Computational Identification of Cis-regulatory Modules in DNA Sequences

Zachary D Burke

A Thesis in the Field of Information Technology

for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

June 2006

Abstract

The form and function of all living organisms are determined and managed in great part by modulations in the transcription of their DNA. The identification of sites where regulatory proteins bind to DNA is thus crucial to understanding regulatory mechanisms. Working with sequences containing regulatory regions (cis-regulatory modules, or CRMs) at known positions, we re-implemented a published CRM-finding algorithm and developed a test suite to identify parameters that would provide optimal and consistent performance across different test sequences. This resulted in a modest but consistent improvement over the published results, although in some cases the optimized performance was dramatically better. We refined the results in two phases. First, we ran the algorithm multiple times with different, high-performing parameter sets and selected only the regions (putative CRMs, or pCRMS) shared among all runs. Second, we searched for motifs in the pCRMs of coregulated sequences and removed regions with low densities of these shared motifs, a step that ultimately proved inconclusive.

Acknowledgments

I would like to thank my advisor Dr. Robert H. Gross for his patience, his insight and his outstanding instruction. I am indebted to Dr. Arijit Chakravarty and to Jonathan Carlson for their many insightful comments and conversations while working on this project. Susan Schwarz was instrumental in generating diagrams with OpenDX. This work was supported by NSF grant number DBI-0445967 to RHG.

Table of Contents

List of Figures

Chapter 1 Molecular Biology's Central Dogma

Part of the central dogma of molecular biology is that an organism's genetic information is encoded in DNA, that this information is transcribed into RNA, and that RNA is translated into proteins (Campbell, 2002). The resulting proteins may then be involved in nearly any aspect of the life of cell function at the molecular level, whether they serve as catalysts for reactions (enzymes) or as the building blocks of cellular structures. In ordinary circumstances, the flow of information is one-directional: DNA is transcribed into RNA and RNA is translated into proteins. There are, of course, exceptions to this role. In retroviruses, the family of viruses that contains HIV, the primary genetic material is RNA. These viruses often contain the code for reverse transcriptase, an enzyme that synthesizes DNA from the virus's RNA template. This DNA can then be inserted into a host's genome (Campbell, 2002).

Gene Regulation

Just as important as understanding the mechanisms of transcription and translation is understanding the factors that modulate them. The form and function of all living organisms are determined and managed by modulations in the transcription of their DNA. Differences in the sets of genes that are present may separate birds from fish, but differences in which genes are expressed separate a blood cell from a brain cell, a normal cell from a cancerous one, and perhaps even a chimpanzee from a human (Cáceres, 2003). Gene expression is modulated to a significant degree by transcription factors,

proteins that bind to specific sites along the DNA and aid in the recruitment of the cellular machinery that is involved in the process of transcription.

Gene regulation is an extraordinarily complex and multi-layered process, as Johnson notes, because "proper expression patterns often depend upon activators and repressors, interactions of tissue-specific elements with basal promoters, and other functional sequences that are often specific to each individual locus" (Johnson, 2005). Thus, although specific mechanisms of gene regulation may be well understood, relating all the pieces in a complete and coherent model remains a daunting challenge whether in the chemical, biological or computational arena.

Transcription Factors, Motifs, Promoters and Enhancers

Gene expression is modulated to a significant degree by transcription factors. Transcription factors are proteins that bind to specific sites along the DNA near a gene and regulate the gene's transcription. The site in the DNA that binds a specific transcription factor is known as a cis-regulatory element (CRE), often called simply a motif. Motifs are typically six to 20 bases long and often have a short, conserved core of bases, while other positions are less constrained.

The sequences immediately upstream of a gene are known as promoters. Promoters typically extend from a few hundred to a few thousand bases upstream from the genes they regulate. The motifs in these sequences are often positionally constrained; they cannot function as regulatory elements unless they are the correct distance from the start of the gene. Many of the binding sites in promoter sequences are involved specifically in the mechanics of gene transcription that are common to all genes. Other

promoter elements have regulatory functions that are unique to the genes they are near. Regulatory sequences farther upstream, and sometimes downstream, from the genes they regulate are known as enhancers. Although there may be constraints on the order and position of motifs within enhancer regions, such restrictions on the entire enhancer are uncommon. This is in contrast to promoter regions whose position and orientation to the genes they regulate are tightly constrained. Unlike promoters, which regulate the expression of individual genes, enhancers may regulate a number of genes in a region.

Regulatory Clusters

Binding sites for transcription factors tend to occur in clusters throughout the genome. In some cases, a region will be dense with copies of the same motif (Markstein, 2002; Lifanov, 2003). These regions are sometimes referred to as homotypic regulatory clusters. When motifs for different transcription factors are grouped together in the same region, this is referred to as a heterotypic regulatory cluster. These different types of binding clusters are collectively known as cis-regulatory modules (CRMs).

Coregulation

Although the gene regulatory network is extraordinarily complex, it is made vastly simpler by the fact that genes whose products function together are often regulated together. This is known as coregulation or coexpression. In microarray experiments, coregulation may be suggested when expression levels for multiple genes change in response to the same stimuli. Gasch summarizes the situation well:

> "This property [coordinate expression] has been frequently exploited in the analysis of genome-wide expression data, as the experimental observation that a

> set of genes is coexpressed frequently implies that the genes share a biological function and are under common regulatory control. Many proteins have multiple roles in the cell, however, and act with distinct sets of cooperating proteins to fulfill each role" (Gasch, 2002).

Microarray data can be misleading, however, as regulatory cascades in tightly coupled networks can give the appearance of coexpression. Nonetheless, knowledge of coexpression can be an extremely powerful asset because it implies that the same control region should be present near all the genes it influences.

Research Overview

We implemented and tuned a CRM-detection algorithm, LWF (Nazina, 2003), and then sought to further optimize its results. This was done in several phases. First, we identified runtime parameters for LWF that generate consistently high-scoring putative CRM (pCRM) regions across a variety of sequences. Second, we ran LWF with several different high-scoring parameter-sets and removed pCRMs that did not have overlaps among the other runs. Finally, we looked for shared motifs among the pCRMs of coregulated sequences and attempted to remove pCRMs that had low densities of shared motifs.

Chapter 2 outlines some of the major computational approaches and challenges to CRM detection. Chapter 3 describes in detail the LWF algorithm and how we assessed its performance and selected optimal parameter sets. Our efforts to winnow LWF's pCRMs through overlap filtering and motif-density scoring are discussed in Chapter 4. Further improvements and opportunities for future research are described in Chapter 5. Chapter 6 covers design decisions and other technical aspects of the tools we wrote.

Chapter 2 CRM Detection

Although CRM detection algorithms are numerous, they may be generally grouped into three categories: search by signal, phylogenetic footprinting, and search by content (Nazina, 2003).

Search by Signal

Signal-based searching is essentially the search for known patterns within a sequence. An example is the SCORE algorithm (Rebeiz, 2002) that identifies CRMs based on their concentrations of binding sites for the transcription factor Suppressor of Hairless [Su(H)]. SCORE labels regions with statistically unlikely concentrations of the transcription factor binding site as potential CRMs. Search-by-signal can be a very reliable technique because the seed pattern has been biologically verified, but it also may be seen as a liability: only regions similar to the seed-sequence will be found. This is useful when the goal is to identify multiple genes in a specific regulatory pathway, as above, but as a method for *de novo* uncovering CRMs, it is a serious limitation.

Phylogenetic Footprinting

Phylogenetic footprinting compares sequences from multiple species and looks for homologous regions among them. The assumption is that selective pressure on regulatory regions causes them to be preserved more than surrounding non-coding regions that are free to change. Conserved regions from multi-species sequence

alignments are therefore likely to have regulatory potential. Implementations of phylogenetic footprinting-based algorithms abound, including Elnitski (2003), Cliften (2003) and Grad (2004). Even algorithms not centered on phylogenetic footprinting may still take advantage of it to refine their results (Halfon, 2002; Sinha, 2003).

Although this technique is extremely powerful, it has important limitations. A critical underlying assumption is that important regulatory sequences are conserved. This often holds true, but it may also be the case that important regulatory functions have multiple regulatory mechanisms that may not all be preserved among species descending from a common ancestor. Because cross-species comparisons also will fail to identify regulatory regions that are unique to a species, they are not sufficient for exhaustively determining a species' CRMs.

Search by Content

Content-based algorithms for identifying CRMs examine differences in the statistical properties of the base and word compositions of regulatory and non-regulatory sequences; differences are assumed to imply the presence or absence of regulatory signals. That is, content-based algorithms assume that although regulatory signals may not share specific words, they are likely to share specific word distributions. Because these approaches are not initialized with particular motifs or position weight matrices (PWMs), they are not so tied to particular regulatory pathways as are signal-based searching algorithms.

A common and widely used example of this type of search is a base-frequency search for CpG islands (cytosine and guanine are linked by a phosphodiester bond, thus

CpG island, as distinct from C-G pairing across the DNA strand) in whole genome sequences. C methylation leads to increased mutability and consequently CpG dinucleotides are more rare in the genome than would be assumed by the independent C and G frequencies (Bird, 1987). For biologically important reasons, methylation is suppressed near gene promoters and in these regions CpG dinucleotides are more common than elsewhere (Durbin, 1998). Identifying CpG-rich areas in unannotated sequences is thus a way to identify potential genes. Word-frequency based approaches (Bussemaker, 2000; Rajewsky, 2002; Nazina, 2003) extend the base-frequency approach by partitioning sequences into words and examining the distribution of words in different regions.

Computational Challenges

Computational challenges to CRM and motif discovery exist at many levels. At a low level, algorithms must deal with very large search spaces: typical genomes contain 10^6 to 10^9 nucleotides. It is possible in some cases to significantly pare the sequences to be analyzed by looking only at the sequences immediately surrounding coding regions. Additional aggravating factors important to the accuracy of the search algorithms but beyond the scope of this paper include modeling the genomic background and determining the statistical significance of a hit in light of the background model; determining whether the position of a regulatory sequence relative to the gene it modulates is significant; and determining whether a motif's position on the + or - strand affects its ability to have regulatory significance.

Assuming these intrinsic issues are accounted for, the higher-level problems of how to manage run-time parameters to get consistent results and avoid over-fitting are real issues.

Nuisance Parameters

Many applications have adjustable run-time parameters that affect the performance of the underlying algorithm. Although this ability to "tune" an algorithm to get the "best" results seems like an obvious benefit, it is in fact a double-edged sword. When working with test data that has CRMs embedded in known positions, it is easy to measure how different parameter values affect the output using objective measures such as the phi-score (Pevzner and Sze, 2000). When working with un-annotated data, however, it is not possible to measure an algorithm's performance except on the laboratory bench where experiments can verify biologically what is predicted computationally. In one study that used computational techniques to identify potential regulatory regions (Berman, 2002), for example, *in vivo* verification found that 18 of the 37 predictions were false-positives (Berman, 2004).

When biologically verified data can be used to bootstrap an algorithm (as in the case when searching for new binding sites for a motif with a known position weight matrix [Mount, 2001; Stormo, 1989; Stormo, 2000]) or an algorithm's application (as in the case where experimental data is thought to resemble test data, implying that similar optimizations may apply), run-time parameters have undeniable value. When little is known about the test data, however, adjustable parameters are little but a nuisance.

Overfitting

Overfitting an algorithm to a small test set is a problem closely associated with the tweaking of runtime parameters. Because it is possible to objectively measure an algorithm's performance against annotated test data, it is likewise possible to adjust an algorithm's runtime parameters to maximize performance against individual sequences. This can have an effect very much like being stuck in a local minimum in a simulated annealing algorithm: although the results appear good, in fact they are sub-optimal. Under the right circumstances, variable parameters can allow an algorithm to be tuned when certain characteristics of the training and test data are available. In practice, such knowledge is often hard to come by and in these circumstances the ability to adjust parameters can do more harm than good.

