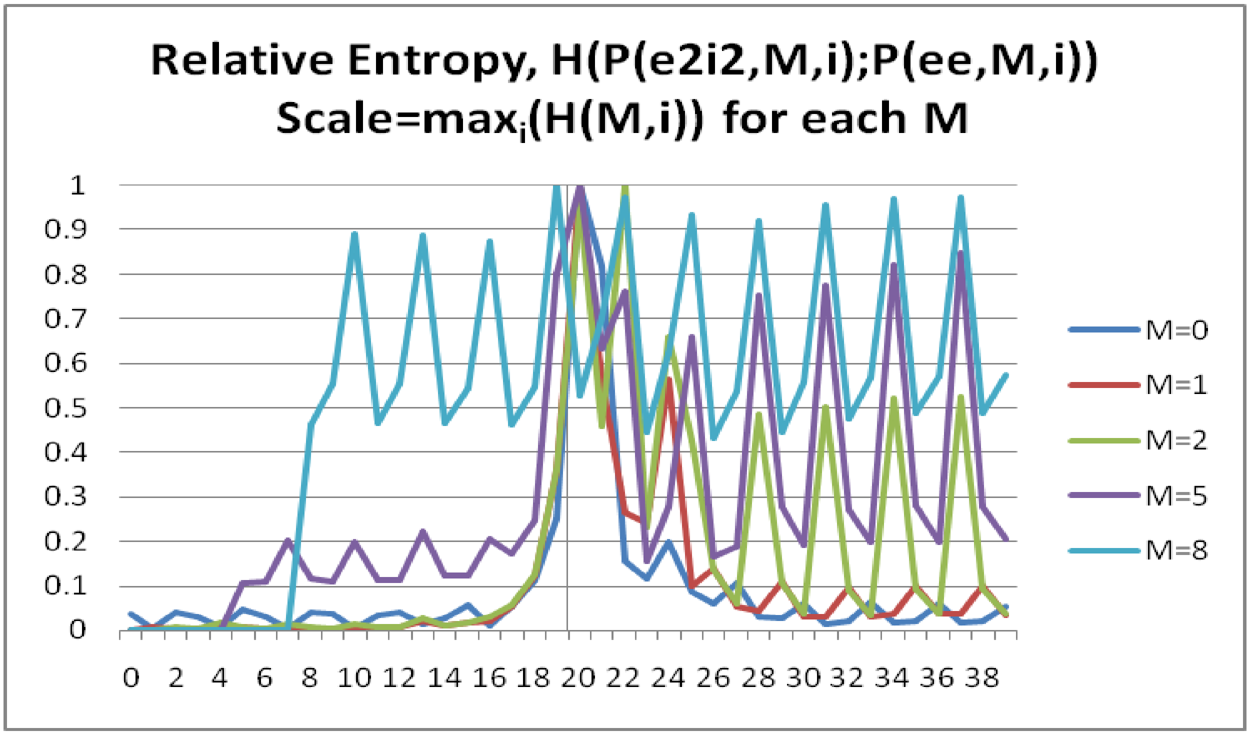


Figure 110 Relative entropy, H(P(e₂i₂, M, i), P(ee, M,i)), for reduced C.E., I-V with max scaling.

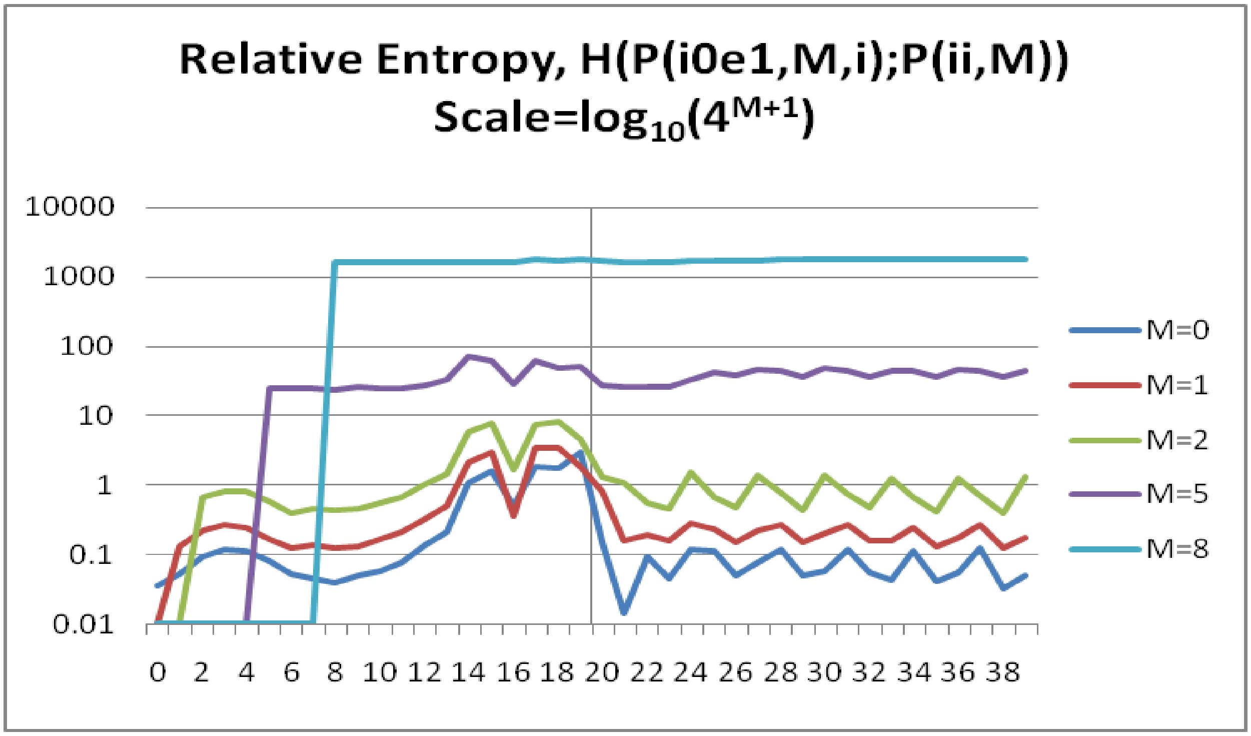


Figure 111 Relative entropy, H(P(i_0e_1, M, i), P(ii, M)), for reduced C.E., I-V with log scaling.

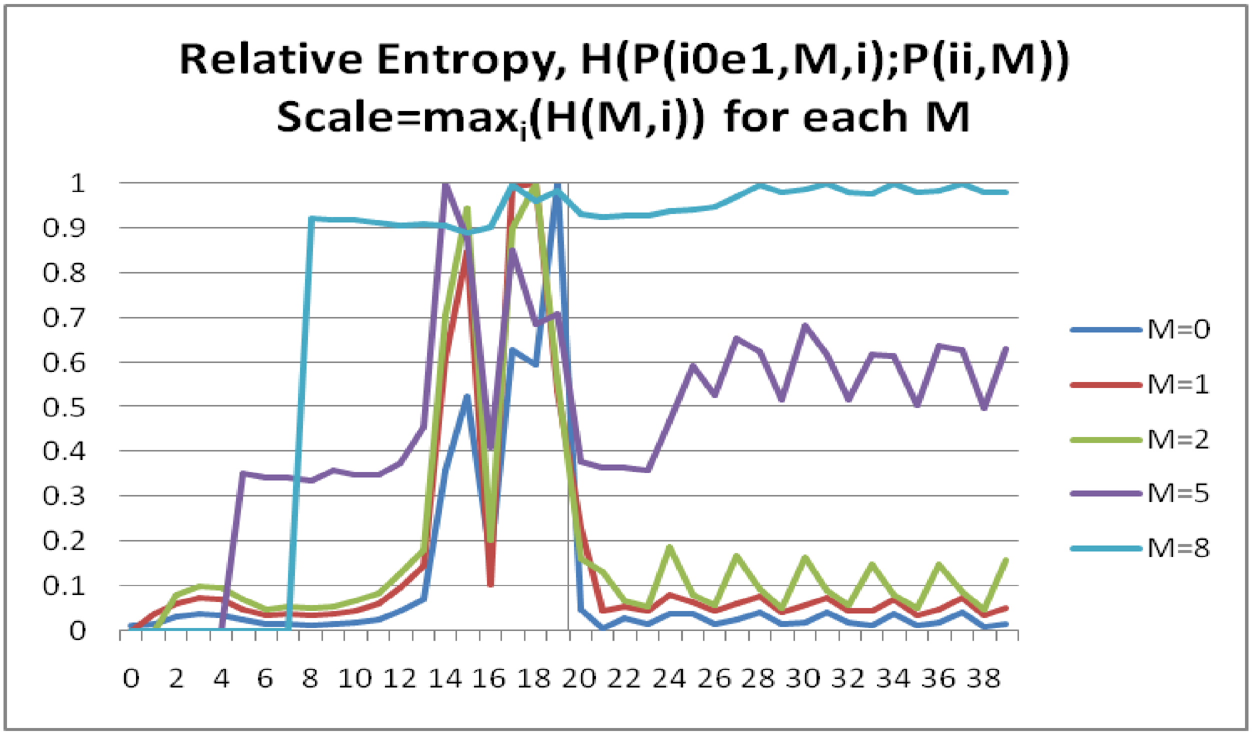


Figure 112 Relative entropy, H(P(i_0e_1, M, i), P(ii, M)), for reduced C.E., I-V with max scaling.

