UNIVERSITY OF CALIFORNIA
RIVERSIDE

Gene Function Prediction Based on Sequence or Expression Data

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Kevin Thomas Horan

December 2011

Acknowledgments

Chapter 3 was originally published in Horan et al. (2008), copyright American Society of Plant Biologists; used with permission. Chapter 5 was originally published in Horan et al. (2010), used with permission.

I would like to thank my computer science advisor Dr. Christian Shelton for his invaluable help over the years. I would also like to thank my biology advisor, Dr. Thomas Girke, for providing a good project to work on, and for his guidance and support through all of it.

ABSTRACT OF THE DISSERTATION

Gene Function Prediction Based on Sequence or Expression Data

by

Kevin Thomas Horan

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2011
Dr. Thomas Girke, Co-Chairperson
Dr. Christian Shelton, Co-Chairperson

One of the primary goals of bioinformatics is the identification of the function of genes. The most reliable way of doing this is through experimentation. However, this is a very slow and expensive process. While this is necessary in the beginning and will continue to be necessary for special cases, it becomes impractical when one considers the number of different genes encoded in the genomes of every living organism. A faster way is to instead identify the function of genes by comparing them to the smaller set of genes with known function. This comparison may be based on many different kinds of data, including sequence similarity and gene expression data (Hawkins and Kihara, 2007).

The goal of this dissertation is to provide tools to aid in the identification of the function of unknown genes. To that end, we first present a study in which gene expression data was used to annotate many unknown genes by clustering the expression data. We then present a tool for clustering gene expression data while also identifying short areas of high sequence similarity (motifs) among members of the clusters. Finally, we present a tool for identifying the functionally relevant sub-sections of protein sequences. These sub-sections can then be used to find proteins containing similar sub-sections, even though the rest of the protein may be quite different. This tool can thus find more distantly related proteins sharing functionally relevant features.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Gene function prediction is a primary goal in bioinformatics. As the number of sequenced genomes increases, the need for automated gene function prediction methods grows. Approaches for investigating each individual gene experimentally are simply no longer practical when one considers the number of genes not only in a single organism, but across every sequenced organism. With the rapid increase in the rate of our ability to sequence new genomes, we now have access to the sequence data of a huge number of genes. Automated tools that can make use of this data, along with the existing body of experimentally obtained data, to assign functions to these newly sequenced genes would be a great benefit. Even if these assignments are only educated guesses, it would reduce the amount of manual experimentation by allowing experimenters to focus on testing for the most likely function, rather than starting from scratch for each gene.

It is commonly known that a gene's function is directly related to the three-dimensional structure of the protein it encodes. Since the structure of the protein is closely related to its function, having the structure information greatly increases the chances of correctly predicting the function of a protein, and thus its gene. However, *ab initio* structure prediction of proteins starting from just their sequence is still very hard to do. Because experimental methods are very time consuming and protein structure prediction methods are not accurate enough, we focused in this study on more readily available data types, such as sequence information and genome-wide mRNA profiling data.

1.1 Background

Gene function identification makes use of a wide range of methods, databases, and models to accomplish its goal. In this section the necessary background areas are explained.

1.1.1 Molecular Architecture of DNA

The most basic part of a genome is the DNA molecule which carries the genetic code for each gene in an organism. DNA is a polymer of nucleotides which are composed of phosphate, 2-deoxyribose and one of four possible bases: adenine, thymine, guanine, and cytosine, often abbreviated as “A”, “T”, “G”, and “C”. Usually DNA polymers form double helices, where two DNA polymers are paired according to strict base pairing rules: “A” with “T”, and “G” with “C” and are referred to as base pairs. A DNA molecule is itself coiled into a tight shape which is called a chromosome. Different organisms can have one or more chromosomes. A gene is a region of the DNA molecule which is transcribed to RNA or messenger RNA (mRNA). Most mRNAs are translated to proteins, which are encoded by protein coding genes. Forming a protein from a gene is a complex process. Its most basic steps include: first, the gene is transcribed into mRNA, a molecule similar in structure to DNA. Next, the mRNA is translated into a polymer of amino acids. Each set of 3 base pairs (triplet) in the mRNA codes for one of 20 different amino acids. The amino acid string then folds into a specific three-dimensional structure, called a protein. The sequence and structure of a protein is what determines its function. It may be a catalyst (enzyme) for certain chemical reactions, or it may become part of a physical structure, such as the cell wall. Identifying the function of genes, and hence the proteins they encode, is the primary goal of this work.

1.1.2 Gene Ontology

The Gene Ontology (GO) (Ashburner et al., 2000) system is composed of a dictionary of gene function terms and a graph structure connecting these terms. The purpose of this system is to unify the terminology used to describe gene products in order to simplify computational analyses. The gene ontology system contains three sub-ontologies: cellular component (CC), molecular function (MF), and biological process (BP). The cellular component system describes subcellular locations of proteins, such as organell or protein complex associations. The biological processes ontology describes complex functional events, such as disease resistance or regulatory processes. The molecular function ontology is dedicated to low-level processes that occur on the molecular level, such as specific enzyme activities. The main difference between a biological process and a molecular function is that molecular functions specify individual actions, like transporting something, or binding to something, whereas a biological process is a larger collection of actions occurring in a particular order.

Each sub-ontology is organized in a directed acyclic graph (DAG) and each node in the graph is a term which has a unique identifier. The connections allow terms to be hierarchically structured, but also to be part of several different parent terms. At the root of each DAG are the more general terms that become more and more specific towards the leaves. One or more of terms from each sub-ontology can be used to describe what is known about each gene.

1.1.3 Sequence Similarity

Since sequence similarity among proteins often correlates well with their functional similarities, it is important to know how similar two sequences are to each other. This information can be used to infer function and many other things from their primary sequences. The simplest approach taken in comparing two sequences is to ask how many edits are required to transform one sequence string into another. This can be answered by aligning the two sequences together while allowing gaps and substitutions, and then counting the number of times gaps had to be inserted in one string or the other, as well as the number of symbols in the same position which where different. Gaps in one string are referred to as deletions, while in the other string the same region would be an insertion.

A simple and optimal way to do this is with the global pairwise alignment algorithm from Needleman and Wunsch (1970). This uses a dynamic programming method to compute the scores for every possible subsolution and then connects the best scoring ones to one optimum pairwise alignment. For each possible pair of symbols, a score is defined by an empirical substitution matrix that is given to the algorithm. For example, pairing an “H” with an “H” might have a score of 11, while pairing an “H” with a “C” would have a score of -4. Gaps are also assigned a score, which is usually negative to penalize them. The final score then is the sum of the scores of each pair of symbols along the chosen alignment. Given two strings, x and y, the running time of this algorithm is $O(|x||y|)$, that is, the product of the length of the two strings.