Chapter 3 Selecting and Tuning a CRM Detector

We considered implementing several different algorithms and ultimately selected a word frequency-based algorithm developed by Nazina et al (2003). Other algorithms we considered were Ahab (Rajewsky, 2002), Stubb (Sinha et al, 2003), a phylogenetic footprinting tool (Elnitski et al, 2003) and CisModule (Zhou and Wong, 2004; EMCModule [Gupta and Liu, 2005] is similar). We picked Nazina's algorithm (dubbed LWF for Local Word Frequency) for several reasons. It does not require as input known transcription factor binding sites, as do Ahab and Stubb. It does not require as input an aligned sequence from a related species as do phylogenetic footprinting algorithms. It can also accommodate long input sequences, in contrast to CisModule, which is optimized for sequences with enhancer regions no more than a few thousand bases from the genes they regulate.

LWF Algorithm Details

LWF is a content-based search algorithm that constructs statistical models of word distributions in positive and negative training sequences and then uses these models to categorize un-annotated sequences. The model-building step is known as training; the categorization step is known as scanning. A detailed explanation of the algorithm follows.

In a given sequence window l, the frequencies F of a word in each position i are collected and sorted into frequency-channels j. That is, the words in any given window

are grouped according to how frequently they occur within the window, and these groups are called frequency-channels. A channel's feature score $S_j(i)$ represents how many words in a window have a frequency within the range $j <= F < j + n$ where n is an integer constant. When a channel's j-value is high it contains highly repetitive words; when its n-value is high it contains words with a broad spectrum of frequencies.

The feature scores S of each window in the sequence are used to build a statistical model of a functional class, e.g. as a regulatory (ω_1) or background (ω_2) region. The distribution of S in each channel $E(S)$ can then be used to solve the classification problem:

$$L_j = \log(E(\omega_1 \mid S_j) / E(\omega_2 \mid S_j))$$

In each position i of each channel j, L_{ji} describes the probability that the window beginning at i is or is not a member of ω_1. The score R for a given window beginning at i is calculated simply as the sum of L_{ji} scores over all frequency channels, with some corrections applied to temper extreme values and unlikely classifications. R_i thus represents the likelihood that the entire window beginning at position i is or is not a member of ω_1. LWF calculates R-scores for windows beginning at every position in the input sequence and then removes a percentage of the lowest scoring segments.

The output from LWF's training phase is a model file that summarizes the statistical properties LWF calculates about its input sequences. In the scanning phase, LWF uses one or more pairs of positive and negative training models to analyze the unknown sequence. The model files and the sequence file are both fed into the algorithm;

its final output is a FASTA-formatted file (FASTA, 2005) containing the sections of the sequence file that the algorithm has labeled as potential CRMs.

Datasets

We used published CRM data for annotated sequences from the model organism Drosophila Melanogaster (Nazina, 2003). The concept of a model organism refers to species whose genomes are widely studied and well understood. In many cases, a model organism's entire genome has been sequenced and much of it has been annotated, meaning that the positions of specific genes, promoters and enhancers are known. The genes in this dataset are all related to embryo development, a particularly well-documented network in D. Melanogaster. This makes it an attractive system to study because it is more likely that changes affecting gene transcription, such as the alteration of a transcription factor's binding site, are likely to have directly observable consequences.

We considered developing tools to generate synthetic datasets but decided that enough biological data was accessible through existing sources. Nazina et al (2003) extensively mined published literature to recover more than 60 CRMs for 20 developmental genes. We converted their raw data into a relational database and reconstructed the static web pages they provided as an application with a web-based front end and a Perl-based API. This allows for more flexible navigation through the data, and also makes it easy to supplement the data as more portions of the D. Melanogaster genome are annotated. By developing a generic schema and writing our applications

against that interface, rather than against individual text files, the applications too are significantly more extensible.

The original dataset contains details about the regulatory regions of 20 genes. We set five of them aside as a test set and divided the remaining 15 into three groups of five. For each set of five, we used CRMs from the remaining 10 sequences as positive training data.

Performance Metrics

LWF has a variety of runtime parameters, and different settings can cause its performance on different sequences to vary widely. In order to be able to use LWF to search for pCRMs in un-annotated sequences, we needed to determine the set of parameters that would give us the most consistent results across a variety of sequences. We opted to calculate the following statistics as quantitative measures of performance:

Phi-score

true-positives / (false-positives + false-negatives + true positives)

The phi-score measures how closely predicted regions match true regions. It has become a standard metric for describing the performance of motif-finding programs since it was first described in Pevzner (2000). Although this score was developed in the context of motif finding it is applicable here as well.

Precision

true-positives / (false-positives + true-positives)

Precision measures how closely an algorithm's positive results match the true-positive data. An algorithm that finds many true-positives and few false positives will have good precision. Because precision does not take false-negatives into account, an algorithm that misses many true-positive but has few false-positives will still have good precision.

Sensitivity

true-positives / (false-negatives + true-positives) (Pagano and Gauvreau, 2000)

Sensitivity measures how well an algorithm covers the positive results. An algorithm that has few false-negatives will have high sensitivity. Sensitivity is important in determining whether an algorithm is good at covering the positive results, but it does not measure how much false-positive data is also returned, along with the true-positives.

Specificity

true-negatives / (true-negatives + false positives) (Pagano and Gauvreau, 2000)

Specificity measures an algorithm's false-negative rate. An algorithm that has few false-positives will have high specificity. Specificity differs from precision, which will also be high when false-positives are low, in that it measures the goodness of fit to the true-negative data. Like precision, however, specificity does not consider false-negatives, so an algorithm can have a good specificity score even while it misses many true-positive regions.

Harmonic Mean

$n / (1/x_1 + 1/x_2 \ldots 1/x_n)$

The harmonic mean is useful for calculating an average of rates. The special case of calculating the harmonic mean for two rates, x_1 and x_2, is given by $(2x_1x_2) / (x_1 + x_2)$. Here, we calculate the harmonic mean of sensitivity and precision.

Signal Strength

sum(each CRM sequence-length) / total sequence-length

We calculated the proportion of each sequence occupied by CRMs because it appeared to vary widely and we wanted to see if and how this variation would affect our results.

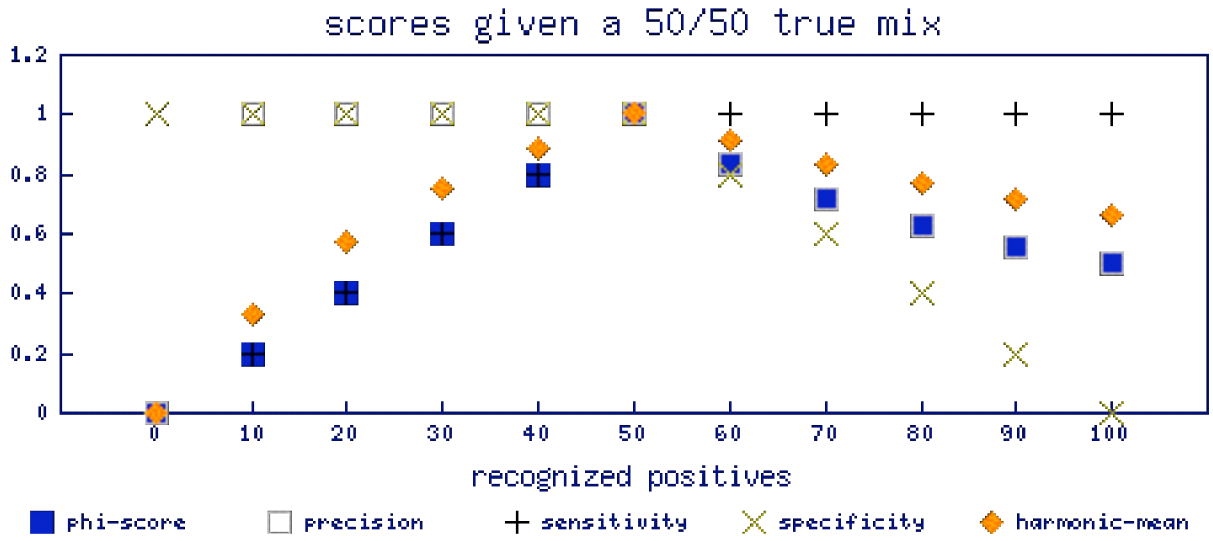

Figure 3-1 Outcomes of different statistical measures of a dataset that should be recognized as 50% positive and 50% negative. The X-axis shows the percent of the dataset that is recognized as positive; the Y-axis shows each measure's score.

Figure 3-1 plots the outcomes of several of these statistical measures in order to highlight differences in how they score the same data. Several trends are worthy of note here. Phi-scores match sensitivity scores when the positive count is underestimated, and they match precision scores when it is overestimated. The harmonic mean of sensitivity and precision follows the same trends, although its penalties for miscounting are not so

severe. Unlike precision, sensitivity and specificity, these functions penalize both undercounting and overcounting, hence their utility in describing the overall goodness of fit of pCRMs to CRMs.

As noted above, LWF is the first phase of a pipeline of tools that each aim to remove small regions from a sequence that do not contain regulatory signals. Regions cannot be recovered once they are removed, so in the early phases of the pipeline it is important to prune very conservatively. In other words, false positives are not a big problem early in the pipeline. In fact, because we plan to winnow pCRMs in multiple stages, there should be a relatively high false-positive rate in the early steps. We chose to use the harmonic mean of precision and sensitivity, rather than the phi-score, for precisely this reason. (This score is abbreviated "harmonic mean" through the rest of this paper.) Even though the phi-score is a widely accepted measure of performance for motif-detection algorithms, the harmonic mean's penalty for false-positives is not so severe as the phi-score's (Figure 3-1). This makes it a better score for our needs.

Exploring the Parameter Space

Our goal in tuning LWF is to find sets of parameters that give consistently good results across all or most sequences in our test sets. LWF's training algorithm has four parameters: word-length, window-length, channel-count and mismatch-count. The suggested ranges and default values are as follows (Nazina, 2003):

Parameter	Range	Default
word-length	1-12	5
window-length	64-512	128
channel-count	2-24	10
mismatch-count	0-4	1

We generated training models for each sequence in our test set with all combinations of the following:

Parameter	Values
word-length	3, 4, 5, 6
window-length	50, 100, 200, 400, 600, 800, 1000
channel-count	9, 12, 15, 18, 21, 24
mismatch-count	0

We anticipated that long windows, short words and a high number of frequency-channels would give us the best results. Although longer words may be more biologically meaningful, it has been shown that motifs tend to consist of highly conserved cores with degenerate extensions (Carlson et al, 2006a). With regard specifically to LWF, longer words will generate fewer channels than will shorter words, reducing the statistical resolution of the model because the signals within each channel are smoothed out to provide a single score for the entire channel. When more channels are present, less smoothing occurs because the values in each channel are closer together. We opted not to allow mismatches at all because of concerns that that mismatches for short words would dampen the signal too much. In retrospect, we should have incorporated mismatching; the section Scanning Parameter: Mismatch-count discusses this in detail.

Given these parameters, there are 2520 possible result sets; due to several factors, only 1546 were actually generated. In some cases the input sequence was simply shorter than the given window-length. In other cases there is not enough data in short windows with long words simply to fill many channels. Other non-generated test-sets are due to how LWF calculates channels. The channel-count input parameter is actually a maximum threshold rather than a fixed value. LWF calculates an expected distribution of word

counts based on the word-length and window-length parameters and this distribution is then used to configure the maximum values that can appear in each channel. Because of the variability in calculating the probability distribution, some channel-counts were omitted.

Distinct vs. Composite Training Sets

In addition to variable runtime parameters, LWF's performance is affected by how its training sequences are created. Positive examples from separate sequences can be joined together to create a single model, or LWF can create separate model files for each separate positive sequence. The advantage of composite training set mode is that subsequent scanning will be very fast because there is only one training model to parse.

There are two issues with composite training. First, in order to incorporate new training sequences, the entire model must be rebuilt. Second, the model created by training on so many sequences at once may actually be too generic and may provide poor results. The advantage of distinct training mode is that new training data can easily be incorporated into the scanning process, but the disadvantage is that scanning more models takes more time.