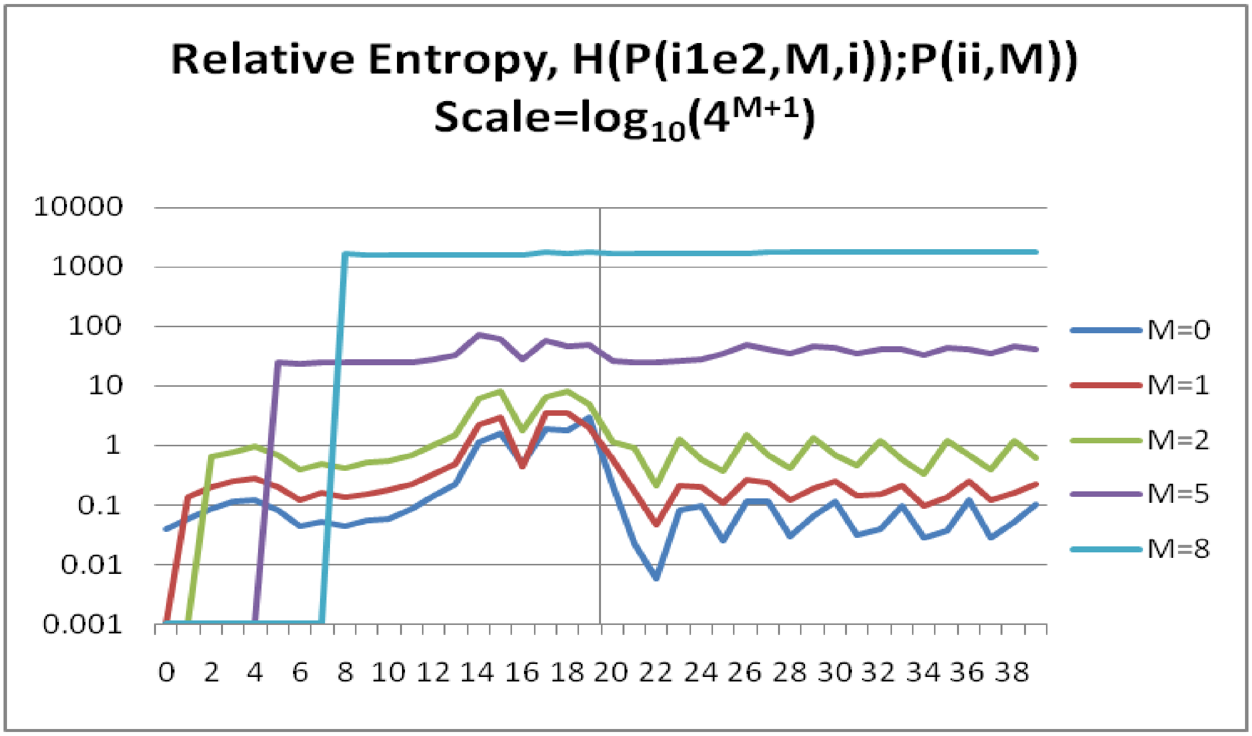


Figure 113 Relative entropy, H(P(i_1e_2, M, i), P(ii, M)), for reduced C.E., I-V with log scaling.

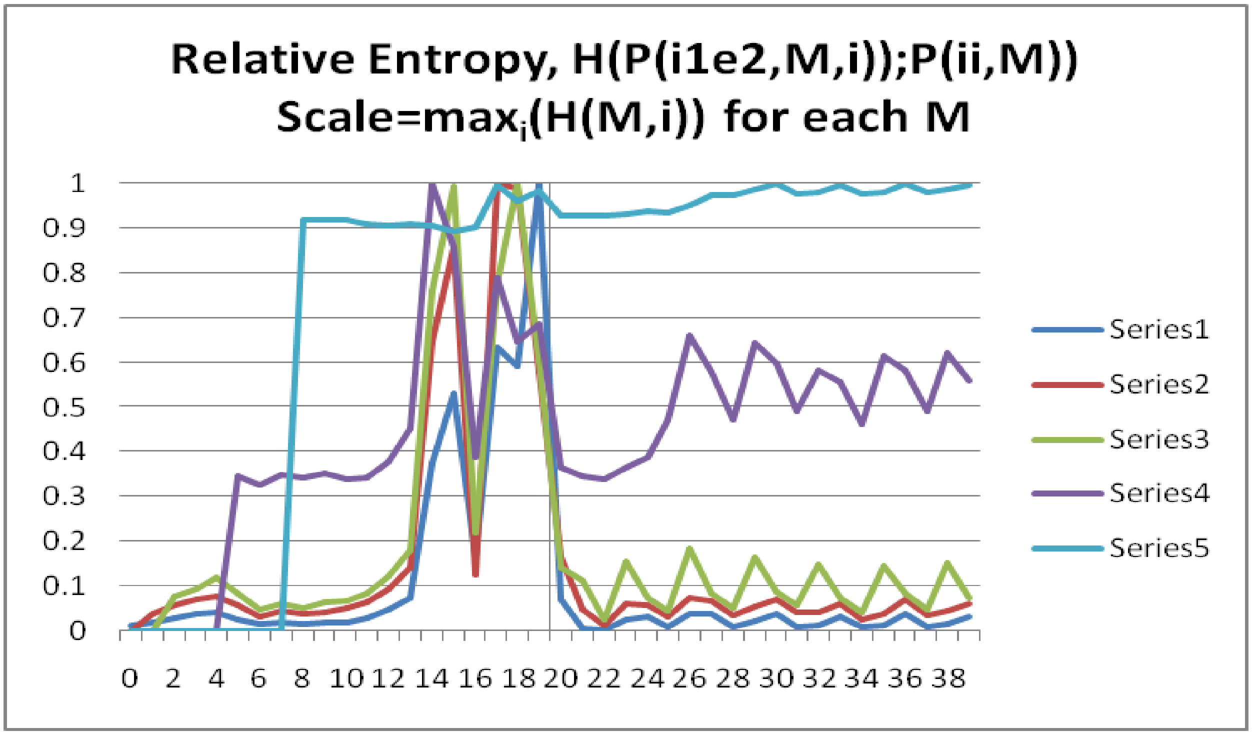


Figure 114 Relative entropy, H(P(i_1e_2, M, i), P(ii, M)), for reduced C.E., I-V with max scaling.

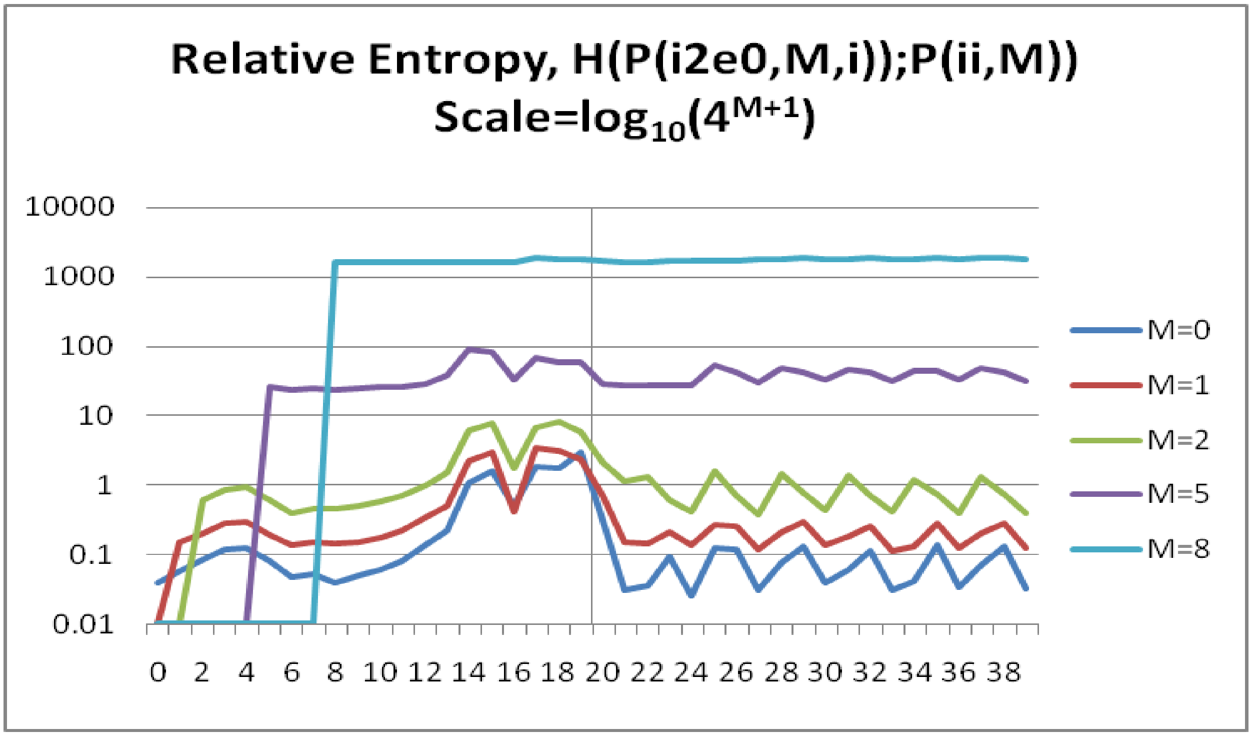


Figure 115 Relative entropy, H(P(i_2e_0, M, i), P(ii, M)), for reduced C.E., I-V with log scaling.