The above algorithm results in a global alignment where two sequences are aligned from their start to their end. Often times one is interested in aligning the most similar sub-regions of two sequences, also known as a local alignments. This can be computed using the Smith-Waterman (Smith and Waterman, 1981) algorithm, which is a variation of the Needleman-Wunsch algorithm. In this case, instead of starting at the lower left corner of the dynamic programming table, the highest scoring cell is identified and then the algorithm backtracks from there with certain boundary rules to find the best scoring local alignment that led to that score. The running time is the same.

Because of the high cost of optimal alignment algorithms, a new algorithm called BLAST (Altschul et al., 1990) was developed which is sub-optimal, but is often several times faster. It uses several heuristics to narrow the search space while still ensuring a high probability of finding a high quality alignment. BLAST has become the default tool for computing sequence similarity. In addition to computing an alignment score, BLAST will also produce an E-value, which is the expected probability of the two given sequences matching with the computed score by chance, given the sequence database.

1.1.4 Gene Expression Data

Gene expression is a measure of how much of a gene product, mRNA and/or protein, is produced from a gene under certain conditions. The most commonly used technologies for measuring gene expression levels on a genome-wide levels are microarrays and high-throughput sequencing approaches, such as RNA-Seq. A microarray is a device containing probes for each gene of interest. Usually, every gene from a particular organism is represented as probes in a highly ordered fashion on the microarray. Each probe is a single or double strand of DNA representing a specific part of a particular gene. The microarray is hybridized with the fluorescently labeled target RNA (or cDNA) of a sample tissue. The targets bind to their complementary probes in a very specific manner. The label bound to each probe is proportional to the mRNA expressed in the sample tissue which can be quantified with a fluorescent scanner.

Microarray experiments are often performed in sets of variable conditions that specific tissues or cell types are exposed to. To allow statistical tests, the biological variance of the samples is estimated by replicating each microarray experiment at least two times. Microarray data is often very noisy, so this is an important first step in computing confidence levels and eliminating noise. By monitoring a set of different conditions for each gene, one can obtain an expression profile for each gene in response to the conditional changes.

1.1.5 Gene Regulation

Regulators of gene expression are usually proteins, such as transcription factors, or short RNA molecules that modulate the transcription of a gene. There are many different mechanisms for accomplishing this, but in this work we are only interested in transcription factors and their binding sites. The latter are short DNA sections (*i.e.* 6-12 base pairs) upstream of a gene. A transcription factor, or set of transcription factors, bind to these sites in order to trigger or prevent the gene from being transcribed. In most cases these binding sites occur around 2000 base pairs upstream of the gene they regulate. This region is called the promoter of a gene. Binding sites are commonly modeled in a similar way as protein families (see Section 1.1.7), using either regular expressions, PSSMs, or HMMs. The latter will be used in this work.

1.1.6 Motif Discovery

Motif discovery is the identification and functional characterization of motifs, or short patterns, in either DNA or protein sequences. In the context of DNA, motifs are usually identified in the promoter region of a gene and indicate the presence of a transcription factor binding site. Genes sharing identical or similar motifs are often co-expressed, and thus are likely to either share a similar function, or take part in related cellular processes. In either case, in this work we will consider such genes as similar and make use of this information during clustering and family assignment. In the context of protein sequence data, a highly conserved motif often maps to the functionally active site of an enzyme or another functionally relevant region of its three-dimensional structure. For example, it may code for the portion of the protein where another protein physically binds to this protein.

Motifs are very good indicators of shared or related function among different genes or proteins because they are often directly involved in their function, and thus cannot tolerate many mutations. When mutations do occur in a motif, the gene may cease to perform its function and would thus no longer be useful. This means that a motif will be highly conserved among genes which share it, even if the rest of the gene has undergone many mutations and thus has a low sequence similarity score on a global level. The term “conserved” has a qualitative meaning indicating an over abundance of something specific, such as a motif, among a set of items. For example, if 90 out of 100 genes contained the exact same motif, one would say that this motif is highly conserved. There is no specific threshold at which point something is said to be conserved however.

Motif discovery is closely related to gene function identification. Given a set of sequences which are known to share a similar function, there are several methods of finding motifs in these sequences which may represent the functionally relevant part of each sequence. Conversely, given a set of known motifs, these can be used to search for other unknown sequences containing the same motif, or motifs, and thus can be used to infer their function.

Most motif discovery methods start with a set of known sequences which share the same function. The easiest way to find motifs from such a set is to align the sequences and look for sub-sections of high conservation. Since motifs are usually highly conserved, there is a good chance that such sub-sections are directly involved in the molecular functions of the corresponding sequences. This requires, however, that a motif occurs in roughly the same location in each sequence, and if there are several motifs, that they share the same spacing in each sequence as well. This is not always the case, and even when it is, generating a multiple alignment can be a hard problem for diverse sequences, often resulting in suboptimal solutions. Another approach is to look for short string sequences that occur more often in this set of sequences than in any other set of sequences from some database. Such a string is said to be over expressed in the given set of sequences. In this case it does not matter where or how many times the string occurs in each sequence.

1.1.7 Family Models

A protein family is a collection of protein sequences which share sequence similarity. New, seed, families are generated by sequence similarity search and clustering approaches as well as manual curation efforts. The latter can be a labor intensive process. Once a seed for a sequence family has been established, the subsequent goal is to identify its most conserved features, and then search for other sequences which share these features. A family model is a particular representation of these conserved features. The three most common models are regular expressions, position specific scoring matrices, and profile hidden Markov models.

1.1.7.1 Regular Expressions

A regular expression is a pattern describing what a string should look like, with some allowable variations. There are several different ways to represent a regular expression; the method used by PROSITE (Hulo et al., 2006, 2008) will be used as an example here, using amino acids abbreviations as the alphabet. The same principals can be applied for any alphabet however. The simplest pattern is just a literal string, such as "K-H-N-N-P". If the first character could also be an "L", for example, the pattern would become "[KL]-H-N-N-P". The square brackets indicate that any of the contained symbols could appear in that location. Sometimes it is shorter to list the set of amino acids that cannot appear in a location. In that case, curly brackets can be used. To indicate that anything can appear, an “x” is used. If the same element of a pattern can occur consecutively several times, that element can be followed with “(i)” or “(i, j)”, which indicates that that element occurs exactly i times, or can occur between i and j times, inclusive. So the first example could also be written as “K-H-N(2)-P”.