Summarizing Results

To summarize LWF's output in a way amenable to further statistical analysis, we wrote a tool (hm.pl) that collects the output from multiple runs of the algorithm and, using our sequence database, calculates statistical properties for each test set's pCRMs as they relate to each specific parameter set. Figure 3-2 and Figure 3-3 show variation in the performance of different parameter sets on the 15 test sequences. Figure 3-2 shows that

although there is quite a bit of variation in performance against different sequences for the same parameter set, performance across parameter sets is quite consistent. This is demonstrated by the high degree of overlap among the middle 50% (red boxes) of highly ranked parameter sets.

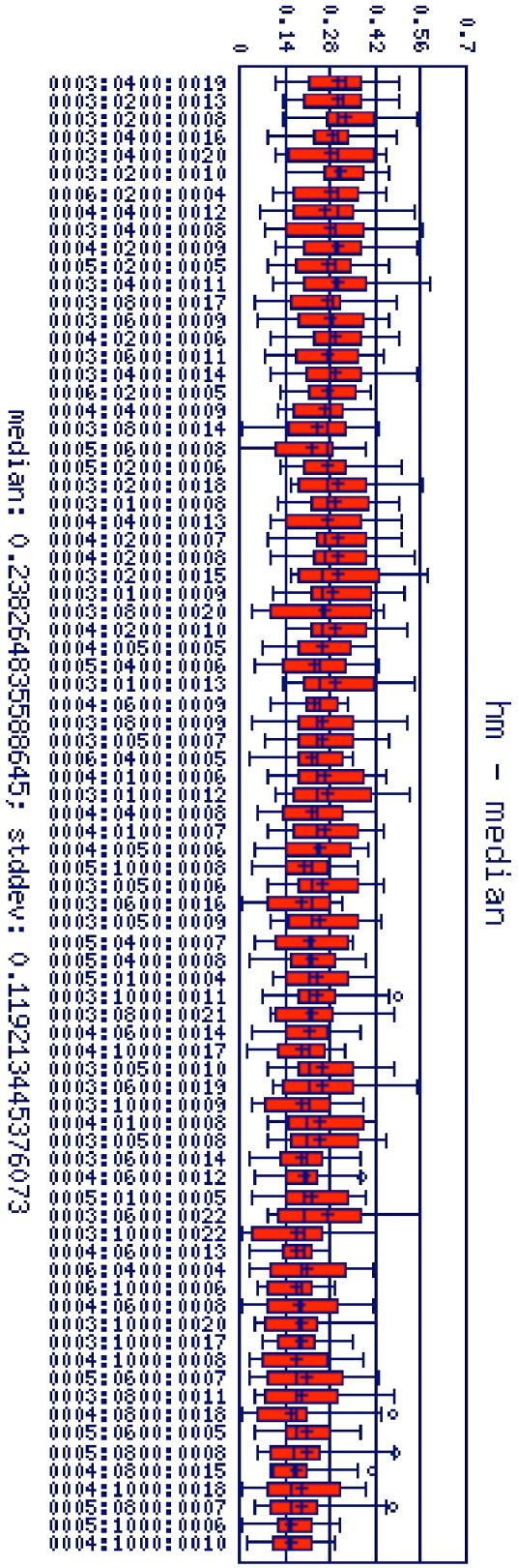

Figure 3-2 Performance of different parameter sets. Each bar represents variation in the performance of a given parameter set among the 15 sequences. The X-axis shows the harmonic mean; different parameter sets, sorted by their median harmonic mean scores, are displayed along the Y-axis. The three numbers in the Y-axis labels indicate each parameter set's word-length, window-length and channel count. In each bar, the middle 50% of the scores are contained within the red box; within the box, the mean is labeled by a + and the median by a |. The whiskers indicate values that fall within 3/2 the interquartile range; values beyond that range are indicated by small circles.

Figure 3-3 presents a different view into the same data and does a better job of showing correlations among the different training set components (word-length, window-length and channel-count) because each is plotted on a distinct axis. In general, high scores tend to be affiliated with short words and medium-length windows. There are good scores for high and low channel-counts within these constraints, suggesting that channel-count does not drive the algorithm's overall performance.

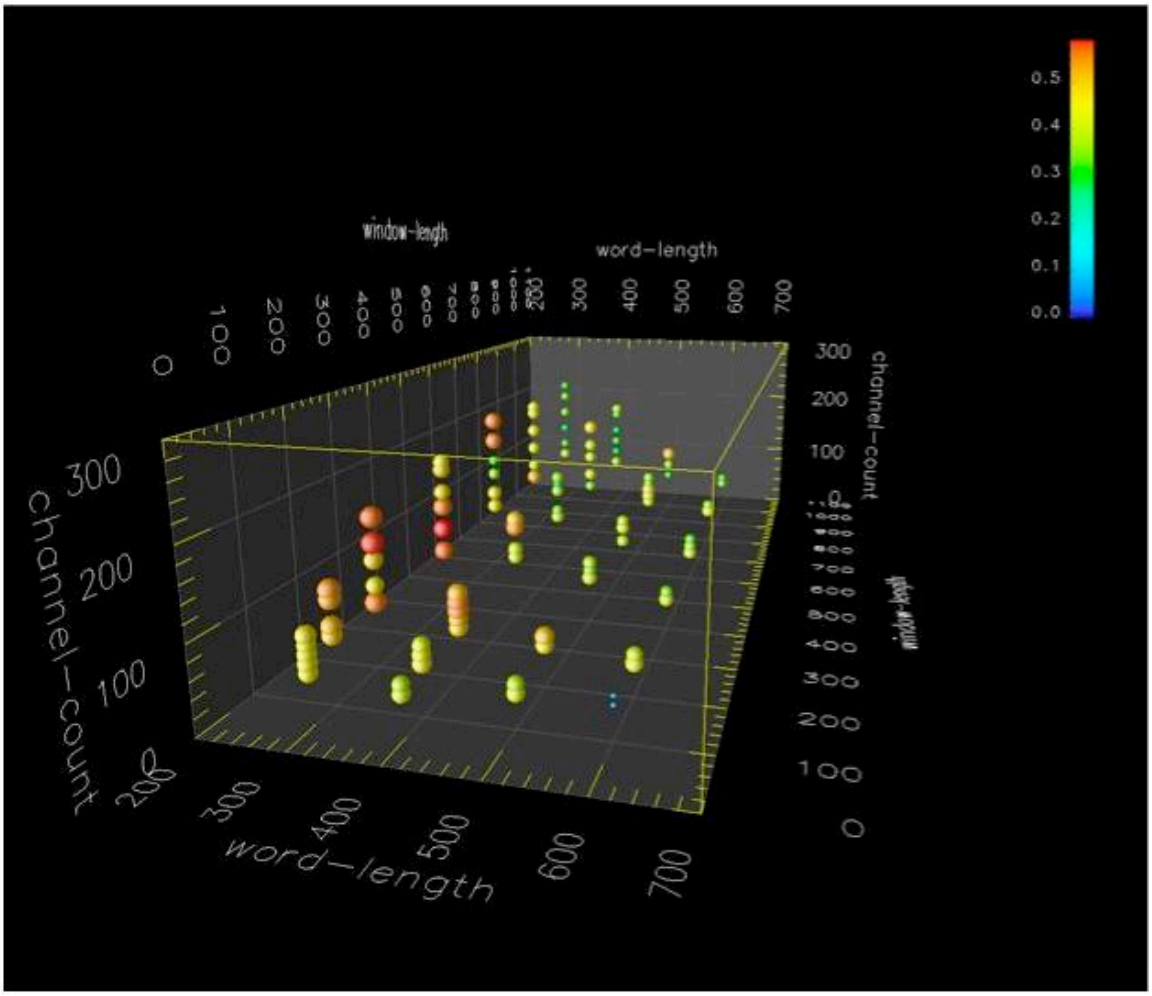

Figure 3-3 The three training run-time parameters word-length, window-length and channel-count vary along the x, y and z axes, respectively (parameter-values have been normalized to accommodate a single scale across all three axes). The color of the point represents the average harmonic mean of the training sets that share the same parameters (0.6 is red; 0.0 is blue). The size of the point represents how many datasets shared a given set of parameters.

We also constructed simple two-dimensional plots to display overlap among the biologically identified CRMs and the pCRMs identified algorithmically by LWF (Figure 3-4). These plots show pCRM blocks (gray) superimposed over CRM blocks (dark blue) and display a normalized *R*-score for each position in the sequence. Recall that an *R*-score represents the likelihood that a window beginning at a given position should be recognized as a pCRM. The plots also include a percent-identity-profile (PIP, i.e. phylogenetic footprint) between Drosophila Melanogaster, the test species, and Drosophila Pseudoobscura, a close relative.

Several important trends are visible in Figure 3-4. First, the pCRMs overlap the CRMs to a significant degree. Although many non-overlapping (background) regions are also labeled as pCRMs, the fact that so many of the true CRMs have overlaps indicates that LWF's core algorithm works well. Second, there is a high degree of correlation among the R-scores (black lines) of the top-scoring parameter sets. This reinforces the conclusion drawn from Figure 3-2 that the top-scoring parameter sets tend to perform consistently on the same sequences. Finally, note that the R-scores and PIP scores often follow very similar trends. The fact that these two scores align well is a crucial point. LWF and phylogenetic footprinting are entirely unrelated algorithms and yet they score similar regions in similar ways. This strongly suggests that there is in fact a signal embedded in the raw sequence data and that it can be uncovered.

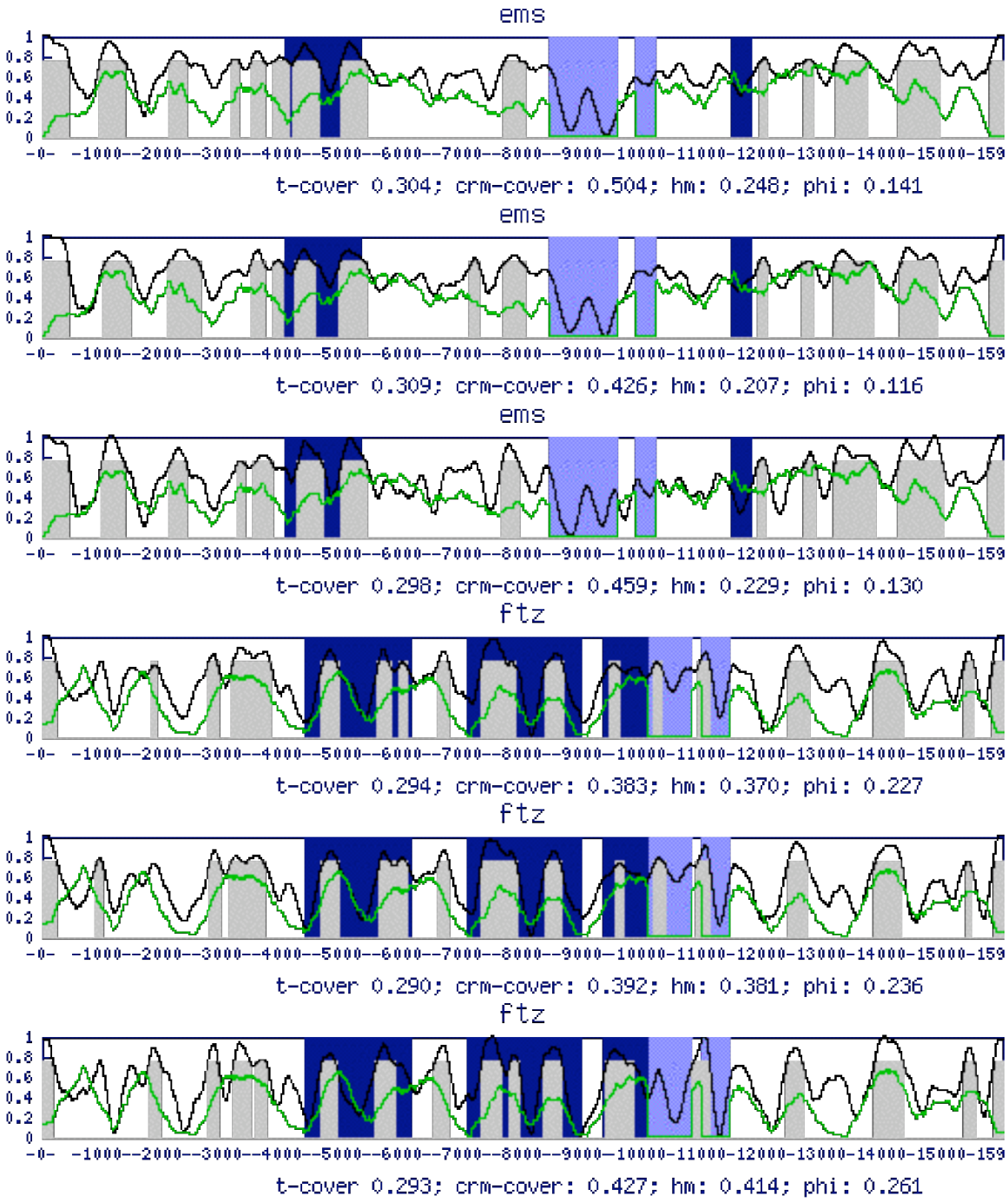

Figure 3-4 pCRMs (gray bars) and CRMs (dark blue bars) from three high-scoring parameter sets for Empty Spiracles (ems) and Fushi-Tarazu (ftz); exons are also plotted (light blue bars). The normalized R-score is plotted in black; the PIP score is in green. The X-axis shows the position in the sequence; the Y-axis shows the normalized R-scores and PIP scores. The t-cover score indicates the amount of each sequence that is identified as belonging to pCRMs; it is calculated simply as sum(each pCRM-length) / sequence-length. The crm-cover score indicates how well the pCRMs cover the CRMs, over all; it is calculated as sum(overlapping pCRM length) / sum(each CRM length). The hm and phi values are the harmonic mean and phi-score, respectively.