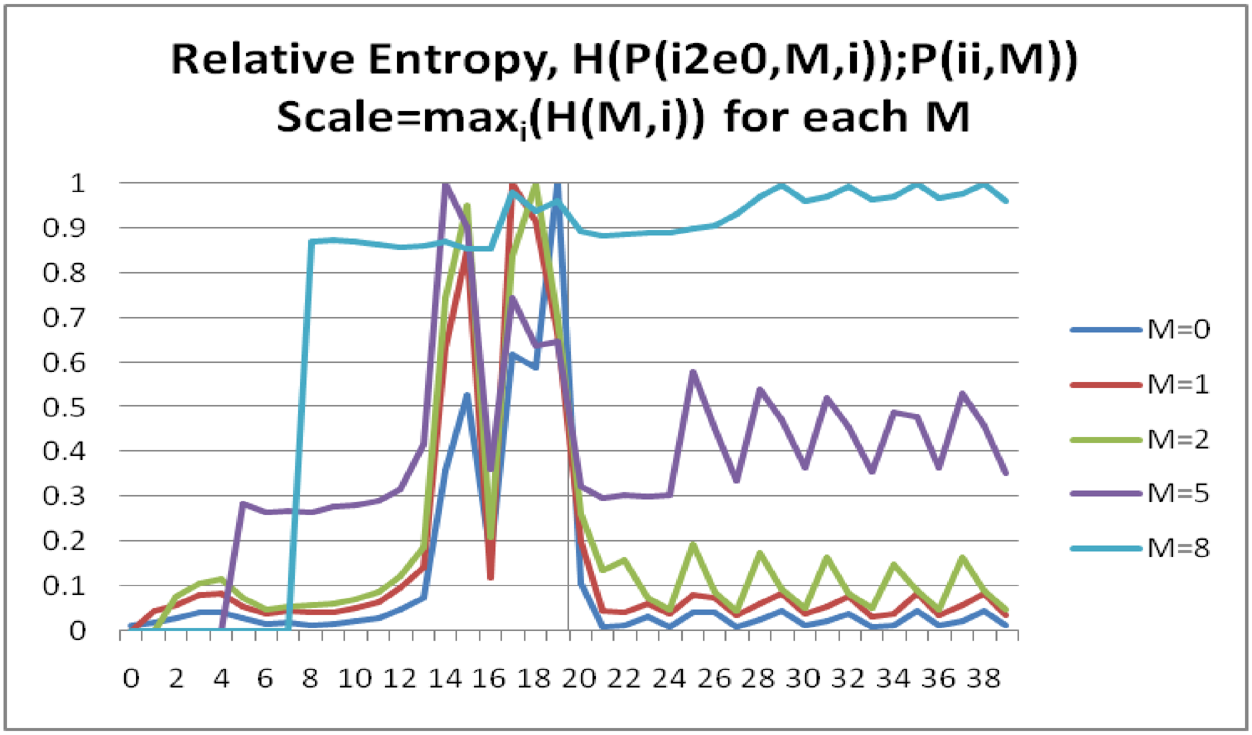


Figure 116 Relative entropy, H(P(i_2e_0, M, i), P(ii, M)), for reduced C.E., I-V with max scaling.

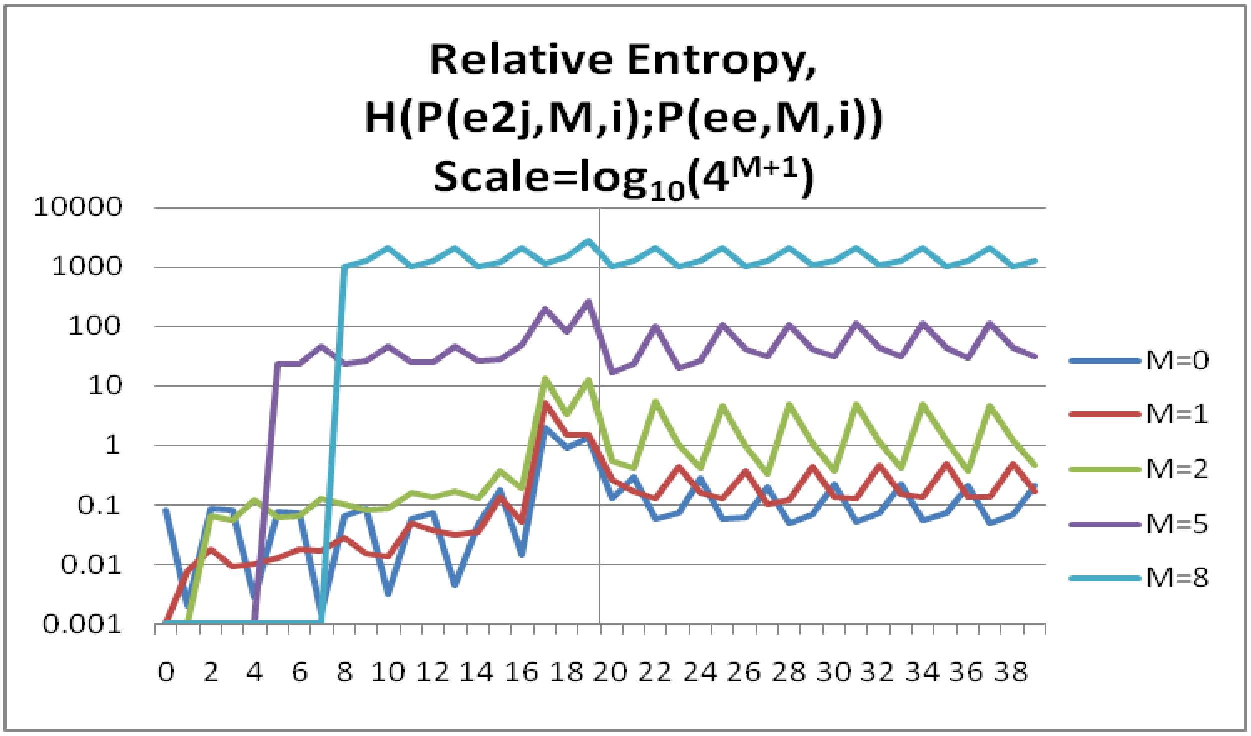


Figure 117 Relative entropy, H(P(e2j, M, i), P(ee, M,i)), for reduced C.E., I-V with log scaling.