While it is possible to create a regular expression automatically from a region of aligned sequences, the resulting pattern often over fits the data. For example:

```
Q4SEE8.1        STVIGKVW.LTVLF..IFR
Q4SIT4.1        -SIFLKVL.LEVGF..MSG
Q4SJY4.1        -----RLWaLQLIF..VTC
Q4SK40.1        STFVGKIW.LTLLI..VFR
Q4T6P3.1        STIVGKIW.LTILF..IFR

PATTERN: S-[ST]-[VIF](2)-[GL]-[KR]-[ILV]-[LW]-x-L-[TQE]-[LV]-[LIG]-[FI]-
         x(2)-[IMV]-[FST]-[RGC]
```

This pattern accurately represents the aligned region, but may capture many unimportant details along with it. The problem is considerably worse in large alignments. As a result, automatically generated patterns can only be used as good starting points for a human, who then modifies the pattern until the desired levels of specificity and sensitivity are reached. This is done by embedding the family members in a larger set of sequences and then using the pattern to identify the sequences which match it. From this the number of false positives (FP), false negatives (FN), true positives (FP), and true negatives (TN) can be counted. The sensitivity can then be computed as $\frac{TP}{TP+FP}$, and the specificity as $\frac{TN}{TN+FP}$. Ideally both of these values should be one, but that is rarely possible to reach in practice.

Because of the requirement for manual tuning and testing, regular expressions are not well suited for large scale analysis. They also are not the most expressive pattern available. For example, in a certain column, 90% of the sequences may contain a “K”, while the remaining 10% contain a “P”. This would result in a pattern like “[KP]” and throws away the fact that “K” is a much better indicator of membership than “P”, even though “P” can still occur sometimes.

1.1.7.2 Position Specific Scoring Matrices

A position specific scoring matrix (PSSM) is a flexible way of storing a string pattern. Rarely is the functional region of a protein always the same sequence of amino acids, or symbols, so searching for exact string matches is not useful. A regular expression is slightly better, but several decisions about what part of the pattern is important must be made in creating a regular expression, and then it cannot say anything about close matches, only whether or not a match was found. A PSSM on the other hand is a statistical model which can be trained on a sample set of strings representing the pattern and then can be used to give the probability that another string matches the pattern.

Given an alphabet Σ and a gapless alignment of length n of strings representing the pattern of interest, a probability distribution, p_i, over Σ is computed for each column of the alignment. The score of a string s of length n is then

$$\sum_{i=0}^{n} \lg p_i(s_i). \tag{1.1}$$

For strings longer than n, a sliding window of length n is used and the highest probability window is considered the best match. Besides their increased flexibility, the main advantage of PSSMs is their simplicity. The main disadvantage however is that they cannot model gaps or insertions, which are quite common in biological sequences. This limits their application to short fragments which can be considered to always appear without gaps or insertions.

1.1.7.3 Profile Hidden Markov Models

A profile hidden Markov model of a sequence family is a statistical model over sequences whose structure consists of a number of states and transitions between states. For each state z there is a distribution, $P(x|z)$ over a set of observations, $x \in \Sigma$. In our case, Σ is the set of amino acids. A transition matrix $T(z_1|z_2)$ defines the probability of transitioning from state z_2 to state z_1. We can view this transition matrix as a graph in which a link exists from z_2 to z_1 if $T(z_1|z_2) > 0$. Figure 1.1 shows the structure used for aligning protein sequences (Durbin, 1998). For each nominal position i there are three possible states: a match state M_i, an insert state I_i, and a delete state D_i. $P(x|M_i)$ is a distribution over amino acids occurring at position i. $P(x|I_i)$ is a background distribution, which is the probability of each amino acid occurring given no other information. This state is used to model noise sections in the input sequence. The delete state does not have a real observation distribution; it requires that nothing be observed (an ϵ observation). This is used to model sections of the input sequence which have been lost.

The transition and emission distribution parameters of an HMM can be learned using the Expectation Maximization (EM) (Dempster et al., 1977) algorithm given a set of observed protein sequences (but not the hidden state sequence), producing a model tuned to this set of protein sequences. Once the model has been trained, we can take

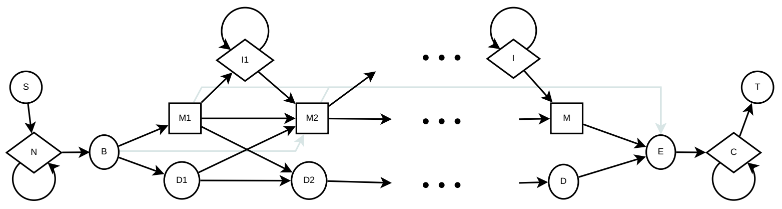


Figure 1.1: The profile HMM of a multiple sequence alignment is illustrated, using the PLAN 7 model in HMMER2. State transitions are illustrated by arrows, match states by squares, insert states by diamonds and delete states by circles.

another protein sequence, S, and ask what is the most likely sequence of HMM states to generate S, and what is the probability of that combination of states and observations. This is done with the Viterbi algorithm (Forney Jr, 1973). To rank the results, it is common to calculate the log-odds

$$\text{score}_{\text{HMM}}(S) = \log \max_{Z} P_{\text{HMM}}(S, Z) - \log P_{\text{back}}(S) \tag{1.2}$$

In this equation, $P_{\text{back}}(S)$ is the probability of S, assuming each amino acid has been drawn independently from the background distribution, while $P_{\text{HMM}}(S, Z)$ is the probability that the HMM would generate the state sequence Z and the observed sequence S. A positive score means that S is more likely to be derived from the HMM than randomly generated from the background distribution. A more detailed description of profile HMMs can be found in (Rabiner, 1990).

1.1.8 Minimum Description Length

The idea behind the minimum description length (MDL) (Grunwald, 2005) principal is to find a model which allows for the highest compression rate of a data set. Such a model must explain the data well so that most of the data can be thrown away given the model, but the model itself must not be too big or it would be better to just keep more of the original data around. Thus, MDL looks for the best and shortest way to describe a data set. In this work we make use of a specific case of MDL, called the Bayesian Information Criterion (BIC) (Schwarz, 1978). This formula balances the likelihood of a model against its complexity. We will use this score to help pick the best model among different candidates later. The likelihood of a model is the probability of a data set under that model. The better a model represents a data set, the higher the likelihood will be, and vice versa. The complexity of a model is the number of degrees of freedom in the model. This is usually the number of free parameters that must be estimated from the data. The BIC score for model M trained on n data points and having N_p free parameters is

$$ll(M) - \frac{1}{2} N_p \lg n \tag{1.3}$$

where ll is the log-likelihood. The goal is to find a model which maximizes this expression. This equation embodies the classic trade-off between overfitting the data and making the model too general. The log-likelihood by itself can be increased all the way to 0 by making the model more complex. In the extreme case, the model simply memorizes the data, giving a log-likelihood of 0. But the complexity term is penalizing the overall score at the same time, so there is no net gain. Conversely, a simple model that does not fit the data well is of no value. The complexity penalty may be small, but the log-likelihood will also be very small, again resulting in no net gain. The best performance is attained by the simplest model that explains the data well. It is also important to note that, for a fixed M, as more data is added the magnitude of the log-likelihood will grow faster than the complexity, so that the influence of the complexity term is less. Thus it naturally prefers simple, more general models when there is less data, but will accept a more complex model if there is enough data to justify it.