Good Results vs. Consistent Results

It is interesting, though perhaps not surprising, to note that the parameters that provide consistently high scores across multiple sequences are not the optimal parameters for individual sequences.

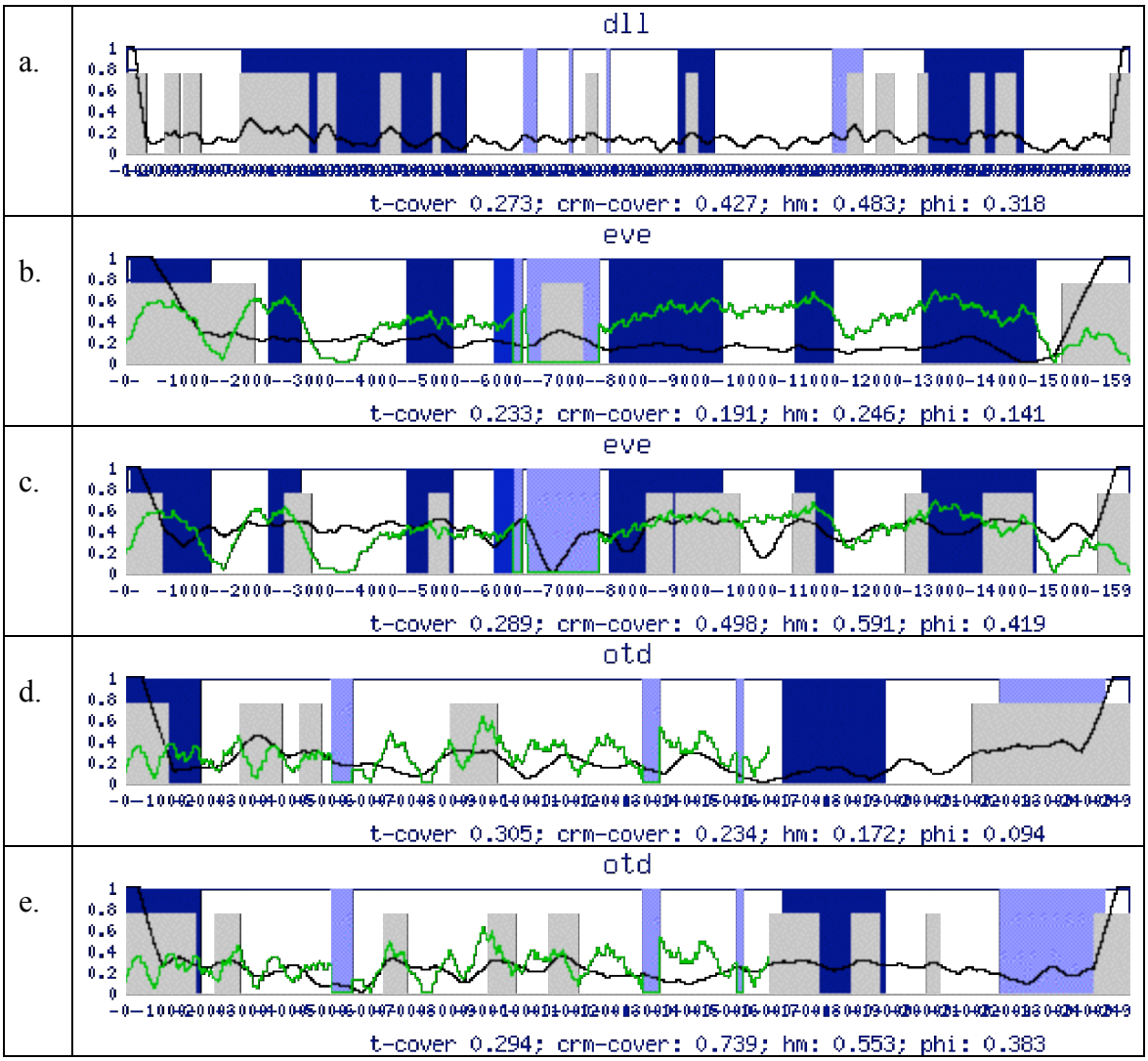

Figure 3-5 A plot of the pCRMs from the top-scoring parameter set for the gene Distalless (dll) is shown at top (a). The normalized R-score is plotted in black; the PIP score is in green. The X-axis shows the position in the sequence; the Y-axis shows the normalized R-scores and PIP scores. The same parameter set used in (a) is also used to analyze Even-Skipped (eve) in (b) and Orthodentical (otd) in (d); analyses using the optimal parameter sets for Even-Skipped and Orthodentical are shown in (c) and (e), respectively.

Figure 3-5 demonstrates clearly the potential for overfitting if the data set is too small. These figures show what would happen if the optimal parameters for uncovering CRMs in Distalless were used to scan for CRMs in Even-skipped (eve) and Orthodentical (otd). Scanning profiles made with the optimal parameter sets for each sequence are also displayed in order to highlight how drastically different local optimizations can be. Note in particular the drastic variation among the crm-cover, hm and phi-score values while the t-cover score remains steady, and the poor correlation between the *R*-scores (black lines) and the D. Pseudoobscura percent identity profiles (green lines). The t-cover score remains relatively stable because it directly reflects a constant (threshold-cutoff) that LWF uses to determine the balance between regulatory (ω_1) and background (ω_2) regions. This means that variation among the other runtime parameters affects not how much of a sequence is positively classified, but how well the positively classified regions align with the true-positive regions.

Chapter 4 Refining LWF

We selected a set of LWF's top-scoring parameter sets, as determined by their harmonic means, and then began looking for ways to further improve the results. In many cases, LWF returns a collection of 10-15 short pCRMs, and although many of them overlap true CRMs, others are simply false-positives. When working with annotated sequences, removing the false positives is simply a matter of ranking the pCRMs using an objective function, such as the phi-score, and removing the low-ranking pCRMs. With unannotated sequences, however, the situation is more complicated because the scoring function must work in the absence of information about true CRMs. To work around this issue, we attempted to winnow LWF's results in two different ways. First, because there was not a clear winner among the top-performing parameter sets in LWF, we combined pCRMs from multiple parameter-sets and removed non-overlapping pCRMs. Because the parameter-sets had only small differences, this helps find the cores of pCRMs that are common to all runs of the algorithm.

Second, we looked for ways to take advantage of knowledge about coregulation of genes. Biologically, examining coregulation is a two-part question: are a set of genes coregulated, and how? Here, the first question has already been answered and all that remains is how to attack the second. Often, the mechanism of gene coregulation is for the same transcription factor to bind upstream of all the coregulated genes. Based on this assumption, we reasoned that all the CRMs of coregulated genes must therefore share something in common. We posited that it would be a motif for a shared transcription

factor and that this motif would be part of any shared CRM. We explored several methods of looking for common motifs among pCRMs in coregulated genes. Although this method requires knowing that sequences are coregulated, the barrier to gaining knowledge of coregulation is substantially lower than that to knowing the positions of the regulatory elements within a sequences.

Selecting Optimal Parameter Sets

Based on the results of our parameter space sampling, we selected parameter sets that performed well across all sequences. As noted above, although some parameter sets provide excellent scores on select sequences, those sets often failed to perform well across the board. This makes them poor candidates for use across sequences, where consistent performance is crucial. We selected the following parameters, based on the results displayed in Figure 3-3:

word-length: 3, window-length: 200, channel-count: 8
word-length: 3, window-length: 200, channel-count: 13
word-length: 3, window-length: 200, channel-count: 18

Multiple Runs and Overlaps

Some parameter-sets clearly offered better results than others for specific genes but no single set stood out against the others across every sequence. This is shown clearly in Figure 3-2 where the middle 50% of the distributions for the top six parameter-sets overlap almost completely. Although there is significant overlap among the harmonic mean distributions, however, in many cases there are specific and important differences among particular parameter-sets, as demonstrated in Figure 3-4. Note, for example, the

pCRM in the second Empty Spiracles (ems) run near position 7000, and the pCRMs near position 2000 in the first and third Fushi-Tarazu (ftz) run, and the pCRM near position 1000 in the second Fushi-Tarazu run. Apart from these outliers, most of the other differences among pCRMs from top-scoring parameter-sets are differences of degree; similarly sized pCRMs show up in roughly the same locations, shifted slightly up- or downstream.

We reasoned that pCRMs that were identified in multiple parameter sets are more likely to be true CRMs and opted to filter results from top-scoring runs by removing pCRM sequences without any overlapping bases among runs of LWF with the top three parameter-sets. Rather than strict filtering and removing all sequences without overlap, as long as any part of a pCRM overlapped with two other runs, we chose to keep the entire pCRM. This avoids pruning the sequence too aggressively at the beginning of the pipeline. We considered other approaches to managing overlaps and settled on the relaxed, triple-overlap algorithm for the following reasons. In comparing only two sequences, selecting true-overlaps between the two sequences is found to be too restrictive, but more relaxed approaches are not sufficiently discriminatory. With three tests sets, using relaxed overlap matching works well and is not computationally prohibitive. We modified the pCRM-CRM overlap plots described above to highlight the pCRMs this overlap algorithm removes (Figure 4-1).