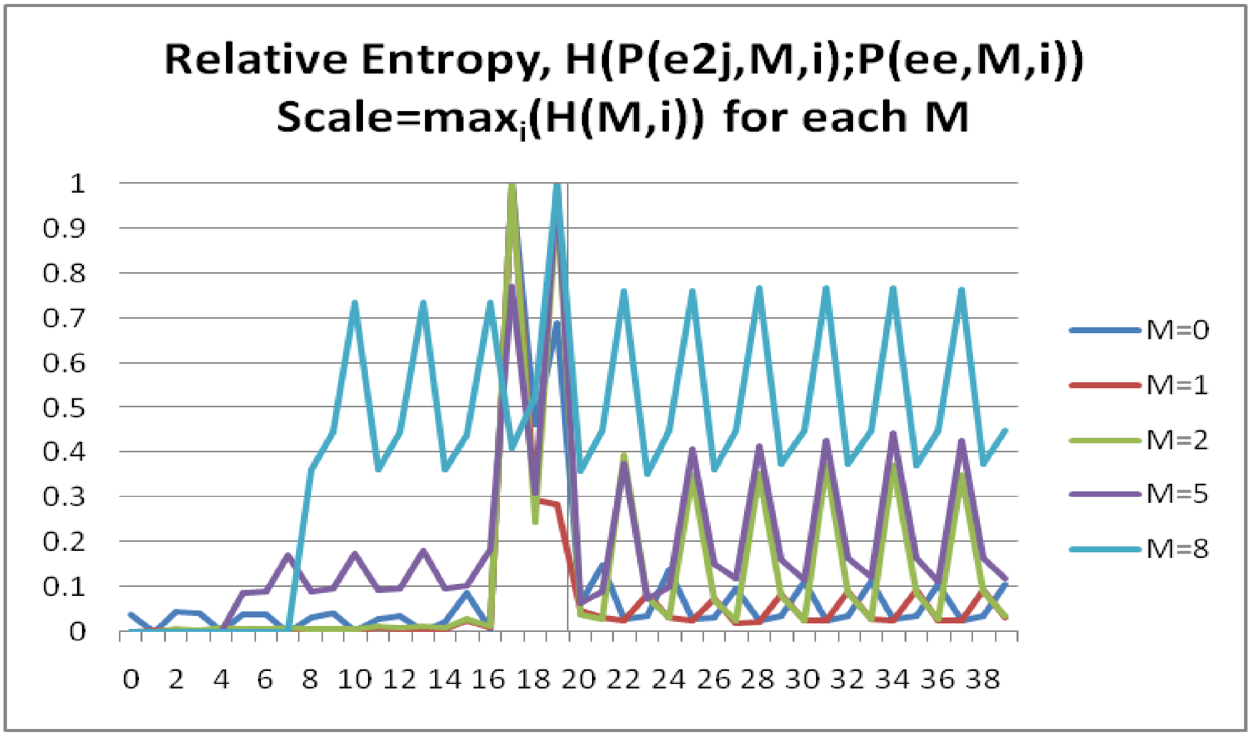


Figure 118 Relative entropy, H(P(e2j, M, i), P(ee, M,i)), for reduced C.E., I-V with max scaling.

8.5 SVM Classification Used in Pattern Recognition Informed (PRI) Sampling Selection

Support Vector Machines (SVMs) are variational-calculus based methods that are constrained to have structural risk minimization (SRM) such that they provide noise tolerant solutions for pattern recognition [**21**] [**56**]. Simply put, an SVM determines a hyper plane that optimally separates one class from another (see Figure 119). Once learned, the hyper plane allows data to be classified according to the region (separated by the hyper plane) in which it resides. Currently there are two approaches to implementing *multiclass* SVMs. One arranges several *binary* classifiers as a decision tree such that they perform a multi-class decision-making function (SVM-external classification). The second approach involves solving a single optimization problem corresponding to the entire data set (with multiple hyper planes), with multi-class discriminator optimization performed internally. The SVM-internal approach, when it is stable and properly generalizable (an active area of research), is preferred, since a tuning over Decision tree topologies and weightings is avoided [**57**]. The on-line discriminatory speed of a trained SVM is simply that of evaluating an inner product, so it's operational constraint on the PRI feedback process is negligible compared to that of the HMM feature extraction stage. For this reason, there is little discussion of SVMs in this paper, even though SVMs comprise much of the complexity of the HMM/SVM PRI feedback system.

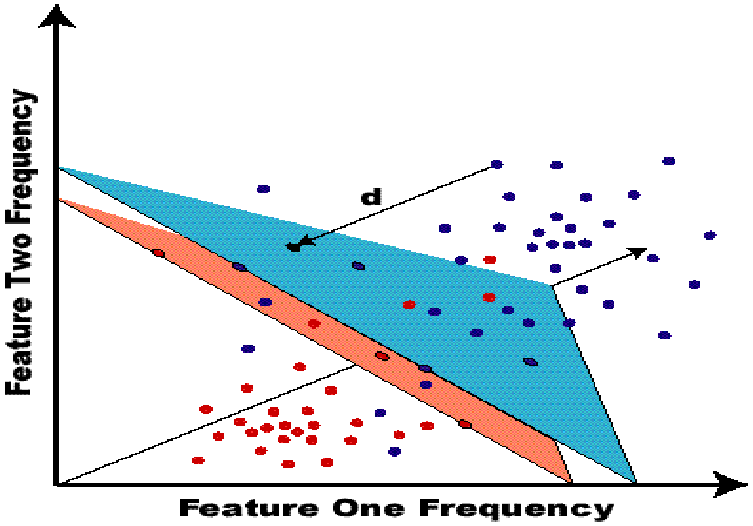


Figure 119 SVM: Hyper plane separability with a Margin (thickness). Support vectors consist of both the blue and red points occurring on the blue and red margin surfaces, respectively. Unlike HMM-based classification, the SVM-classification provides built-in confidence levels as part of the classification output.

8.6 Code Design for HOHMM

8.6.1 High Level Design Description

The HOHMM application performs both training and/or testing tasks according to the specifications in the input configuration file. For each configured task, a client task context is created in order to monitor the progress – according to the configured wait time, if any – of the corresponding server task process.

Protection is provided to some degree against multiple submittal attacks in the following manner. For each configured task, the client task context performs a status check on the assigned server before starting the corresponding task process on the server. During the status check, the client task context inquires as to whether a process with the given runtime parameters for the given server (local or remote) task is already in progress on the given server platform. If so, then the client task context avoids starting the task process on the server and either continues to monitor the progress of the existing process on the server or exits – according to the configured specifications.

The sequence in Figure 120 - Figure 126 depicts a single, fully connected flowchart representing the operations and collaborations of the key modules according to the following list of major code categories.

1. Training Code
    a. Training Code – Client Side
    b. Training Code – Server Side
2. Testing Code
    a. Testing Code – Client Side
    b. Testing Code – Server Side

See **8.6.2 Interfaces of Key Modules**, page149, for a list of the key modules and their interface features.

Main Sequence on Client – Includes 1)Training and 2)Testing

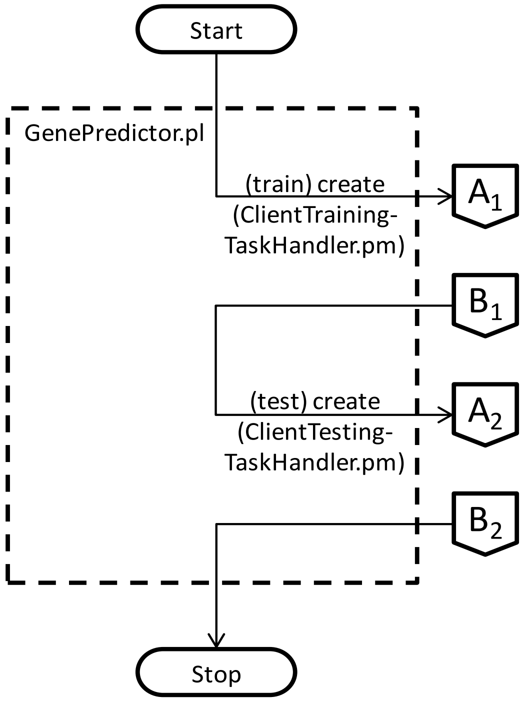


Figure 120 Flowchart for Main Sequence on Client – Includes 1)Training and 2)Testing