1.1.9 Arithmetic Encoding

The complexity term in MDL is basically a measure of how much a model can be compressed. We will use arithmetic encoding to serve this purpose whenever we use MDL in this work. Arithmetic encoding is a method for compressing a string $S = s_1 s_2 \ldots s_n$, where $s_i \in \Sigma$. The basic idea is to use a probability distribution, p, over Σ to decide how many bits to use for the encoding of each symbol. The entire string is then encoded as a single rational number between 0 and 1. For this reason, a fractional number of bits can be used for each symbol, since we never need to represent a single symbol as a discrete sequence of bits. The optimal number of bits to use for symbol $s \in \Sigma$ is $-\lg p(s)$. This assigns fewer bits to symbols with high probability, and more bits to symbols with low probability. The compressed size of S is then $-\sum_{i=1}^{n} \lg p(s_i)$. For the purposes of this work, we do not need to go into the details of actually performing the encoding, we only need to know how many bits a certain symbol sequence can be compressed to.

For example, if we wanted to encode the string "AAAAAABAAAAACC", we would first compute an empirical distribution over the symbols A, B, C, as shown in Table 1.1. A normal block encoding scheme would assign a unique bit string to each symbol. Since we have 3 symbols in this example, each symbol would requires 2 bits to represent it. Thus, with block encoding we need $2 * 14 = 28$ bits. In arithmetic encoding,

Symbol (s)	Count	Probability ($p(s)$)	Bits ($-\lg p(s)$)
A	11	0.78	0.34
B	1	0.07	3.8
C	2	0.14	2.8

Table 1.1: Arithmetic encoding of the string “AAAAAABAAAAACC”

each symbol uses only $-\lg p(s)$ bits, so we need $11 * 0.34 + 1 * 3.8 + 2 * 2.8 = 13.14$ bits.

If the probability distribution over Σ is fixed and known to both the encoder and the decoder, than only the number of bits per symbols must be considered. However, if the symbol distribution is not known before hand, there are two main approaches to sending it to the decoder. The first approach is to scan all the symbols first and compute an empirical distribution, $\hat{p}$ and then use this to encode each symbol. In addition to the number of bits required to send the symbols though, the number of bits needed to send $\hat{p}$ to the decoder must also be included. $\hat{p}$ can be encoded in any way, including arithmetic encoding again, though at some point this recursion must stop. Usually a block encoding will be used, assigning a fixed bit string to each symbol.

The second method is called adaptive arithmetic encoding. In this scheme the encoder starts with a uniform distribution and encodes the first symbol. Then it updates the distribution to include the observed symbol. The second symbol is then encoded according to this modified distribution and so on. By the end of the symbol sequence, we have the empirical distribution again. The advantage of this method is that the encoder does not need to send the empirical distribution to the decoder. The decoder decodes the first symbol according to the uniform distribution, then updates it with the newly decoded symbol and so on, in the same way as the encoder. Thus, each symbol is decoded with the distribution it was encoded with.

Both of these methods increase the average number of bits needed per symbol compared to the optimal case where both the sender and the receiver already share p. Thus, if the receiver does not have p, care must be taken to decide which method will work best, or even if arithmetic encoding should be used at all. In some pathological cases, it can even increase the size of S. If p can be represented compactly, compared to the compressed size of S, then the first method may be best. However, if p is very complex, and there is a lot of data to encode, then the adaptive method may be best. The disadvantage of adaptive encoding is that compression can be very poor near the beginning, before a good estimate of the symbol distribution is obtained. Thus, adaptive encoding is best suited to longer symbol sequences, where it can be assumed that most of the data will be encoded with a good approximation of p.

1.2 Overview

The first study we performed was to identify how many of the currently sequenced genes in *Arabidopsis thaliana* have a known function, either experimentally, or putatively. This was done by classifying each gene as known or unknown according to several different tests, including clustering the sequence data to see which genes did not not match well with anything known, clustering genes by expression profiles, and analysing the GO terms assigned to each gene.

Having established that there are still many unknown genes, even within Arabidopsis, we worked on creating a better clustering algorithm that could incorporate many different kinds of data. As part of this, we examined how much existing motif data could be used to help cluster genes, and also found a way to use the clustering to help find motifs, in an iterative fashion.

Clustering genes by the similarity of their encoding protein sequences is effective in finding similar members, but can easily miss more distantly related or homologous genes. To improve clustering performance in this area, we developed sub-HMMs. These are short sub-regions of an HMM representing a large domain of a protein sequence. Each sub-HMM is taken from an information rich part of the larger domain model, so that together, they retain as much of the most important information as possible, while still greatly simplifying the overall model. We can then look for the occurrences of some or many of these sub-HMMs in other proteins, which may represent a distant relationship between them.

Finally, we found that after extracting sub-HMMs from Arabidopsis we had a large number of sub-HMMs, and many of them where quite similar to each other. Thus, we created a way to cluster these sub-HMMs together to create a smaller set of more general sub-HMMs which could still describe each protein sequence accurately.

Chapter 2

Related Work

Much work has been done in the area of gene function identification. In this work we focus only on using microarray expression data, DNA sequences, and protein sequences. There are many different ways of using this data to assign gene functions, however, we will only discuss two of the most common methods here. In the first method, some or all of the data is used to cluster the genes without regard to which genes are known and which are unknown. Clusters which end up with known genes can be used to label other unknown genes. In the second method, seed clusters are manually created by grouping known genes into families of genes with similar or related functions. Given these seed clusters, the remaining unknown genes are then placed in the most similar cluster. Once a clustering result is obtained, one way to measure the quality of it is to compute a score based on the similarity to functional classifications of genes, such as Gene Ontologies (GO) (Gibbons and Roth, 2002; Ashburner et al., 2000).

2.1 Clustering

The most common way of predicting gene function is to cluster genes together and then determine the clusters function by examining the function of the known genes in the cluster. All unknown genes within each cluster are then assigned the function of the cluster. For each type of data, a distance function is first defined between genes, then any generic clustering method can be used. In the case of sequence data, the BLAST E-value is often used as a distance. For expression data, a correlation coefficient is computed between the vectors of expression levels for each gene. Common coefficients are the Spearman and Pearson correlation coefficients. There are hundreds of studies in which one of these data types have been used to cluster genes. Some examples used for clustering gene expression data are hierarchical clustering (Eisen et al., 1998), K-means (Tavazoie et al., 1999), self organizing maps (SOMs) (Nikkila et al., 2002), and quality threshold clustering (Heyer et al., 1999). Of all these methods, K-means and hierarchical clustering are the most popular; K-means because it is easy to use, and hierarchical clustering because it provides a richer set of information about the clustering.