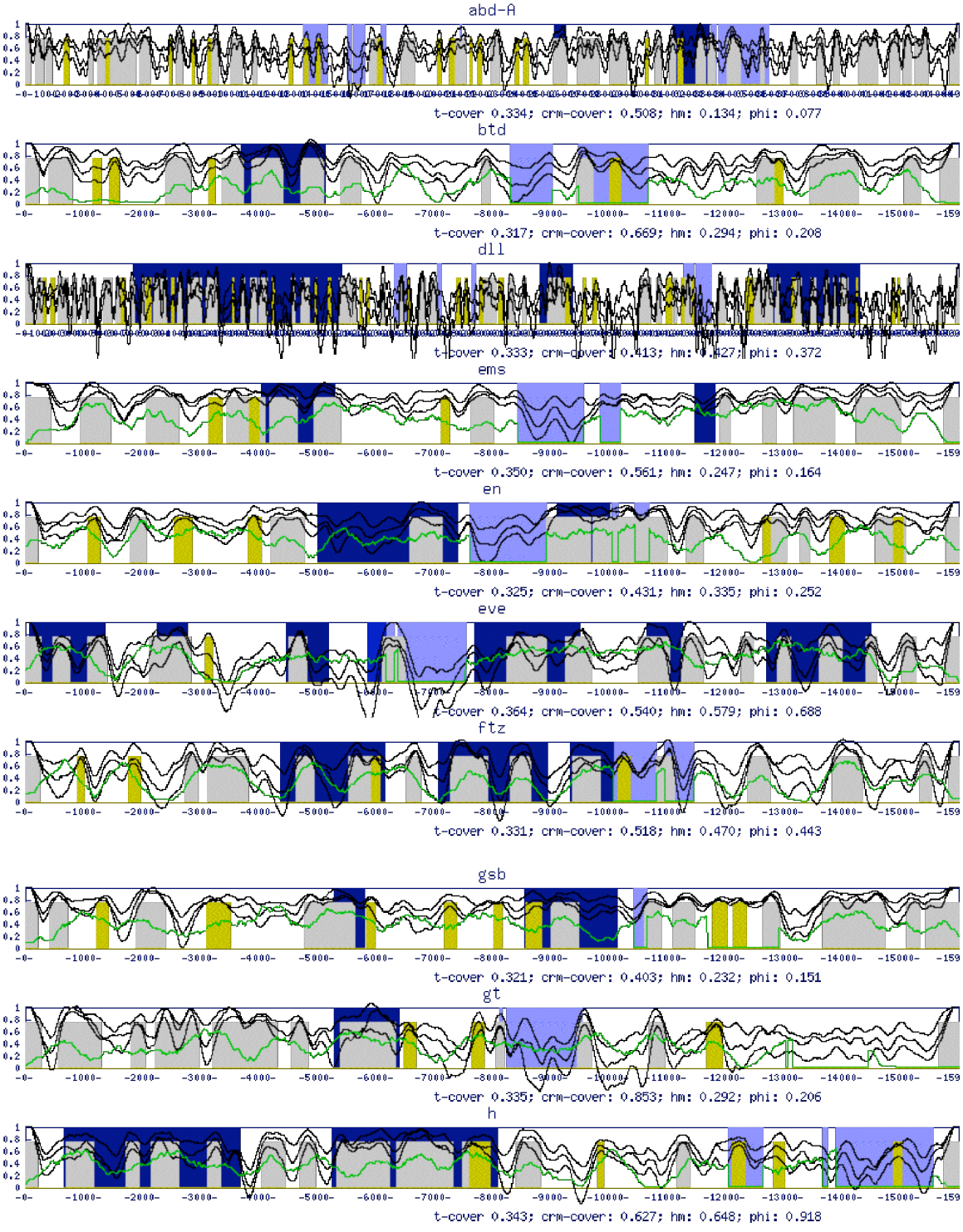

Figure 4-1 pCRMs without overlaps among three runs of LWF (yellow bars) will be removed; those with overlaps (gray bars) will be kept. Dark blue indicates CRMs; blue indicates exons. The X-axis shows the position in the sequence; the Y-axis shows the normalized R-scores (black lines) and PIP scores (green lines).

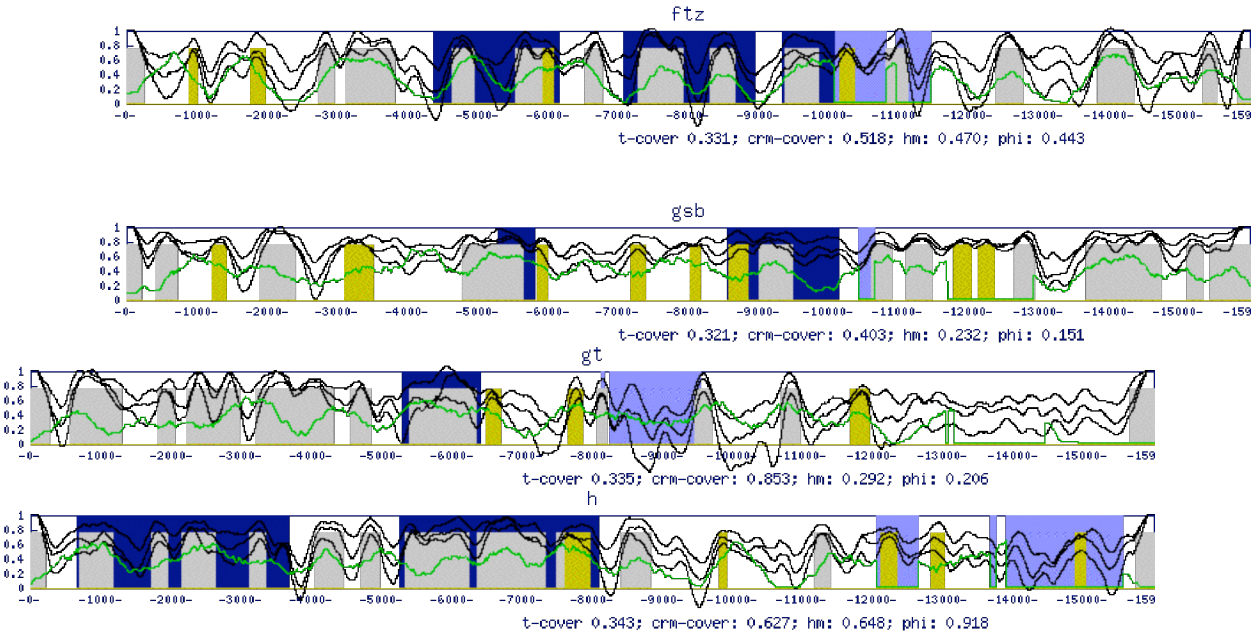

Figure 4-1 makes clear that LWF's statistical model of the true-positive regions is very robust and has good specificity. In most cases, the differences among the true-positive pCRMs are differences of degree (pCRMs are in similar positions but are shifted slightly up- or downstream). Many of the false-positives, however, are in wholly different positions, and this is what allows the overlap algorithm to work. This is made more clear in Figure 4-2, which plots the top-scoring parameter sets' harmonic means against those of the default parameters as defined in Nazina, and Figure 4-3, which plots the post-overlap harmonic means against the pre-overlap scores. In both cases, the gain in precision is generally modest but also relatively consistent, with two noteworthy departures: scanning with the default parameters fails to uncover any pCRMs for Kruppel (kr) and Paired (prd), but good pCRMs are uncovered for both with the optimized parameter sets we selected.

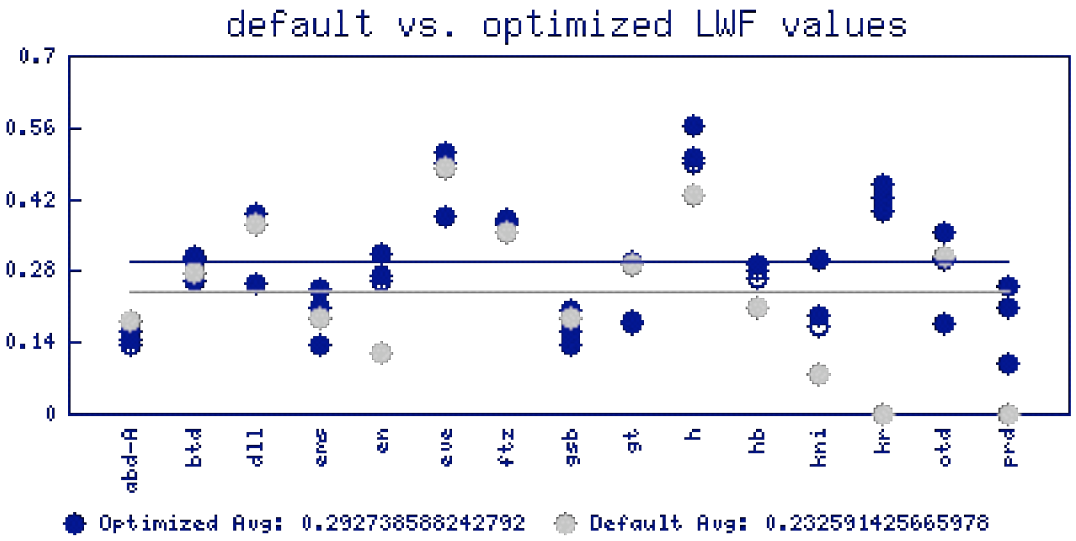

Figure 4-2 Harmonic means for each sequence scanned with the default parameters (gray) and the top three optimized parameter sets (blue). The X-axis shows the different sequences that were scanned; the Y-axis shows the harmonic mean. The gray and blue lines show the average harmonic means for the default and optimized parameters, respectively.

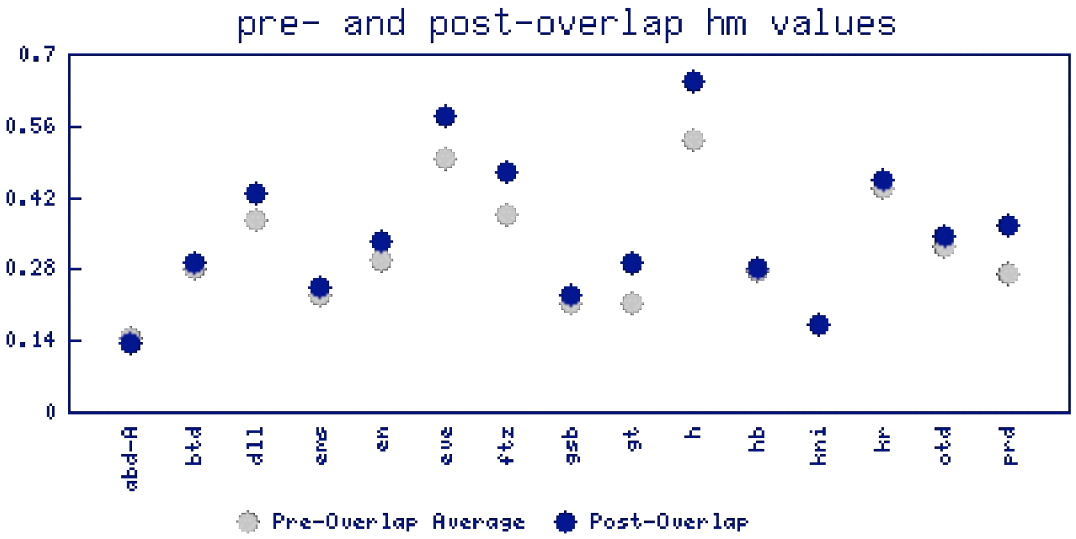

Figure 4-3 Harmonic means for each sequence before overlap removal (gray) and after it (blue). The X-axis shows the different sequences that were scanned; the Y-axis shows the harmonic mean.

Using Coregulatory Knowledge

As noted in chapter 1, genes are often turned on and off in sets. This is known as coregulation because in many circumstances a single transcription factor is the primary actor affecting the regulation of multiple genes. The reality may of course be vastly more

complicated as gene networks can have tens, if not hundreds, of factors acting to activate or repress transcription. A discussion of the details of the microarray experiments (Mount, 2001) that provide this data or of the computational and mathematical frameworks that have been developed to model these interactions is beyond the scope of this paper, but the core notion of genes being regulated in sets is germane: many of the genes in our dataset are known to be coregulated (Lifanov, 2003; Nazina, 2003).

Because LWF's algorithms compare new sequences to existing statistical models, LWF cannot take advantage of knowledge about relationships between sequences, i.e. knowledge of coregulation. All LWF can do is compare a model of an unknown sequence to that of a known sequence and answer the question, "Are they similar?" It cannot also answer questions such as, "Are models of these two unknown sequences similar?" Unfortunately, this is precisely the question that must be answered when looking for pCRMs that come from sequences that are coregulated. If the same transcription factor modulates the expression of two different genes, then the CRMs of those genes must share something in common that causes their shared transcription factor to bind to both sequences.

One issue with this approach is that if a gene is involved in multiple regulatory networks, the process of winnowing based on motifs common to all the genes in a coregulated set may inadvertently cast out pCRMs that are valid but that are not involved in the current coexpression network. For example, in our training set, the gene Even-skipped has seven known CRM sites and eight different regulators. None of the regulators have binding sites in all the CRMs. If we are studying the coregulatory

network of genes regulated by Bicoid, only three of Even-skipped's pCRMs should share motifs with other genes, meaning that four true-positive regions would be filtered out.

Searching for Shared Motifs

Knowing that sequences are coregulated, we attempted to winnow the pCRMs identified by LWF by searching for motifs across all the pCRMs from a set of coregulated genes and ranking the pCRMs within each gene based on the density of those motifs.