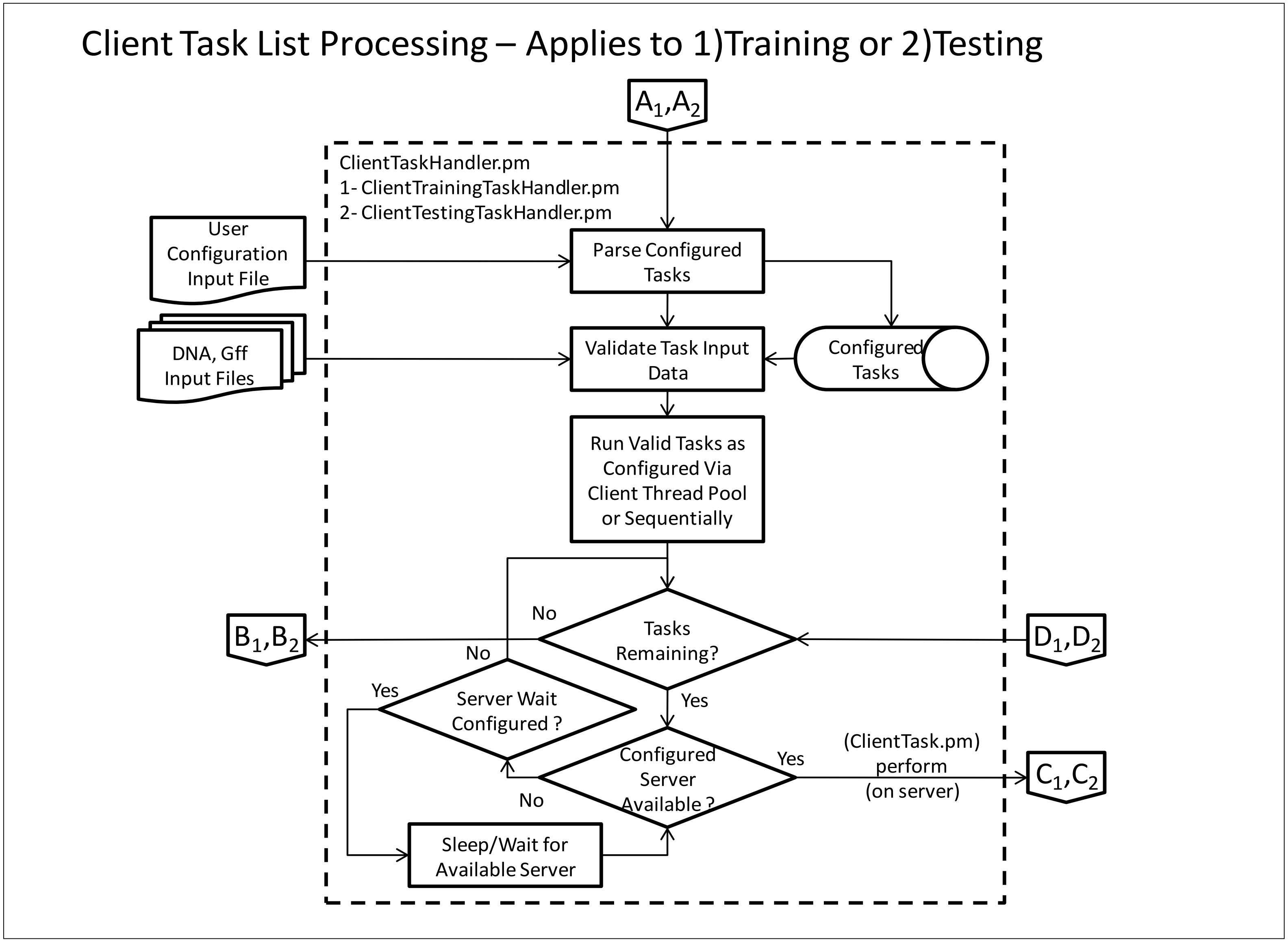

Client Task List Processing – Applies to 1)Training or 2)Testing
A_1,A_2
ClientTaskHandler.pm
1- ClientTrainingTaskHandler.pm
2- ClientTestingTaskHandler.pm
User Configuration Input File
DNA, Gff Input Files
Parse Configured Tasks
Validate Task Input Data
Configured Tasks
Run Valid Tasks as Configured Via Client Thread Pool or Sequentially
Tasks Remaining?
No
B_1,B_2
D_1,D_2
No
Server Wait Configured ?
Yes
Yes
Configured Server Available ?
Yes
No
(ClientTask.pm) perform (on server)
C_1,C_2
Sleep/Wait for Available Server

Figure 121 Flowchart for Client Task List Processing - Applies to 1)Training or 2)Testing

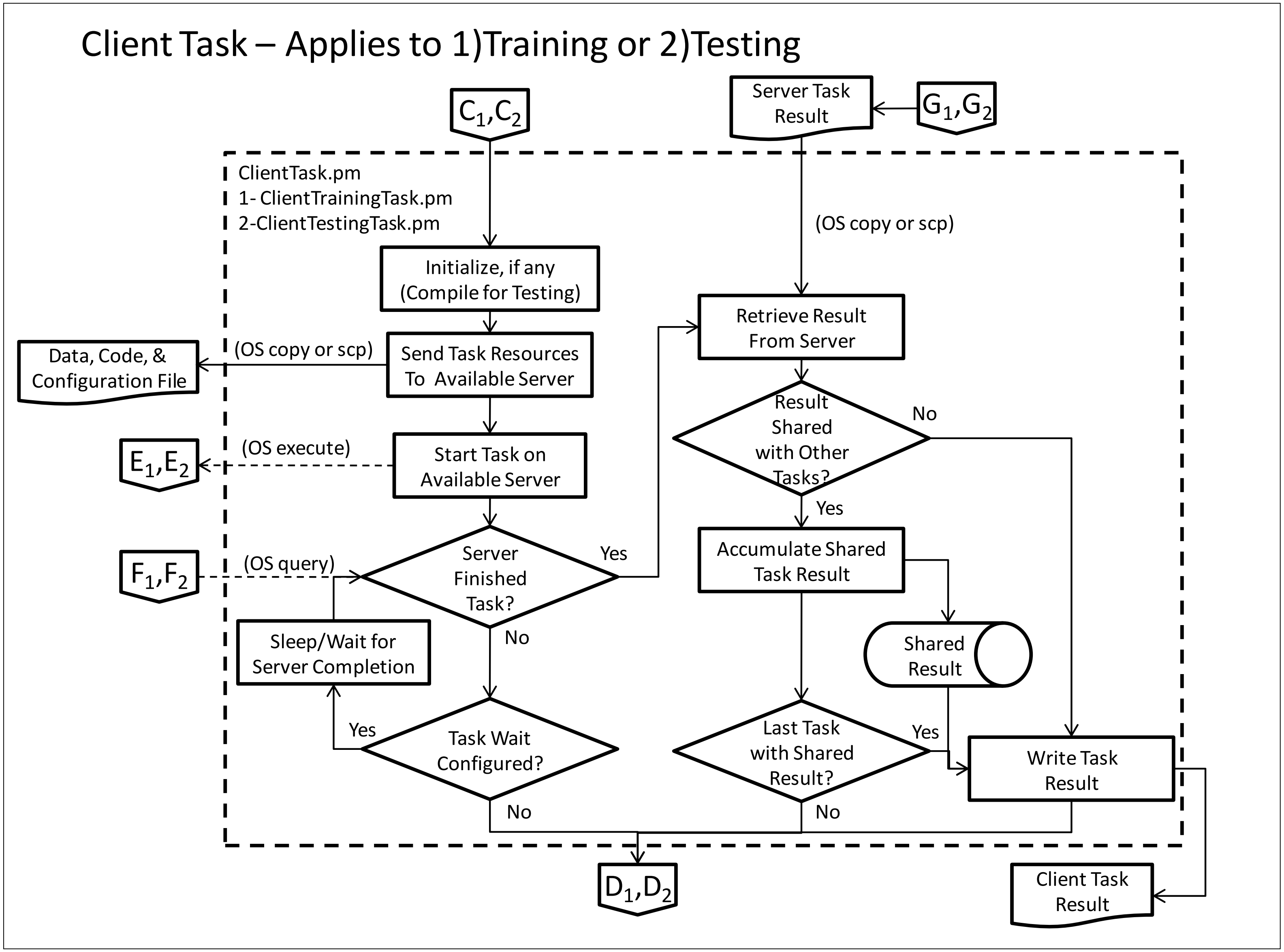

Client Task – Applies to 1)Training or 2)Testing
C1,C2
Server Task Result
G1,G2
ClientTask.pm
1- ClientTrainingTask.pm
2-ClientTestingTask.pm
(OS copy or scp)
Initialize, if any (Compile for Testing)
Retrieve Result From Server
Data, Code, & Configuration File
(OS copy or scp)
Send Task Resources To Available Server
Result Shared with Other Tasks?
No
E1,E2
(OS execute)
Start Task on Available Server
Yes
Accumulate Shared Task Result
F1,F2
(OS query)
Server Finished Task?
Yes
No
Sleep/Wait for Server Completion
Shared Result
Yes
Task Wait Configured?
No
Last Task with Shared Result?
Yes
No
Write Task Result
D1,D2
Client Task Result