When using hierarchical clustering, there are three main ways to define a cluster-to-cluster distance. The first is to use the distance between the two nearest points from each cluster (single linkage), second, use the distance between the centers of each cluster (average linkage), and finally, use the distance between the farthest pair of points (complete linkage). Complete linkage is the most conservative since it requires every pair of cluster members to be at least as similar to each other as the worst pair. This prevents transitive relationships from pulling distant sequences together, as can happen when using single or average linkage, which often lead to poor quality clusters (Gibbons and Roth, 2002).

Some work has also been done on using multiple data sources together to perform the clustering. Both Barash and Friedman (2002) and Kundaje et al. (2005) use DNA motifs and expression data simultaneously to cluster genes. Another approach is to build a model which can incorporate several diverse types of data, such as that built by Kasturi and Acharya (2005), which can combine expression data, promoter sequences, known motifs, and gene ontologies; and a probabilistic relational model (Segal et al., 2001; Koller and Pfeffer, 1998).

2.2 Motif Discovery

When combining expression data with DNA sequence data, it is common to look for motifs in the promoter region of each gene. There are two main ways to find these sites. The most popular method is to first cluster the expression data using any clustering method, and then search for common promoter motifs within each cluster, with the assumption that since genes within a cluster share a similar expression pattern, they may also share a common transcription binding site (Gasch and Eisen, 2002; Tavazoie et al., 1999). The second method is to combine the clustering and motif discovery steps into an iterative process, where the current clustering aids in the search for motifs, and the current set of motifs aids in the clustering. This method is used by Holmes and Bruno (2000) as well as Segal et al. (2003).

Motif discovery is a highly active area of research and there are thus many tools and methods for finding motifs, differing by the type of data they work with and the kind of motifs they are looking for, among other things. The major statistical algorithms used by most tools include Gibbs sampling, expectation maximization (EM), and heuristic enumeration of strings.

Some tools that use EM are MEME (Bailey and Elkan, 1995), The Improbizer (Ao et al., 2004), and Cosmo (Bembom et al.). In this method, an unaligned set of sequences and a motif width is given by the user. Every substring of the given width is then used as a pseudo dataset. An PSSM is usually used to model the motif pattern. It is first initialized to some pattern and then scored on each substring. The score of each comparison is the probability that it came from the motif model rather than a background distribution. After many iterations of EM, the motif model will tend to settle on the most over-expressed pattern in the data. Once a motif is discovered, each occurrence of it is statistically erased so that it is not found again. Then the process is repeated to find another motif. LOGOS (Xing et al., 2004) uses a similar method, but makes use of a more complex model which is able to model the locations of motifs within sequences as well and the relationship between different motifs. In this way it is able to capture both the local information which describes the motif pattern, as well as global information, which describes where motifs occur in relation to each other. Local motifs are modeled with HMMs and the whole model is learned with a variant of EM.

Gibbs sampling is another common method and is used by Motif Sampler (Thijs et al., 2001), Bioprospector (Liu et al., 2001), AlignACE (Hughes et al., 2000), and GLAM (Frith et al., 2004), among others. Gibbs sampling is a type of Markov chain Monte Carlo (MCMC), which is a method for sampling from a Markov chain. Each node in the Markov chain is randomly initialized to a value and then one variable, chosen at random, is sampled from, conditioned on the rest. The sample from this one variable combined with the others which are fixed represents one sample from the whole model. This process is then repeated, often many times between actual complete samples. Model parameters are then estimated based on these samples. A PSSM is commonly used to model the motif.

Several other tools are based on enumerating strings in clever ways and checking for over-expression. Weeder (Pavesi et al., 2001) for example, enumerates every string up to a certain length and looks for over-expression of any of these short sequences within the given dataset, compared to their frequency within the entire organism. It also looks for variations of known motifs from yeast. Another tool, called MDscan (Liu et al., 2002b), makes use of ChIP data to help decide which enumerated strings are most likely to be important motifs. Once it finds some candidate strings, it uses them to initialize PSSMs which are then used to find more examples of the motif and to refine the PSSM.

It should be noted that motif discovery is often a goal in itself and is usually done after a clustering has been created. In this work, we are only interested in motif detection insofar as it can be used to show that some genes are similar, in either DNA or protein sequence data.

2.3 Protein Sequence Families

Another approach to gene function identification is to start with a group of functionally characterized proteins and organize them into families which share a similar function. This can be done either through clustering, or by hand, using additional knowledge about the hierarchical structure of the functions of the known genes. The common features of the family members are then extracted and used to search for other genes which share these features. There are many ways to represent the features of a family. Some common representations, or models, are regular expressions, such as those used by PROSITE (Hulo et al., 2006, 2008); Position Specific Scoring Matrices (PSSM), used by PSI-BLAST (Altschul et al., 1997); and profile HMMs, such as those used by SAM (Karplus et al., 1998) and HMMER (Durbin, 1998; Eddy, 1996, 2008). For each of these methods, the model can be compared to a given sequence and produce a score indicating how closely the sequence, or some sub-section of the sequence, matches the model. This score can be used to decide which family a new sequence should be added to.

2.3.1 Fragment Based Models

All the models given above either only model a single important sub-section of each sequence in the family, or else model all sequences from start to end. There are also fragment based models which represent a family as a set of sub-models, each of which only represents some sub-section of each family member. These sub-models can be any of the types listed above, *e.g.*, a PSSM or profile HMM. There are several advantages of this kind of model. First, it is more flexible since the order of the sub-models need not be fixed for all family members. Second, it is more compact since it only models the relevant parts of the family. This leads to fewer parameters and better generalization.

One example of this type of model is presented by Plotz and Fink (Plotz and Fink, 2005). They start with a signal-like protein sequence representation (Plotz and Fink, 2004) for each family. From this representation they extract the most relevant parts as individual HMMs, which they call Sub-Protein Units (SPUs). For each family, a sequence of SPUs in a fixed order is learned which best models that family. Regions not covered by any SPU are modeled with a background sequence. Another related method is Meta-MEME (Grundy et al., 1997). It takes as input a set of motifs discovered by MEME (Bailey and Elkan, 1995), which are represented as PSSMs. It then finds the average locations of these motifs on all of the family members and builds a profile HMM by embedding the PSSMs into the HMM and representing gaps between motifs as insertion loops in the HMM. Since PSSMs do not allow gaps, there is no information about insertion or deletion probabilities for the motif sections of the generated HMM. Therefore, these probabilities are set to 0 and thus the HMM is not as expressive as it could be. The advantage though is that there are fewer parameters to estimate, so Meta-MEME should work better when trained on small families. A similar method developed by Sun and Buhler (2009) attempts to speed up searching with profile HMMs by extracting un-gapped subsections (blocks) of HMMs and then modifying the match distributions in each position to make each block as sensitive as possible. These blocks are then used as pre-filters to eliminate sequences which would not match the whole HMM well. In this case the sub-models are used only to speed up the search, and the full model is still retained to compute the final score.