Motif searching is a complicated process in its own right and there is a legion of tools dedicated to that task. Examples include Consensus (Hertz, 1999), MEME (Bailey, 1995), Gibbs Sampler (Lawrence, 1993) and AlignACE (Roth, 1998) (see also Bussemaker, 2000; GuhaThakurta, 2001; Keich, 2002; Pevzner, 2000; Rigoutsous, 1997; Sinha, 2000; van Helden, 1998; Zhu, 2002). MEME and Gibbs Sampler both use modified expectation maximization (EM) algorithms to find the conserved patterns in related sequences. EM works by starting with a random alignment that is used to generate a scoring matrix. Each sequence is then matched against the matrix, which is then updated to maximize the alignment of the sequences. This process is repeated iteratively until no changes are made to the matrix. Gibbs Sampler works similarly but omits one sequence from each round of alignment and then uses that left-out sequence's score against the matrix to update it (Hudak, 1999).

Another approach to motif finding called beam searching is based on the assumption that over-represented long motifs must contain on over-represented shorter motifs. A beam search works by starting with a short core, iteratively expanding it, and

pruning all but the highest scoring cores prior to the next round of expansion. This pruning vastly limits the search space, the enumeration of which is a major drawback in many other algorithms (Carlson et al, 2006a). Beam searching is at the core of SCOPE (Suite for COmputational identification of Promoter Elements), a collection of motif-finding algorithms being developed in our lab. SCOPE returns motifs ranked by their "significance score", a log-odds score that conveys the statistical likelihood that a given motif occurs at a greater frequency than would be expected based on the background frequency of that motif in the genome. Low scores correlate with motifs that are as common elsewhere in the genome as in the source sequence; high scores suggest that the motif may have regulatory significance because it is rare in the rest of the genome, or because it occurs frequently in test sequences, or both. Our strategy was to use SCOPE to filter out pCRMs that did not contain any overrepresented motifs.

We started by building a basic framework for iteratively ranking and trimming pCRMs based on their motif content as returned by SCOPE. There are three phases to each round of the iteration. First, we search for motifs in the pCRM sequences using SCOPE, then we rank pCRMs based on their motif densities. Finally, we remove pCRMs with low motif densities. As when exploring LWF's parameter space, we wrote a series of scripts to run different ranking algorithms with different parameters in order to determine the optimal settings. Input options include:

motif-count: how many of the top scoring motifs should be used to rank the pCRMs
iteration-count: how many times to iterate between ranking and winnowing
drop-count: how many pCRMs should be dropped in each round
zero-count: how pCRMs with no motif hits should be removed (one at a time or all at once)

We also updated our CRM-pCRM overlap diagrams to incorporate motif position data in place of LWF's feature score information, and to shade pCRMs based on their rank. We scored pCRMs within each sequence based on their density of the motifs from SCOPE with the highest sig-scores. We used motif density rather than raw motif counts in order not to favor long pCRM windows with only a few motifs over short pCRM windows with a higher density but lower over all count.

Motif-score Based Filtering

In our first iteration of ranking, we tested the top four and top seven motifs, as ranked by SCOPE, and kept 90% of the pCRMs from each sequence in each round. When multiple pCRM blocks lacked motif hits, we removed them one at a time at random. It quickly became apparent that this simple ranking algorithm was not sufficient. Because the distribution of pCRMs from sequences could vary greatly, SCOPE sometimes returned motifs that scored well only in some sequences. This is indicated clearly in Figure 4-4 where Even-Skipped (eve) lacks any instances of the fourth and seventh top-scoring motifs. Like LWF, SCOPE is not sensitive to the origin of the input sequences. Thus, although we are exclusively interested in motifs that appear in all the input sequences, SCOPE does not accommodate this constraint. It should be emphasized that this is not a limitation of SCOPE but merely a reflection of how we are using it. SCOPE identifies motifs that are over-represented in its input as compared with the genomic background.

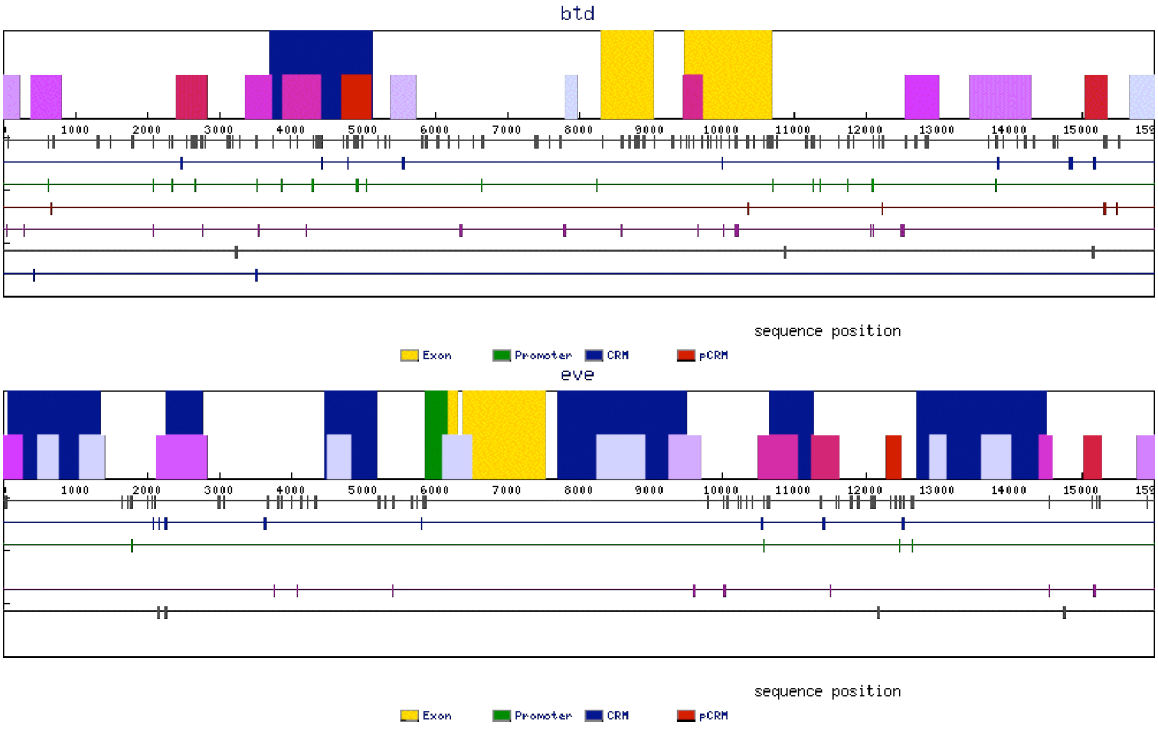

Figure 4-4 Positions of top-scoring motifs as identified by SCOPE in Buttonhead (btd) and Even-Skipped (eve). Motif positions are labeled by vertical tics below the tall bars (CRMs, promoters and exons) and short bars (pCRMs). The pCRMs (short bars) are colored based on the density of motifs they contain. Dark red represents high density; light blue represents low density.

What Figure 4-4 makes clear is that this approach to motif identification does not address the assumption underlying this approach, namely that coregulated genes should have CRMs that share common motifs. Instead, motifs from pCRMs in one or two of the input sequences dominate the analysis and little is learned about what all the pCRMs have in common.

Gene-coverage Based Filtering

We attempted to ameliorate this issue by filtering out motifs that did not occur in at least one pCRM in each gene in the set. Given our working assumption that a shared transcription factor binding site is the mechanism of coregulation, motifs that are not shared across all genes in the set are spurious and should not be used to rank pCRMs. We

refer to this as a "coverage" filter because only motifs that "cover" all the genes in the set are carried forward to the pCRM-ranking process. When working solely with sig-scores, we could simply take the top motifs from SCOPE and rank each pCRM based on the density of those motifs. This was fast because we only searched for the top motifs. Incorporating coverage as part of a motif's score means that many more motifs must be scored in case motifs with low sig-scores but high-coverage scores could bubble up and out-rank motifs with high sig-scores but poor coverage. This substantially slows down the process.

We experimented with several different methods for ranking motifs based on different weighting schemes for their sig-scores and coverage percentages. Ultimately, we settled on using coverage as a filter and then ranking motifs based solely on sig-scores. In this approach, a motif must occur at least once in a pCRM in each gene sequence. Motifs that are not able to cover all the sequences from a set of coregulated genes are cast out. The reasoning behind this approach is that, even if a motif represents a valid transcription factor binding site, if the motif is not shared across all genes in the set then it cannot be the mechanism for coregulation in this set. The pCRMs are then ranked by these full-coverage motifs, and those with the lowest motif density are cast out at the end of each round.

Figure 4-5 shows a detailed selection of our results for genes coregulated by Bicoid. In this example, there are a few continuous, if marginal, improvements in the pCRMs' phi-scores (Kruppel and Orthodentical), other improvements are short-lived (Buttonhead and Knirps). The average phi-score does pick up slightly from the fourth round to the fifth, but it fails to regain the score it started with, and the trend is unlikely to

continue because so many of the true-positive pCRMs have already been removed. Note that the Hunchback (hb) plot (blue line) in this figure is spurious; Hunchback contains a binding site for Bicoid in its promoter region but the scores shown in this figure are based only on pCRM-CRM overlaps in enhancer regions.

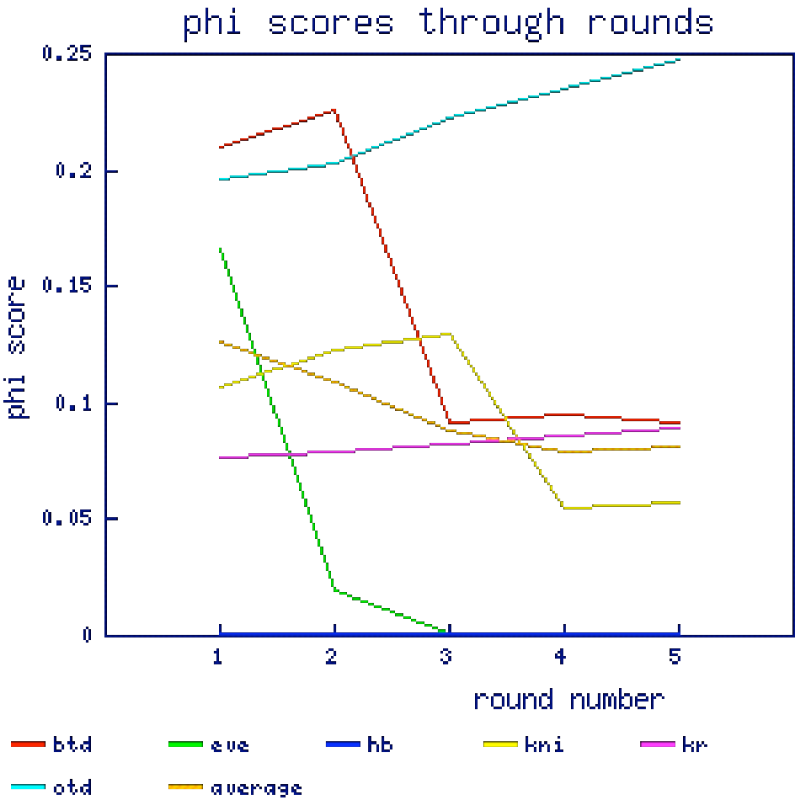

Figure 4-5 The effect of iterative pruning on phi score. The X-axis indicates the iteration number; the Y-axis indicates the phi score.

The trends in this example — a small early improvement followed by a significant drop in scores, and a tendency of some sequence's pCRM scores to improve marginally while the rest drop substantially — are representative of the other coregulated genes in our test set. Although we believe coverage-based motif scoring to be a biologically sound approach to modeling gene coregulation, the results are clearly disappointing. In limited cases we have been able improve on the results from LWF after

overlap-post processing, but an approach that performs consistently and to a statistically significant level has eluded us.

Refinement Testing

In order to assess the efficacy of our refinements, we parsed the five sequences that were set aside from the original 20-sequence dataset with the LWF-overlap filter we developed. Because the shared-motif filtering tools do not consistently improve the results, we did not use them to parse this dataset. The overlap-filtering results are summarized in Figure 4-6, which shows that our refinements do not, on average, improve the results.