Figure 122 Flowchart for Client Task - Applies to 1)Training or 2)Testing

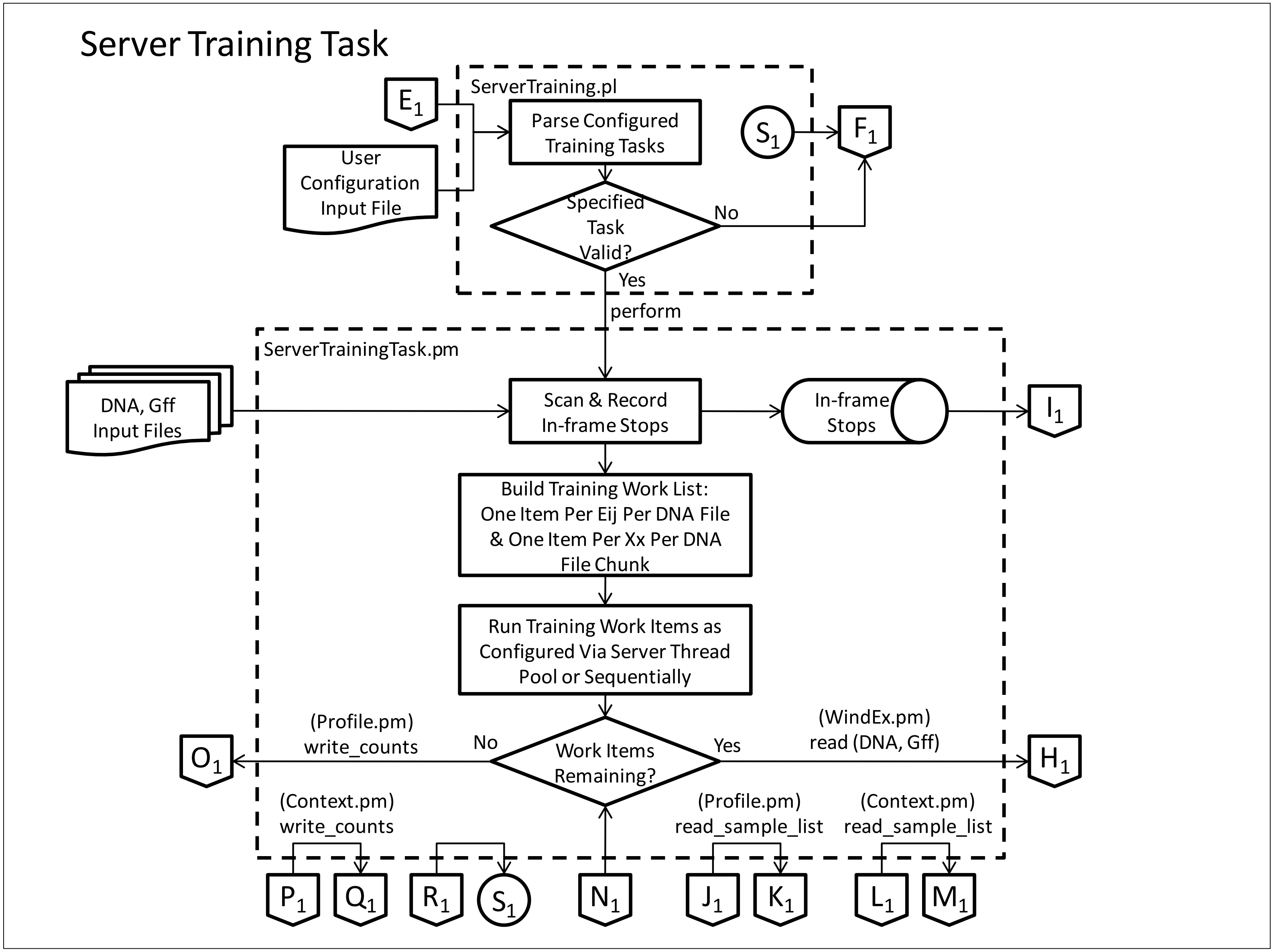

Server Training Task
ServerTraining.pl
E_1
User Configuration Input File
Parse Configured Training Tasks
S_1
F_1
Specified Task Valid?
No
Yes
perform
ServerTrainingTask.pm
DNA, Gff Input Files
Scan & Record In-frame Stops
In-frame Stops
I_1
Build Training Work List: One Item Per Eij Per DNA File & One Item Per Xx Per DNA File Chunk
Run Training Work Items as Configured Via Server Thread Pool or Sequentially
(Profile.pm) write_counts
No
Work Items Remaining?
Yes
(WindEx.pm) read (DNA, Gff)
O_1
H_1
(Context.pm) write_counts
(Profile.pm) read_sample_list
(Context.pm) read_sample_list
P_1
Q_1
R_1
S_1
N_1
J_1
K_1
L_1
M_1

Figure 123 Flowchart for Server Training Task

Server Training Task, cont.

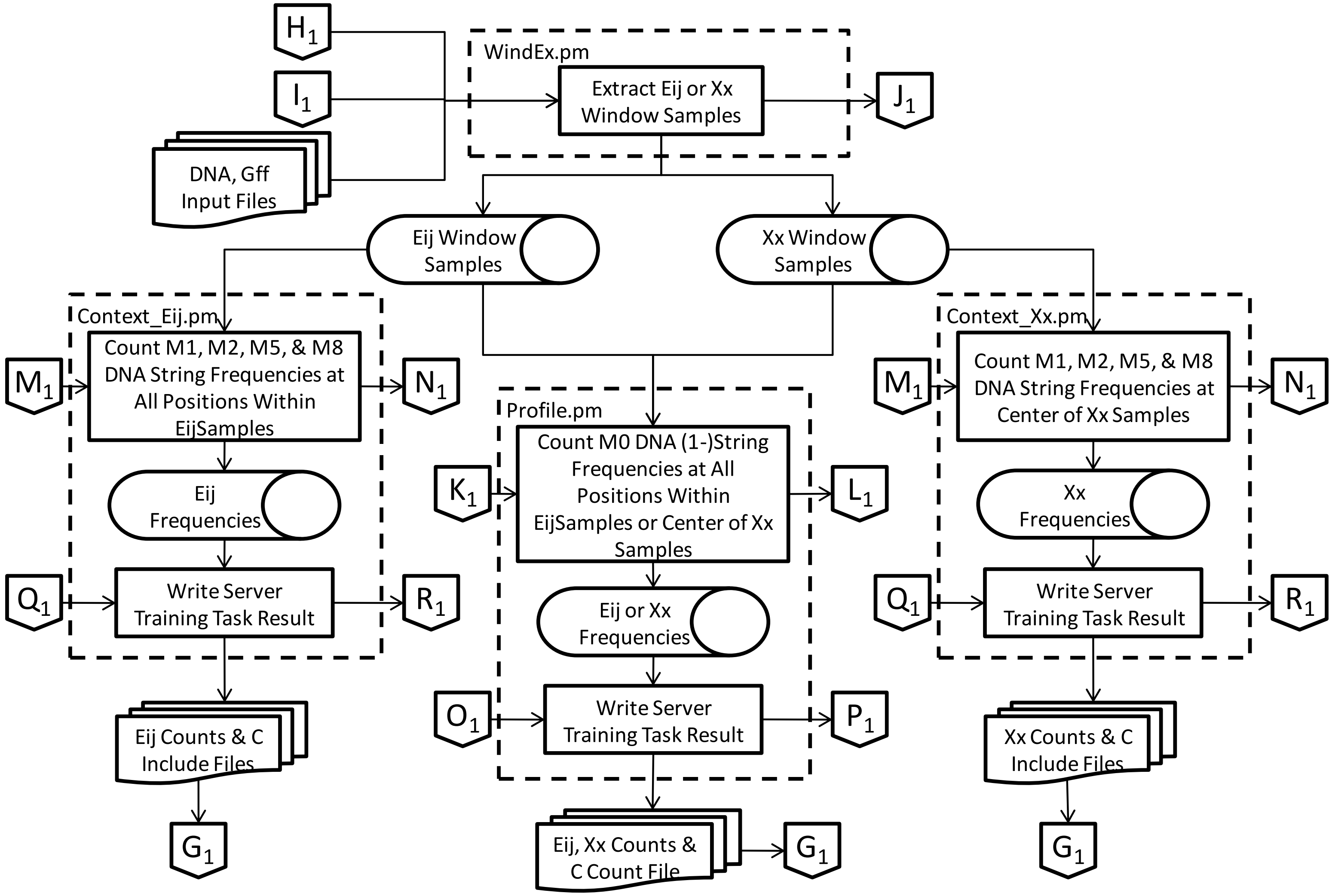

H1
I1
DNA, Gff Input Files
WindEx.pm
Extract Eij or Xx Window Samples
J1
Eij Window Samples
Xx Window Samples
Context_Eij.pm
M1
Count M1, M2, M5, & M8 DNA String Frequencies at All Positions Within EijSamples
N1
Eij Frequencies
Q1
Write Server Training Task Result
R1
Eij Counts & C Include Files
G1
Profile.pm
K1
Count M0 DNA (1-)String Frequencies at All Positions Within EijSamples or Center of Xx Samples
L1
Eij or Xx Frequencies
O1
Write Server Training Task Result
P1
Eij, Xx Counts & C Count File
G1
Context_Xx.pm
M1
Count M1, M2, M5, & M8 DNA String Frequencies at Center of Xx Samples
N1
Xx Frequencies
Q1
Write Server Training Task Result
R1
Xx Counts & C Include Files
G1

Figure 124 Flowchart for Server Training Task, cont.