2.4 Comparison to Current Work

In Chapter 4, we try to cluster genes using expression data while also identifying DNA motifs in an iterative process. The clustering was done with a naive Bayes model incorporating context specific independences (Boutilier et al., 1996). In the work of Holmes and Bruno (2000) they use a probabilistic model for both the expression data and the motifs, whereas in our work we enumerate a small set of candidate motifs and then perform a local search for better motifs by mutating candidate motifs. The method created by Segal et al. (2003) relies on being given a set of fixed candidate motifs created beforehand. These candidates are used to aid in the clustering of the expression data.

In Chapter 5, we create a fragment-based model using profile HMMs as the sub-models. The basis of this work is similar to work done by Plotz and Fink (2005), and Meta-MEME. The goal of Plotz and Fink was to make use of a more detailed signal-like protein sequence representation, as well as to reduce the number of parameters used in order to improve performance on small families. They also enforced a strict ordering of sub-models, which was learned from the data. The work presented here however does not enforce any ordering, and works on more commonly available amino acid sequence data. In contrast to Meta-MEME and BLOCKMAKER, our model uses profile HMMs for the sub-models, which are more expressive than PSSMs since they are capable of modeling gapped regions. The work done by Sun and Buhler (2009) was focused on increasing the search speed of profile HMMs, which is only a minor concern in this work. We also do not retain the full model as they did.

Chapter 3

Large Scale Co-Expression Analysis*

About 40% of the proteins encoded in eukaryotic genomes are proteins of unknown function (PUFs). Only a small percentage of the proteins encoded in animal or plant genomes are sufficiently characterized with regard to their cellular functions. The functions for the majority of these proteins remain either completely unknown (40%) or only partially understood (Gollery et al., 2006, 2007).

Their functional characterization remains one of the main challenges in modern biology. In light of this significant knowledge deficit, our understanding about existing molecular functions appears to be fundamentally incomplete. This is even more evident when we assume that the vast space of unexplored molecular and biological functions is composed of proteins with at least comparable or even greater diversity and importance for cellular processes than the known space. Efforts to narrow this knowledge gap will provide a wide spectrum of opportunities for advancing our understanding about plant and non-plant systems.

In this chapter we identified the PUF encoding genes from Arabidopsis (Arabidopsis thaliana) using a combination of sequence similarity, domain-based, and empirical approaches. Large-scale gene expression analyses of 1,310 publicly available Affymetrix chips was performed to associate the identified PUF genes with regulatory networks and biological processes of known function. To generate quality results, we restricted the data to expression sets with replicated samples. First, genome-wide clustering and gene function enrichment analysis of clusters allowed us to associate 1,541 PUF genes with tightly coexpressed genes for proteins of known function (PKFs). Over

* Originally published in Horan et al. (2008), copyright American Society of Plant Biologists; used with permission.

70% of them could be assigned to more specific biological process annotations than the ones available in the current Gene Ontology release. The most highly overrepresented functional categories in the obtained clusters were ribosome assembly, photosynthesis, and cell wall pathways. Interestingly, the majority of the PUF genes appeared to be controlled by the same regulatory networks as most PKF genes, because clusters enriched in PUF genes were extremely rare. Second, large-scale analysis of differentially expressed genes was applied to identify a comprehensive set of abiotic stress-response genes. This analysis resulted in the identification of 269 PKF and 104 PUF genes that responded to a wide variety of abiotic stresses, whereas 608 PKF and 206 PUF genes responded predominantly to specific stress treatments. The provided coexpression and differentially expressed gene data represent an important resource for guiding future functional characterization experiments of PUF and PKF genes. Finally, the public Plant Gene Expression Database (http://bioweb.ucr.edu/PED) was developed as part of this project to provide efficient access and mining tools for the vast gene expression data of this work.

3.1 Introduction

Two major methods are in use for defining proteins of unknown functions (PUFs) in model organisms. The widely used similarity approach considers all proteins as PUFs that show no detectable sequence or structural similarities to functionally characterized proteins in reference databases (Leinonen et al., 2004; Boeckmann et al., 2003). In contrast to this, the more conservative empirical approach defines as PUFs all proteins that lack direct experimental evidence as support for a specific function. Conceptually, the empirical approach incorporates most PUFs identified by the similarity approach, as well as functionally uncharacterized sequences that share sequence similarities with proteins of known function (PKFs). Sequence families and ortholog (similar genes from different species) clusters are particularly affected by this fundamental difference between the two unknown definitions. For instance, when a group of related sequences contains one or more members of known function, then the similarity approach tends to assign all of them to the known space, whereas the empirical approach distinguishes between functionally characterized and uncharacterized candidates within groups of related sequences. As a result of this difference, most similarity-based PUFs of a given genome are either singletons or members of families that consist exclusively of uncharacterized sequences. These performance characteristics of the similarity concept result in an underestimation of the number of PUFs, because many genes in eukaryotic organisms are members of poorly characterized gene families (Horan et al., 2005). To illustrate this, all members of large families, like protein kinases or cytochrome P450s, will be assigned by the similarity approach to the known protein space, even though most of their members remain functionally uncharacterized (Horan et al., 2005; Nelson et al., 2004; Wang et al., 2003).

Dividing gene products into only two categories of known and unknown sequences is an oversimplification of a complex knowledge system with incremental and multifaceted differences. Consequently, every definition for drawing a strict separation line remains artificial and controversial. While acknowledging these difficulties, this chapter will adopt this two-class system mainly for practical reasons.

To advance our knowledge beyond a roadmap of knowing what we don't know, it is important to develop and apply approaches for predicting putative functions for PUFs. Bioinformatic techniques provide here a wide spectrum of opportunities. For instance, PUFs can be associated with remotely related PKFs by using sensitive sequence and structure similarity search strategies (Altschul et al., 1997; Eddy, 1996). The detected similarities can reveal important clues for testing their functions experimentally. Additionally, one can predict functional features from their sequences, such as sub-cellular targeting signals, secondary structures and membrane domains (Schwacke et al., 2003; Gollery et al., 2006). Proteomics and protein interaction technologies provide additional important functional links (Johnson and Liu, 2006). However, for plants the required proteome resources are not yet available on a genome-wide level. One of the most promising and readily available information resources for systematic functional assignment studies of PUF genes represent large-scale gene expression data from public microarray databases. These data sets offer vast opportunities for associating PUF genes with molecular functions and cellular processes of co-regulated PKF genes.