This is surprising given that the overlap filter provided nearly across the board improvements when scoring the other 15 sequences in the original dataset. Although it may be that we simply over-trained our algorithm on those sequences, another explanation is that the overlap filter does not perform so well on long sequences as on short sequences, and the sequences in this set are disproportionately long when compared to the cross-validation set. For example, note the scores for Abdominal-A (length: 45,000 bases), Distalless (length: 60,000 bases) and Orthdentical (length: 25,000 bases) when compared with other sequences in Figure 4-2. Additional testing with a dataset containing long and short sequences may help clarify whether the difference in performance is related to overfitting or sequence length.

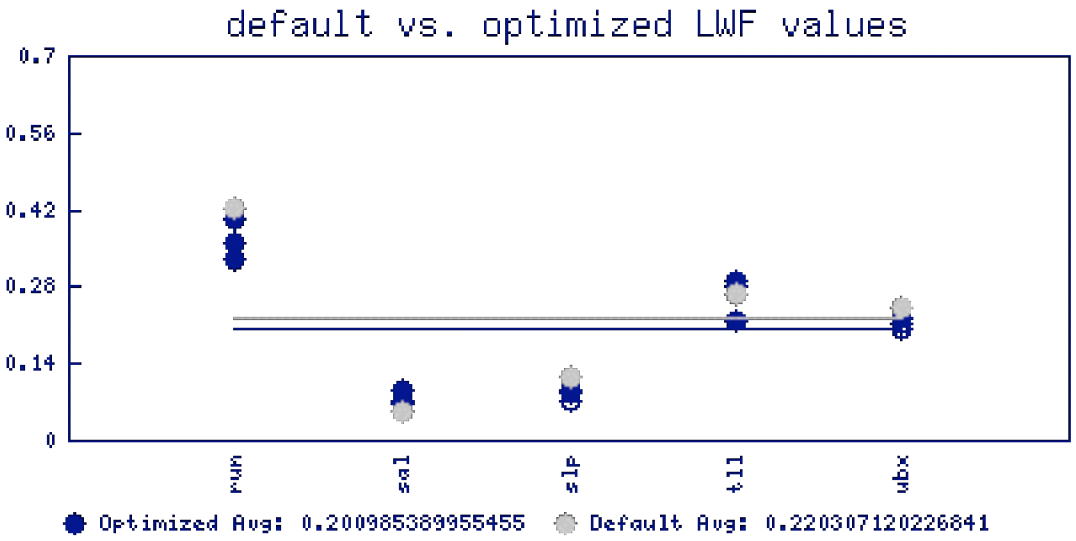

Figure 4-6 Harmonic means for each sequence scanned with the default parameters (gray) and the top three optimized parameter sets (blue). The X-axis shows the different sequences that were scanned; the Y-axis shows the harmonic mean. The gray and blue lines show the average harmonic means for the default and optimized parameters, respectively.

Chapter 5 Further Research

Gene regulation is exceptionally and inherently complex. Here, we have just begun to assess the validity of an approach to genome-wide identification of regulatory binding sites using both statistical models and co-regulatory knowledge. There are a variety of opportunities to explore this process further, many of which evolved directly from these investigations.

Fine-tuning LWF

Here, we explored the parameter space of LWF's training options on a large dataset in order to avoid issues of over-tuning the training parameters to individual positive examples. We have not thoroughly examined variation among the scanning parameters in order to see how they affect the outcome. Just as we explored the parameter space of the training parameters, it would be prudent to sample the space of the scanning parameters by scoring the same sequences with different scanning parameter sets. This section describes the parameters in detail and discusses how we think variation among some parameters might affect the results. In general, we eschew specific recommendations in favor of a broad sampling of the parameter space to uncover the optimal selections.

Scanning Parameter: Profile-cutoff

Sequence windows with a log-odds score above the cutoff are recognized as positive. The profile-cutoff ranges between 0 and 1; smaller values are stricter and result

in a smaller number of sequence windows as pCRMs. The profile-cutoff is essentially a slide, allowing the algorithm to be more sensitive (high cutoff) or more specific (low cutoff). The profile-cutoff is a rough tool. It does not change how the algorithm performs, but rather how its results are interpreted. The default cutoff is 0.1; we worked with a cutoff of 0.3. Although this decreased the ratio of signal to noise, it also allowed more true-positive data to filter through the pipeline to later steps where we applied further winnowing algorithms.

Scanning Parameter: Peak-width-cutoff

During the scanning phase, LWF classifies each position *i* in the input sequence by examining the word distribution in the window beginning at *i*. Consecutive positively-classified positions for windows of potential CRMs all along the input sequence, but positively recognized windows narrower than the peak-width-cutoff are cast out of the pCRM collection. It is difficult to asses how to adjust this parameter because, although very short sequences (50-100 nucleotides) are not likely to have regulatory potential (implying that cutoff value could be relatively high), in many cases we find dense collections of short pCRMs with relatively little space between them. We wonder if, rather than casting out these narrow pCRMs we should instead attempt to join them into a block and treat them as a single unit. This option is discussed further below.

Scanning Parameter: Smoothing-rate (Smoothing Window)

Extreme values within a channel are tempered to prevent a few extreme values from dominating the channel. Because of how the smoothing function is calculated, the rate can vary from 0 (no smoothing) to half the width of the channel (strong smoothing).

It is unclear what kind of direct effect more uneven distributions of words within channels would have. (Again, due to how the smoothing rate is calculated, it can only be decreased from its default, thus leading to channels with more extreme values.) Strong smoothing is a sensible approach because LWF is based on comparing word distributions among different sequences. Without smoothing, it seems unlikely that many windows would contain distributions sufficiently similar to the positive models to be recognized as pCRMs, but it could be that extreme values in some channels are precisely what set regulatory regions apart from the background. Clearly, this is a hypothesis that needs to be tested.

Scanning Parameter: Mismatch-count

Although longer motifs may be more biologically meaningful, shorter words give better statistical resolution. A compromise between the two is to allow mismatches in longer words. This allows LWF to use longer words without defining wide frequency channels with many unique words. This is a key point to investigate and, in retrospect, choosing to disregard it when we sampled the training parameters was an oversight.

In the context of LWF, allowing mismatches when creating training models parallels allowing smoothing during sequence scanning; both are aimed at tempering extreme values in order to compare the core aspects of otherwise disparate models. Without mismatches, word counts vary more widely, lengthening the tails of the channel distribution and flattening its center. When mismatches are permitted, words that vary in only a few positions will be pulled together into single channels, and this may help create CRM and background models that are more distinct from one another.

Scanning Algorithm: Key Framing Between Windows

LWF calculates word distributions for windows beginning at each position in a sequence. This provides exceptional statistical resolution and allows LWF to precisely determine the width of the pCRMs it finds. We wonder, however, if LWF could continue to perform well during the scanning phase while calculating word distributions at fewer positions. Rather than calculating word distributions at every position, we propose a sort of key-framing approach that would calculate distributions at set intervals and then set average scores for the windows beginning at positions between the key frames. Even a small adjustment, such as calculating distributions at every third position, could dramatically reduce the scanning time while still maintaining a high degree of statistical resolution.

Other Approaches to Refining the Results

Joining or Resizing pCRM Blocks

LWF optimizes the lengths of the pCRMs it finds by scoring the windows at each position along the sequence and by dropping blocks that are shorter than the peak-width cutoff parameter. In some cases, we have found this process can generate long stretches of short alternating positively- and negatively-categorized sequences. We have considered several different algorithms for joining these narrowly separated pCRMs. A simple-minded approach would be to pick a maximum span *s* and to join all pCRMs that are separated by *s* or fewer positions. A less arbitrary approach could work by joining pCRMs that are separated by a span narrower than either of the pCRMs to be joined.

The main issue with any approach to resizing pCRM blocks is how to determine whether the resized block is a better candidate pCRM. When working with annotated sequences, it is easy to see where combining, stretching or trimming pCRM blocks will improve their scores, but defining an objective scoring system in the absence of such data is challenging. One solution, rooted in the biology of homotypic regulatory clusters (dense clusters of specific motifs), is to look for multiple copies of a single motif across consecutive pCRMs and to join pCRMs that are sufficiently close and that have high concentrations of the same motifs.

Ranking pCRMs in Coregulated Sequences

Our initial efforts to winnow LWF's pCRMs by searching for shared motifs among coregulated sequences proved frustrating. Nonetheless, we believe the biological reasoning behind this approach to be sound: coregulated genes must use some of the same transcription factors, which means their regulatory sites must share some of the same motifs. It may simply be that we have not filtered out enough background data for motif-finding algorithms to separate signal from noise, or it may be that we need to alter how we are searching for motifs.

For example, we are currently identifying motifs using SCOPE's default scoring function. This function identifies motifs in the input sequence that are statistically overrepresented compared the background sequence of the species they come from. In the specific context of coregulation, however, we are more interested in how pCRMs from different gene compare to each other than in how they compare to the genomic background. Background occurrences are crucially import (it is necessary to distinguish repetition in the genomic background from repetition of a regulatory motif) but a

different scoring function may provide better results. Another approach that would emphasize how the pCRM sequences compare to one another would be to work with tools based on multiple sequence alignments. Again, background levels are an important consideration when examining motifs extracted through an alignment of multiple pCRM sequences, but these approaches could at least play a part in identifying which pCRMs merit further investigation.

Regardless of the particular approach, ranking pCRMs by their content of shared motifs is an avenue of investigation we believe to be worthwhile.

Joining pCRMs from Multiple Tools

LWF is one of a number of algorithms available for identifying pCRMs. We have attempted to keep only a loose coupling between LWF and our other algorithms in order to facilitate the use of different, interchangeable tools in different steps along the pipeline from pCRM identification to overlap detection to the iterative search for motifs in coregulated sequences. It may also be possible to combine the results from multiple pCRM-identifying tools and to use an overlap removal algorithm similar to the one described here (chapter 0). A naïve Bayesian network could also be used to set a confidence level for a pCRM, which could be a very useful step to assist with ranking pCRMs in an iterative approach where poorly ranked sequences are dropped between rounds.

Using Synthetic Data to Determine Baseline Performance

Because of the relative abundance of annotated sequence data for D. Melanogaster developmental genes and their regulatory sequences, we chose to work

only with biological data. This has allowed us to sidestep the issues that surround synthetic datasets, such as whether the search algorithm is just being tuned to the algorithm used to plant binding sites in computationally composed sequences. The task of building a synthetic dataset generator that accurately, or at least adequately, models the genome of a particular species is nontrivial, especially since we don't really have a model for true CRMs. Nonetheless, because a synthetic dataset will contain a set number of known motifs in known locations, this allows for the measurement of an algorithm's baseline of performance. In order to better assess how LWF handles different levels of noise, and to help evaluate our motif-based approach to filtering out pCRMs among coregulated genes, we would be well-served by testing these algorithms under more constrained conditions.

Iterative Background Removal Early in the Pipeline

Our approach to CRM identification has been explicitly two-fold: use an algorithm like LWF to look for pCRMs in individual sequences and then use our knowledge of coregulation to group pCRMs into sets and winnow them further based on the assumption that all sequences in the set must share some common motifs. This approach requires that the first-stage be relatively aggressive in order to prune enough sequence data so that motif-finding algorithms can work effectively in the later, iterative stages. We wonder if a less aggressive, iterative approach would work in the early stages of the pipeline. (This specific approach will not work with LWF because its algorithm is both exhaustive and localized. Removing a segment in the middle will not affect how the edges of the sequence are classified.) An implementation that does not use such localized

models could set a relaxed profile-cutoff and then could gradually, and hopefully more sensitively, remove background regions through successive iterations.

Chapter 6 Implementation Details and Design Decisions

The code for this project was written in Java and Perl. A few small scripts used to automate repetitive tasks were written in sh (Unix shell). As a general rule, Java was used where the problem was well defined or the algorithm was stable and Perl was used where flexibility, rather than speed, was the highest concern.

LWF in Java

Our LWF implementation is written in Java. This offered a roughly two-fold performance improvement over the original Perl implementation. This is a relatively straight port of the original algorithm and as such the structure has not been significantly altered, although in many places it has been refined to better separate the concerns. For example, the classes dealing with user interaction are now separate from those dealing with the core algorithms or data storage.