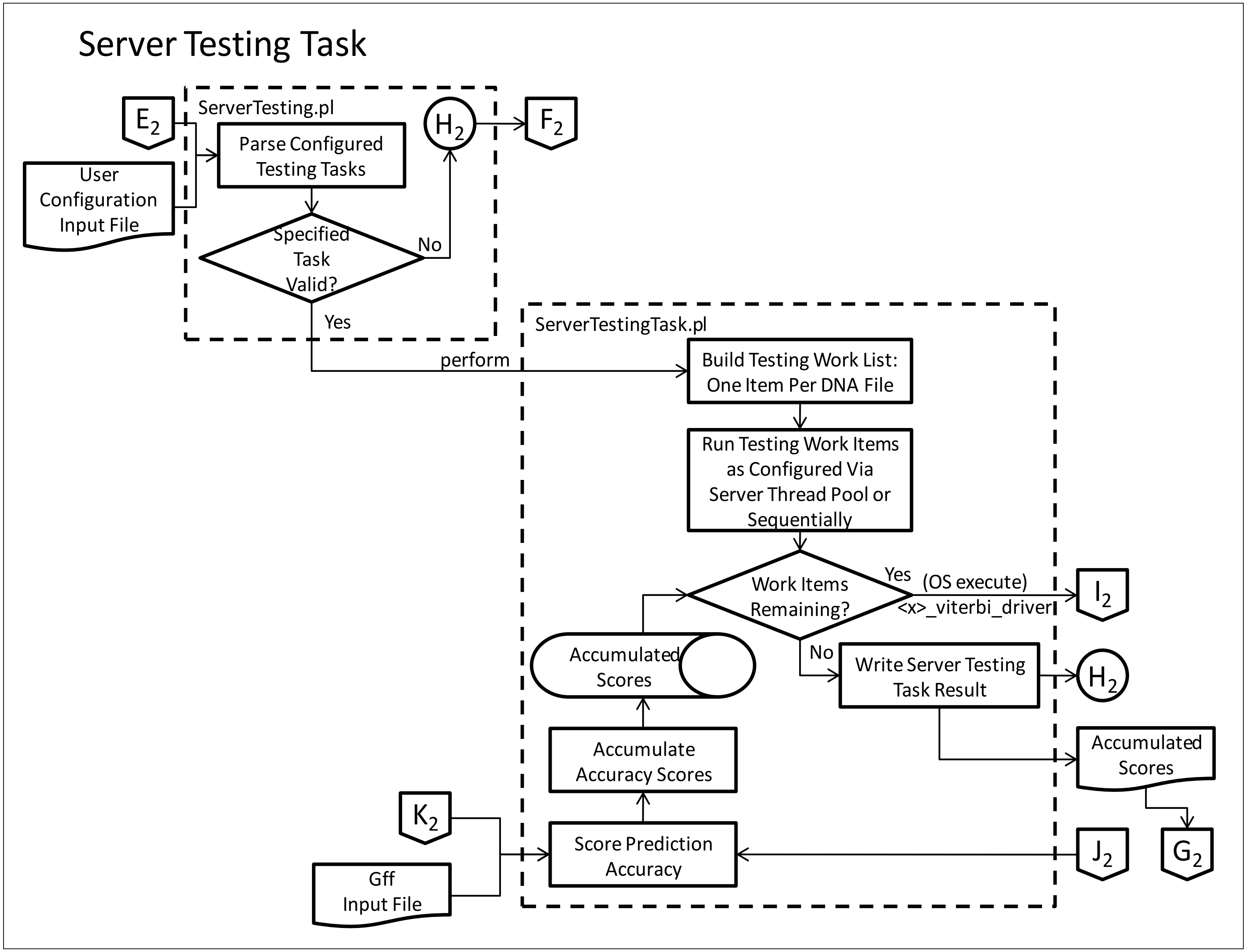
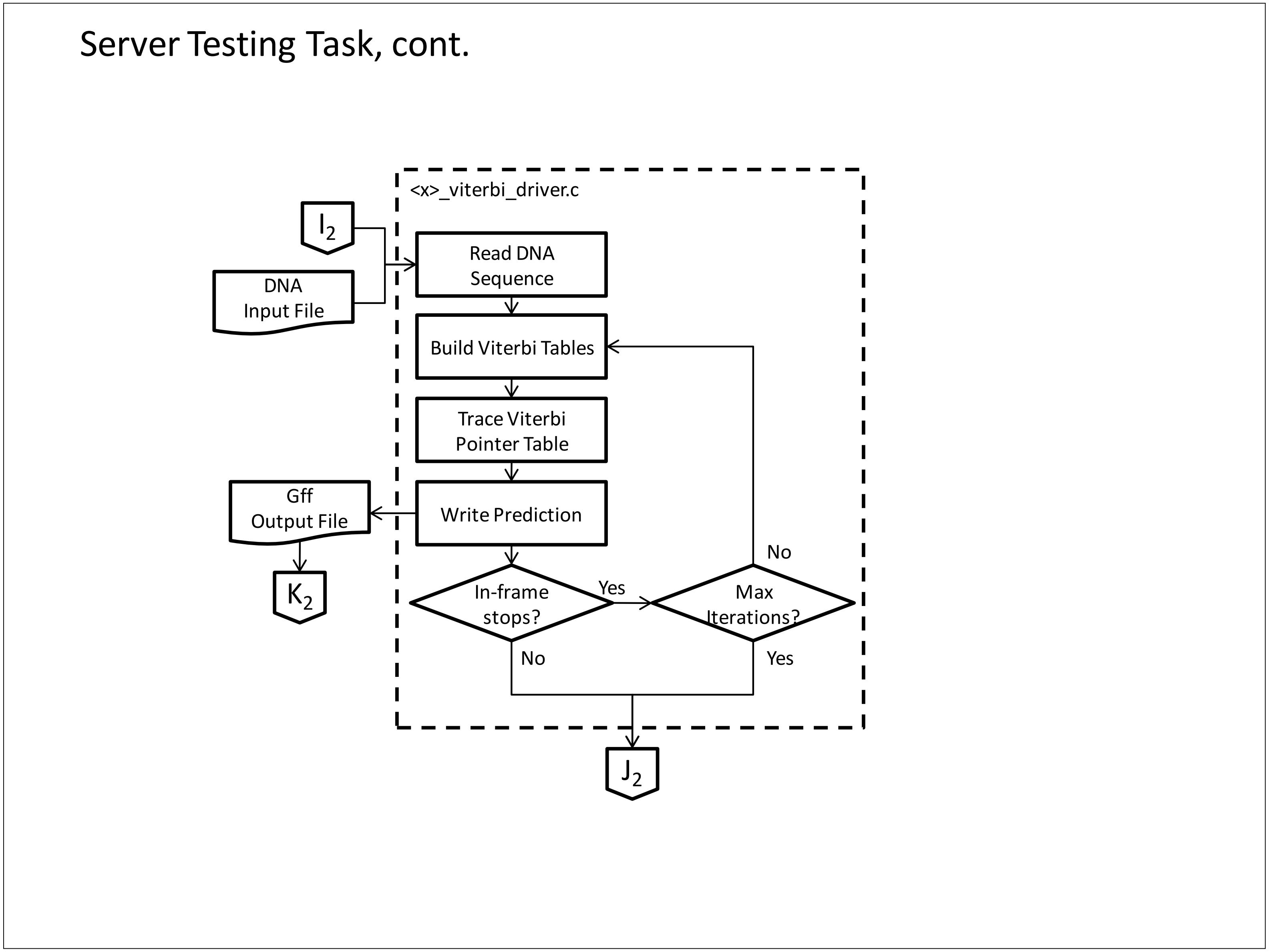

Server Testing Task
ServerTesting.pl
E_2
User Configuration Input File
Parse Configured Testing Tasks
Specified Task Valid?
No
Yes
H_2
F_2
perform
ServerTestingTask.pl
Build Testing Work List: One Item Per DNA File
Run Testing Work Items as Configured Via Server Thread Pool or Sequentially
Work Items Remaining?
Yes
(OS execute)
<x>_viterbi_driver
I_2
No
Write Server Testing Task Result
H_2
Accumulated Scores
Accumulate Accuracy Scores
Accumulated Scores
K_2
Gff Input File
Score Prediction Accuracy
J_2
G_2

Figure 125 Flowchart for Server Testing Task



Server Testing Task, cont.

<x>_viterbi_driver.c

I2

DNA Input File

Read DNA Sequence

Build Viterbi Tables

Trace Viterbi Pointer Table

Write Prediction

Gff Output File

K2

In-frame stops?

Yes

No

Max Iterations?

No

Yes

J2

Figure 126 Flowchart for Server Testing Task, cont.