3.2 Results and Discussion

In this section we identified and analyzed the genome-wide PUF encoding genes from Arabidopsis using both empirical and similarity strategies. Large-scale analysis of publicly available gene expression array data allowed us to associate PUF with PKF genes based on similarities of their expression and treatment response profiles. For this, cluster analysis was used to identify groups of co-regulated PUF and PKF genes based on the similarity of their expression profiles across a wide range of tissue and treatment samples. Subsequently, enrichment analysis of Gene Ontology terms was applied to annotate the obtained clusters by over-represented gene functions. Second, statistical analysis of differentially expressed genes (DEGs) allowed us to identify PUFs that exhibit generic and specific expression changes in response to a large number of different abiotic stress treatments. Finally, the Plant Gene Expression Database was developed to provide to the public efficient data mining utilities for the complex differential expression and clustering data of this project.

3.2.1 Identification of PUFs

To obtain for this study a comprehensive set of PUFs from Arabidopsis, we compared three profoundly different PUF identification methods. The three approaches are based on Gene Ontology annotations, sequence similarities and protein domain searches.

First, we mined the Gene Ontology (GO) annotations to estimate the number of PKFs and PUFs from a manually curated knowledge system that combines empirical and computational methods for assigning gene functions (Berardini et al., 2004; Falcon and Gentleman, 2007). Alternative pathway annotation systems from KEGG and AraCyc could have been used for the same purpose (Kanehisa et al., 2006; Mueller et al., 2003). However, due to the limited number of Arabidopsis genes (<40%) assigned to pathways, the GO system, with close to 95% genome coverage, appears to be currently the more efficient resource for identifying nearly complete PUF sets. This number includes the direct assignments to the root term of each ontology which are the new GO annotations for sequences of unknown function (see Material and Methods for more details).

The evidence codes of the GO annotations specify which functional assignments are supported by experimental evidence data from the public domain and which annotations are solely based on computational prediction methods (Ashburner et al., 2000). To gain insight into the nature of the annotations with regard to the evidence type for assigning members to the known and unknown space, we combined in Table 3.1 the current set of thirteen evidence codes into four custom categories. The category with the highest level of functional support (Empirical) is based on direct evidence from traditional single sample experiments, the second one is based on large-scale screening data (Large-Scale), the third one on computational predictions (Sequence), and the fourth one are the GO-based PUF entries that lack functional support from experiments or *in silico* analyses. The detailed assignment schema of the evidence codes to the four categories is provided in the legend of Table 3.1.

According to the above strategy, 32-38% of the Arabidopsis genes are currently annotated by the GO system as PUF encoding genes (Table 3.1). This is largely in agreement with the estimates from previous studies (Wortman et al., 2003; Gollery et al., 2006). Interestingly, only 7% of all entries are functionally characterized by traditional one-gene-at-a-time experiments in the Molecular Function (MF) ontology and 14% in the Biological Process (BP) ontology, while 34% and 18% have functional support from high-throughput experiments, respectively. This means that 93% of the genes from Arabidopsis code for poorly characterized proteins or PUFs when the most conservative empirical criteria are applied within the MF ontology. The relative amount of PUFs for the combined empirical and large-scale categories is 59% in the MF ontology and 68% in the BP ontology. The Cellular Component (CC) ontology contains by far the largest number of entries with sequence-based annotations and the lowest for the empirical categories. This trend is due to the majority of the CC annotations presently being based on computational *ab initio* predictions of sub-cellular localizations, whereas annotations with experimental support are much less frequent in this category than in the other two ontologies. The subsequent analysis steps of this study utilize the standard PUF set from the MF ontology containing 8,665 members. These genes are exclusively assigned to the root term of the MF ontology (GO:0003674) and they carry the evidence code ND (no biological data available). The MF category was selected here, because protein functions are most profoundly described at the mechanistic molecular level, whereas the other two ontologies, BP and CC, provide rather indirect information in this regard.

To compare the results obtained from the MF ontology with alternative PUF identification methods, we also used one sequence similarity and one domain-based approach using Hidden Markov models. First, all predicted Arabidopsis proteins were searched against the Swiss-Prot database with the BLASTP program (Altschul et al., 1990; Wu et al., 2006). Protein sequences that showed no similarities to functionally characterized proteins in the Swiss-Prot database were classified as PUFs using an expectation value (E-value) of 10^{-6} as cutoff. Second, the same protein set was used to search the Pfam database with the HMMPFAM program (Eddy, 1996; Bateman et al., 2004). Likewise, sequences without similarities to protein domains of known function (E-value $\geq 10^{-2}$) or those matching exclusively domains of unknown function (DUF) were considered PUFs. Due to different calculation methods, the E-values of the two search algorithms are not directly comparable. Therefore, we chose for both methods conservative cutoff values that are commonly used for sensitive sequence similarity searching with low false positive detection rates *e.g.* (Gollery et al., 2006; Horan et al., 2005; Girke et al., 2004). Table 3.2 provides a comparison of the results from the three

	Empirical	**Large-Scale**	**Sequence**	**PUFs**	**Missing**
MF	1,918	9,061	4,677	8,665	2,228
%	7	34	18	33	8
BP	3,731	4,777	4,462	10,194	3,385
%	14	18	17	38	13
CC	3,333	1,661	8,527	8,426	4,602
%	13	6	32	32	17
Any	5,837	11,005	12,851	14,071	0
%	22	42	48	53	0

Table 3.1: The number of protein coding loci from Arabidopsis are given for custom categories of evidence codes of the three gene ontologies: *MF*, Molecular Function, *BP*, Biological Process and *CC*, Cellular Component. A description of the evidence codes is available on the Gene Ontology project site (http://www.geneontology.org/GO.evidence.shtml). The number of loci with annotations in *Any* of the three ontologies are given in the last two rows. The percentage values are calculated relative to the total number of protein coding genes represented in the three ontologies. The evidence codes are grouped into the following custom categories of functional assignments: *Empirical* data (IC, IDA, IGI, IMP, IPI, TAS), *Large-Scale* experiments (IEP, RCA, NAS, NR), *Sequence* similarity or feature predictions (IEA, ISS) and *PUFs* lacking functional data (ND). The column *Missing* accounts for genes that lack annotations within the listed ontologies.

different PUF identification approaches. Based on the chosen confidence thresholds, all three approaches identified PUF sets of comparable sizes with 8,272-8,681 members, while 5,456-6,260 PUFs are common among two and 4,667 among all three methods. To simplify the description of the subsequent functional analysis steps of this study, the remaining text is restricted to the PUF set obtained from the MF ontology. The GO PUF set was given preference, because of the high quality of the manually curated GO annotation system and its broad acceptance in the scientific community.