LWF Analysis in Perl

We chose to write our data analysis scripts in Perl for two reasons. First, its loose typing and flexible data structures allowed us to easily adapt our algorithms as we explored different ways of analyzing the data. Second, several excellent, low-level graphical libraries are available. Although many plotting libraries are also available for Java, we found they struggled to handle data sets with more than a few thousand points.

Our largest plot contains several hundred thousand points, and all contain at least 16,000 (one point for each sequence position).

Several of the LWF analysis scripts make use of the input/output library scan_parse.pm. The main public functions in this library are line_read and line_write. Each takes a line-formatter as an argument; the formatter is then used to format or parse the given data. By separating the data analysis from the data management and allowing them to interact only through this simple interface, both pieces are free to vary independently without affecting each other. As we explored different measures of LWF's performance early in our research, our data fields changed constantly and this approach allowed us to quickly adapt our algorithms without having to retool the entire interface.

Many of our analytical tools make use of the GD::Graph (Verbruggen1999), GD::Graph::boxplot (Wright, 1999) and Statistics::Descriptive (Kuskie, 1998) libraries. All are freely available. GD::Graph provides a set of routines for generating bar, line and scatter plots. GD::Graph::boxplot provides routines for generating box or bar-and-whisker plots. Statistics::Descriptive provides a basic set of functions useful for describing statistical properties of a distribution such as its mean, median and standard deviation.

D. Melanogaster Developmental CRM Database

In order to facilitate our statistical analysis, we created a database containing our test-set sequences and their annotations, including the full sequence, its regulatory and coding regions, and which transcription factors bind in which regions. We then wrote a simple Perl-based API in order to simplify access to the data.

We wrote this application in Perl because we wanted to offer it as a web-based application and the Perl-CGI interface is simple, well established and widely deployed. The data API is implemented in a single data access object (DAO) (nazina_db.pm) and most the web interface is implemented in a CGI script (index.cgi) that calls methods on the DAO to access the database.

Genbank to FASTA Parser

SCOPE uses FASTA files that include only position and sequence data to calculate background occurrences of motifs. In order to extract the background (non-coding) sequences of an organism's genome, we wrote a tool that parses Genbank files (Genbank, 2006), which include fully annotated sequence data, and outputs selected sequences in FASTA format. We wrote this tool in Java in order to take advantage of the substantial sequence-parsing frameworks developed by the BioJava project. Using this framework and a command-line parsing framework from the Apache project, we developed a tool that extracts different kinds of sequences from Genbank files and displays them as FASTA-formatted sequences.

This tool is implemented in two separate modules; one is dedicated to handling user input and parsing the command line and the other deals solely with sequence parsing and formatting. This approach keeps the user interface (command line parsing) separate from the business logic (Genbank file parsing) and allows them to vary independently.

Output is displayed on standard out. This makes it easy to manage output using any IO feature available to the shell running the program, rather than being constrained to the application's own IO features such as writing to a specific file.

Graphics

With the exception of Figure 3-3, all figures were generated by scripts we wrote using the GD::Graph and GD::Graph::boxplot Perl packages. Figure 3-3 was generated using OpenDX.

Chapter 7 Summary and Conclusions

To identify regulatory regions in unannotated sequences of DNA, we implemented and refined an existing algorithm (LWF) and experimented with different methods aimed at further winnowing the results through iteratively searching for shared motifs among coregulated sequences.

The assumption underlying the core algorithm is that word distributions in regulatory sequences are different from those in the genomic background. This content-based approach to analysis is different and more powerful than signal-based approaches that seed the search with examples of known patterns. LWF's scoring profiles also compare favorably with percent-recognition-profiles comparing the two closely related species D. Melanogaster and D. Pseudoobscura. Unlike phylogenetic footprinting approaches, however, LWF does not require the availability of multiple, closely related, fully sequenced species in order to identify potential regulatory sites.

In order to assess LWF's performance throughout its parameter range and on a variety of different sequences, we sampled a broad range of the parameter space and constructed training models from sample sequences in a three-way cross validation test. We found that LWF is highly susceptible to overfitting and that its default parameters do not provide optimal results. After ranking parameter sets by the average harmonic mean of their precision and recall across all fifteen sample sequences, we found relatively little variation among the top-scoring parameter sets. Across many of the top-scoring parameter sets the predicted CRMs that cover true CRMs tend to overlap while predicted

CRMs covering background sequences (false positives) do not. By selecting only overlapping pCRMs generated by the three top-scoring parameter sets, we can consistently and substantially reduce LWF's false-positive rate.

Based on the principle that coregulated sequences must share some common motifs in their regulatory regions, we attempted to further winnow the pCRMs identified by LWF with a motif-based iterative search. We used the motif-finding program SCOPE to extract a ranked list of probable motifs and then filtered out motifs that did not occur at least once in a pCRM for each gene in the set. We then calculated the density of motifs in each pCRM and used this a scoring function by which to rank the pCRMs; the lowest scoring elements were dropped and the process was iterated. Although we believe this to be a sound approach, we were not able to implement it in an algorithm that achieved statistically significant results.

References

Bailey, T. and Elkan, C. (1995). Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization, Machine Learning 21, 51-80.

Berman, B. P. et al. (2002). Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the Drosophila genome, PNAS 99, 757-762.

Berman, B. P. et al. (2004). Computational identification of developmental enhancers: conservation and function of transcription factor binding-site clusters in Drosophila melanogaster and Drosophila pseudoobscura, Genome Biology 5, #9, article R61.

BioJava. (2005). <http://www.biojava.org> (Cited March 2006).

Blanchette, M. and Tompa, M. (2003). FootPrinter: a program designed for phylogenetic footprinting, Nucleic Acids Research 31 #13, 3840-3842.

Bussemaker H. J., Li, H., Siggia, E. D. (2000). Building a dictionary for genomes: Identification of presumptive regulatory sites by statistical analysis, PNAS 97, 10096-10100.

Cáceres, M. et al. (2003). Elevated gene expression levels distinguish human from non-human primate brains, PNAS 100, 13030-13035.

Campbell, N. and Reece, J. (2002). Biology, Sixth Edition, Benjamin Cummings, San Francisco, CA.

Carlson, J. M., Chakravarty, A., and Gross, R. H. (In press, 2006a). BEAM: A beam search algorithm for the identification of cis-regulatory elements in groups of genes. Journal of Computational Biology.

Carlson, J. M., Chakravarty, A., Khetani, R.S. and Gross, R. H. (pending, 2006b). Bounded search for de novo identification of degenerate cis-regulatory elements

Cliften, P. et al. (2003). Finding Functional Features in Saccharomyces Genomes by Phylogenetic Footprinting, Science 301, 71-76.

Durbin, R., Eddy, S., Krogh, A., Mitchison, G. (1998). Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge, UK.

Elnitski, L. et al. (2003). Distinguishing Regulatory DNA From Neutral Sites, Genome Research 13, 64-72.

FASTA. (2005). <http://www.ncbi.nlm.nih.gov/blast/fasta.shtml> (Cited March 2006).

Gasch, Audrey P. and Eisen, Michael B. (2002). Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. Genome Biology 3:11, 1-22.

Genbank. (2006). <http://www.ncbi.nlm.nih.gov/collab/FT/index.html> (Cited March 2006).

Grad, Y., Roth, F. Halfon, M., and Church, G. (2004). Prediction of similarly acting cis-regulatory modules by subsequence profiling and comparative genomics in Drosophila melanogaster and D. pseudoobscura, Bioinformatics 20 #16, 2738-2750.

GuhaThakurta, D. and Stormo, G. (2001). Identifying target sites for cooperatively binding factors, Bioinformatics 17, 608-621.

Gupta, M. and Liu, J. (2005) De novo cis-regulatory module elicitation for eukaryotic genomes, PNAS 102 #20, 7079-7084.

Halfon, M., Grad, Y., Church, G, and Michelson, A. (2002). Computation-Based Discover of Related Transcriptional Regulatory Modules and Motifs Using an Experimentally Validated Combinatorial Model, Genome Research 12, 1019-1028.

Hertz, G. and Stormo, G. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences, Bioinformatics 15, 563-577.

Hudak, J. and McClure, M. A. (1999). A comparative analysis of computational motif-detection methods, Pacific Symposium on Biocomputing 4.

Johnson, D., Zhou, Q., Yagi, K., Satoh, N., Wong, W., and Sidow, A. (2005). De novo discover of a tissue-specific gene regulatory module in a chordate, Genome Research 15, 1315-1324.

Keich, U. and Pevzner, P. A. (2002). Finding motifs in the twilight zone, Bioinformatics 18, 1374-1381.

Kuskie, C. (1998). Statistics::Descriptive. <http://search.cpan.org/~colink/Statistics-Descriptive-2.6/Descriptive.pm> (Cited March 2006).

Lawrence, C. et al. (1993). Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment, Science 262, 208-214.

Lifanov, A., Makeev, V., Nazina, A. and Papatsenko, D. (2003). Homotypic regulatory clusters in Drosophila, Genome Research 13, 579-588.

Markstein, M., Markstein P., Markstein, V. and Levine, M. (2002). Genome-wide analysis of clustered dorsal binding sites identifies putative target genes in the Drosophila embryo, PNAS 99, 763-768.

Mount, D. (2001) Bioinformatics: Sequence and Genome Analysis, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Nazina, A. G. and Papatsenko, D. A. (2003). Statistical extraction of Drosophila cis-regulatory modules using exhaustive assessment of local word frequency, BMC Bioinformatics 4:65, doi:10.1186/1471-2105-3-30.

Pagano, M. and Gauvreau, K. (2000). Principles of Biostatistics, Second Edition. Duxbury, Pacific Grove, CA.

Pevzner, P. A. and Sze, S. (2000). Combinatorial Approaches to Finding Subtle Signals in DNA Sequences, Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology.

Rajewsky, N., Vergassola, M., Gaul, U. and Siggia, E. D. (2002). Computational detection of genomic cis-regulatory modules applied to body patterning in the early Drosophila embryo, BMC Bioinformatics 3:30, http://www.biomedcentral.com/1471-2105/3/30.

Rebeiz, M., Reeves, N., and Posakony, J. (2002). SCORE: A computational approach to the identification of cis-regulatory modules and target genes in whole-genome sequence data, PNAS 99 #15, 9888–9893.

Rigoutsous, I. and Floratos, A. (1997). Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm, Bioinformatics 14, 55-67.

Roth, P. et al. (1998). Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation, Nature Biotechnology 16, 939-945.

Sinha, S. and Tompa, M. (2000). A Statistical Method for Finding Transcription Factor Binding Sites, Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology.

Sinha, S., van Nimwegen, E. and Siggia, E. D. (2003). A probabilistic method to detect regulatory modules, Bioinformatics 19, Supplement 1, i292-i301.

Stormo, G. and Hartzell, G. (1989). Identifying protein-binding sites from unaligned DNA fragments, PNAS 86, 1183-1187.

Stormo, G. (2000). DNA binding sites: representation and discovery, Bioinformatics 16, 16-23.

Thygesen, H and Zwinderman, A. (2005). Modelling the correlation between the activities of adjacent genes in drosophila, Clinical Epidemiology and Biostatistics, BMC Bioinformatics, 6:10, doi:10.1186/1471-2105-6-10

Verbruggen, M. (1999). GD::Graph. <http://search.cpan.org/~mverb/GDGraph-1.43/Graph.pm> (Cited March 2006).

van Helden, J., André, B and Collado-Vides, J. (1998). Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies, Journal of Molecular Biology 281, 827-842.

Wright, N. (1999) GD::Graph::boxplot. <http://search.cpan.org/~gaffer/GDGraph-boxplot-1.00/boxplot.pm> (Cited March 2006).

Zhou, Q., and Wong, W. (2004). CisModule: De novo discovery of cis-regulatory modules by hierarchical mixture modeling, PNAS 101, #33, 12114-12119.

Zhu, Z., Pilpel, Y., and Church, G. (2002). Computational identification of transcription factor binding sites via a transcription-factor-centric clustering (TFCC) algorithm, Journal of Molecular Biology 318, 71-81.