8.6.2 Interfaces of Key Modules

8.6.2.1 Generic Code for Training and Testing

8.6.2.1.1 GenePredictor.pl

Parameters:

1) Name of user configuration file, where the file contents must be in Perl syntax
2) (Optional in general but required for 3) below) String containing Perl syntax for list of strings, where each string is taken as a regular expression to be used to select by name specific *training* task(s) to be performed from those defined in the user configuration file
3) (Optional) String containing Perl syntax for list of strings – similar to 2) above for selecting by name specific *testing* tasks to be performed from those defined in the user configuration file

Outputs: elapsed time message

8.6.2.1.2 ClientTaskHandler.pm

(function) create

- Parameters:
  1) List of regular expressions (See **8.6.2 Interfaces of** Key Modules
  2) Generic Code for Training and Testing
  3) GenePredictor.pl)
- Returned value: An instance of ClientTaskHandler for processing only those configured tasks selected by the inputs
- Outputs: none

8.6.2.1.3 ClientTask.pm

(function) create

- Parameters
  1) $task_type (“ClientTestingTask”, “ClientTrainingTask”)
  2) $task_name
- Returned value: An instance of Task of the specified type with the specified task_name attribute
- Outputs: none

8.6.2.1.4 ServerTask.pm

(function) create

- Parameters
  1) $task_type (“training”, “testing”)
  2) $task_name
- Returned value: An instance of ServerTask of the corresponding type (ServerTrainingTask, ServerTestingTask) with the specified task_name attribute
- Outputs: none

8.6.2.2 Training Code

8.6.2.2.1 Training Code – Client Side

8.6.2.2.1.1 ClientTrainingTaskHandler.pm

(function) create (See **8.6.2.1.2 ClientTaskHandler.pm**)

8.6.2.2.2 Training Code – Server Side

8.6.2.2.2.1 ServerTraining.pl

- Parameters
  1) Name of user configuration file (See **8.6.2.1.1 GenePredictor.pl**)

2) Name of training task to be performed as defined in the user configuration file

- Outputs: elapsed time message

8.6.2.2.2.2 ServerTrainingTask.pm

(function) perform

- Parameters: none
- Outputs:
  1) Various status messages
  2) Elapsed time message

8.6.2.2.2.3 WindEx.pm

(function) create

- Parameters
  1) $sample_window_size – sample window size
  2) $bad_exon_scan – flag (1 if performing in-frame stops, 0 otherwise)
  3) $ee_sample_period – exon sampling period (default = 3)
  4) $ii_sample_period – intron sampling period (default = 1)
  5) $jj_sample_period – junk sampling period (default = 1)
- Returned value: An instance of Windex
- Outputs: none

(function) read

- Parameters
  1) $gff – name of gff annotation file
  2) $dna – name of DNA sequence file
  3) $directory – name of directory containing $gff and $dna files
  4) $dimer – one of the 13 strings denoting the 13 possible dimers 'je0', 'e0i0', 'e1i1', 'e2i2', 'i0e1', 'i1e2', 'i2e0', 'e2j', 'e0e1', 'e1e2', 'e2e0', 'ii', or 'jj'
  5) $max_chunk_size – (largest) size of substring of DNA sequence
  6) $chunk_index – index of chunk using 0-origin
  7) $bad_exon_hash_ref – reference to hash structure containing in-frame stops
  8) $boundary_sampling_policy – flag ('pad' to pad truncated samples, omit otherwise)
- Returned value: an array of substrings sampled from the DNA sequence for the specified dimer
- Outputs:
  1) (file)<dna-name>_bad_exon (containing in-frame stops)
  2) (file)<dna-name>_internal_stops (containing all internal stops, in-frame and otherwise)
  3) $dna – name of DNA sequence file

8.6.2.2.2.4 Profile.pm

(function) create

- Parameters
  1) $dimer – (See **8.6.2.2.2.3 WindEx.pm**)
  2) $sample_window_size – sample window size
  3) $filter – flag (1 if filtering out in-frame stops, 0 otherwise)
- Returned value: An instance of Profile
- Outputs: none

(function) read_sample_list

- Parameters
  1) @sample_list – an array of substrings sampled from the DNA sequence for the owner's dimer
- Returned value: none
- Outputs: none

(function) write_counts

- Parameters: none
- Returned value: none
- Outputs:
  1) (file) <dimer>_profile_count – containing frequency of occurrence of 1-strings of all previously encountered sample lists since time of creation

8.6.2.2.2.5 Context.pm

(function) create

- Parameters
  1) $dimer – (See **8.6.2.2.2.3 WindEx.pm**)
  2) $sample_window_size – sample window size
  3) $filter – flag (1 if filtering out in-frame stops, 0 otherwise)
  4) (optional) $counts – array containing previously determined frequencies of occurrence of DNA substrings
- Returned value: An instance of Context of the type Context_EIJ or Context_XX as appropriate for the specified $dimer
- Outputs: none

(function) read_sample_list

- Parameters
  1) @sample_list – an array of substrings sampled from the DNA sequence for the owner's dimer
- Returned value: none
- Outputs: none

8.6.2.2.2.6 Context_EIJ.pm

(function) create (See **8.6.2.2.2.5 Context.pm**)

(function) read_sample_list (See **Context.pm**)

(function) write_counts

- Parameters: none
- Returned value: none
- Outputs:
  1) (file) <dimer>_context_count – containing frequency of occurrence of 2-, 3-, 6-, and 9-strings at *all* window positions *as support allows* in all previously encountered sample lists since time of creation

8.6.2.2.2.7 Context_XX.pm

(function) create (See **8.6.2.2.2.5 Context.pm**)

(function) read_sample_list (See **Context.pm**)

(function) write_counts

- Parameters: none
- Returned value: none
- Outputs:

1) (file) <dimer>_context_count – containing frequency of occurrence of 2-, 3-, 6-, and 9-strings at *center* window positions *only* in all previously encountered sample lists since time of creation

8.6.2.3 Testing Code

8.6.2.3.1 Testing Code – Client Side

8.6.2.3.1.1 ClientTestingTaskHandler.pm

(function) create (See **8.6.2.1.2 ClientTaskHandler.pm**)

8.6.2.3.2 Testing Code – Server Side

8.6.2.3.2.1 ServerTesting.pl

- Parameters
  1) Name of user configuration file (See **8.6.2.1.1 GenePredictor.pl**)
  2) Name of testing task to be performed as defined in the user configuration file
- Outputs: elapsed time message

8.6.2.3.2.2 ServerTestingTask.pm

(function) perform

- Parameters: none
- Returned value: none
- Outputs:
  1) Various status messages
  2) Cumulative prediction scores for all DNA files tested
  3) Elapsed time message

8.6.2.3.2.3 <x>_m8520_viterbi_driver.c

(function) main

- Parameters
  1) dna_file – DNA sequence file
  2) chunk_start – starting position in DNA sequence (0-origin)
  3) max_chunk_size – size of DNA subsequence for testing
  4) max_viterbi_passes – maximum # of Viterbi passes for in-frame stop removal
  5) log_stop_penalty – log (base-10) of probability penalty for in-frame stop removal
  6) Markov – maximum Markov order in DNA sequence (0, 2, 5, or 8)
  7) footprint_size – size of footprint state in dimers
  8) left_base_extent – left extent, L, of emitted DNA subsequence at each Viterbi table position
  9) right_base_extent – right extent, R, of emitted DNA subsequence at each Viterbi table position
  10) sequence_gain – defaults to 1
  11) viterbi_output_prefix – prefix of gff output file
  12) dump_stop_filter_log – flag (1 to log in-frame stops, none otherwise)
  13) dump_traceback – flag (>0 to log Viterbi traceback including symbolic state sequence printed in rows of specified width, none otherwise)
  14) dump_viterbi_probabilities – flag (1 to log Viterbi probabilities, none otherwise)
  15) dump_viterbi_pointers – flag (1 to log Viterbi back pointers, none otherwise)
  16) dump_viterbi_start – starting column index for Viterbi probability and/or pointer log (-1 for all columns)

17) dump_viterbi_end – ending column index for Viterbi probability and/or pointer log (ignored if dump_viterbi_start = -1)

- Outputs:
  1) Various status messages
  2) Predicted gff annotation files with file name format <viterbi_output_prefix>_pass_<n>.gff, with one such file for each of the Viterbi iterations performed and n=1, …, max_viterbi_passes
  3) Elapsed time message

8.6.3 Future Infrastructure

Besides the extensions indicated above in **6.2 Future Work**, the following additional modifications to the existing system's infrastructure with moderate levels of additional design and coding effort.

1. Full job checkpoint/restart capability.
2. Full support in a single configuration input file for multi-fold cross validation – with non-redundant sharing of training results from folds for use in subsequent testing. Current support for cross-validation is limited due to lack of support for shared results of training among the various folds of cross-validation in a single session.
   a. Currently, all training tasks in a given configuration input are assumed to share a single set of count file output.
   b. Currently, only testing tasks in a given configuration input are allowed to selectively share summed score file output.
3. Generalization of the above to a user-configurable work flow for support of a many-to-many mapping of training tasks to testing tasks.

9 Vita

Carl Baribault was born in Franklin, LA, and received his B.S. in Physics from Loyola University, New Orleans. He then went on to obtain an M.S. in Physics from the University of North Carolina at Chapel Hill. After working as a geophysicist in petroleum exploration for several years, he returned to academics for an M.A. in Mathematics from the University of Texas at Austin. Since then he has worked as a programmer for many years developing applications in various fields including mainframe computer engineering, natural gas trading, real estate property management, digital mapping, and most recently as graduate assistant involved in research in bioinformatics, including nanopore signal control and acquisition and structural gene prediction.