3.2.2 Relative Amount of Expressed Genes

To functionally associate PUF with PKF encoding genes based on the similarity of their mRNA expression profiles, large-scale gene expression analysis of publicly available Affymetrix GeneChip® microarrays was performed. Only experiment sets containing at least two replicate samples were used for this analysis to enable statistical analysis of differentially expressed genes (DEG) and to increase the confidence of the obtained results. In total, the study included the raw expression data from 1,310 Affymetrix chips from the AtGenExpress and GEO sites (Schmid et al., 2005; Barrett et al., 2006). Table 3.3 provides a summary of the chosen experiment sets that covers a

Method	**SWP**	**Pfam**	**GOMF**
SWP	8,681 (32%)	6,260 (23%)	5,456 (20%)
Pfam		8,272 (31%)	5,788 (21%)
GOMF			8,665 (32%)
All	4,667 (17%)		
Any	12,781 (47%)		

Table 3.2: This table provides a matrix representation of the number of PUFs determined by the three different identification methods: BLASTP searches against the *SWP* database, HMMpfam searches against *Pfam* and the *GOMF* approach from Table I. The amount of PUFs common between pairwise comparisons of methods are provided in the corresponding row and column intersects of the matrix. The numbers of PUFs identified by *All* three methods or by at least one of them (*Any*) are given in the last two rows, respectively. The percentage values are calculated relative to the total number of protein coding genes.

Category	Cel	Samples	Comp	ExpSet
Abiotic Stress	524	254	129	10
Biotic Stress	200	72	55	6
Chemical Treatment	99	46	35	9
Tissue & Development	237	79	40	1
Genotype	86	29	28	4
Hormone Treatment	164	80	46	11
Sum	1310	560	333	41

Table 3.3: This table provides an overview of the different categories of GeneChip® microarray experiments (1st column) that were analyzed in this study. The following numeric columns contain the number of raw data (Cel) files, the amount of the corresponding biosamples (Samples), the number of performed comparisons in the DEG analysis (Comp) and the number of experiment sets (ExpSet) the raw data are derived from.

wide spectrum of treatment series and tissue samples.

The relative amount of expressed genes can be expected to be lower in the PUF than in the PKF category, because many predicted PUF genes may be the result of genome annotation artifacts or may represent untranscribed pseudogenes. In addition, a certain fraction of PUF genes may be expressed below the detection limit of the GeneChip® microarray technology. To estimate the extent of these limitations, the amount of detectable genes across all experiment categories was compared between the PUF and PKF sets. The present call information of the non-parametric Wilcoxon signed rank test of the MAS5 algorithm provides for this purpose relatively reliable estimates (Liu et al., 2002a; Schmid et al., 2005; McClintick and Edenberg, 2006). According to this test, the amount of detectable genes between the PUF and PKF sets differs 0.5-8% within the five frequency intervals plotted in Figure 3.1. Based on these rather small relative differences, it is likely that the majority of the PUF genes are expressed at high enough levels to obtain for them meaningful data in the downstream cluster and differential gene expression analyses of this study.

3.2.3 Cluster Analysis

Since many dynamic cellular processes are tightly associated with coordinated transcriptional changes, cluster analysis of gene expression profiles can be used to identify candidate sets of co-regulated genes that are directly or indirectly involved in related processes (Steinhauser et al., 2004a; Gachon et al., 2005; Toufighi et al., 2005; Haberer et al., 2006; Team, 2008; Jen et al., 2006; Vandepoele et al., 2006; Wei et al., 2006; Gutiérrez et al., 2007). For instance, if a group of genes exhibits correlated expression profiles and it is significantly enriched in genes involved in a specific process then it is reasonable to assume that some of the PUF members of this cluster may share overlapping functions with its functionally characterized members. This association-based approach was applied here on a genome-wide level to systematically assign PUF to PKF genes based on the similarity of their expression profiles. Despite the great potential of this approach, it is important to keep in mind that correlation does not prove causal relationships. It only provides useful leads for establishing hypotheses and causal links in downstream investigations. Accordingly, the results of this study need to be interpreted as preliminary computer predictions that offer useful information for guiding future gene characterization experiments. Final evidence about gene and protein functions cannot be inferred directly from this data. Alternative network modeling approaches were not considered for this study, because of the lack of efficient statistical methods to efficiently represent, score and interpret the resulting network architectures on a genome-wide scale *e.g.* (Wolfe et al., 2005; Gutiérrez et al., 2007; Ma et al., 2007). At this point, the traditional clustering approach appears to be more practical for the goals of this study.

To generate reliable and biologically relevant gene clusters from expression data, we evaluated several available clustering algorithms (*e.g.* K-means, SOM) and selected agglomerative hierarchical clustering as the method of choice (Murtagh, 1985; Eisen et al., 1998; de Hoon et al., 2004; Team, 2008). The hierarchical clustering method was chosen because of three main advantages: (1) the method requires no prior knowledge about the optimum number of the final clusters, (2) it is extremely robust in joining highly similar items into proper similarity groups and (3) it provides an information-

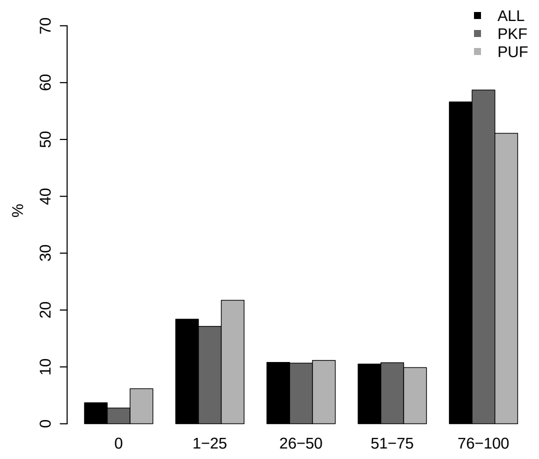


Figure 3.1: The relative amount of present calls is plotted for all genes (ALL), the PKF set and the PUF set using the five frequency intervals (bins): 0, 1-25, 26-50, 51-75 and 76-100% present calls. All experiment sets of this study were used for generating this plot.

rich data output that represents the relative distances between all clustered items in a dendrogram (Becker et al., 1988). The main disadvantages of the approach are the complexity of its data output, the lack of predefined boundaries between clusters and its weaker performance in identifying local expression similarities in a small subset of the samples (Prelić et al., 2006). However, most of these challenges can be overcome by applying efficient post-processing methods of the obtained dendrograms, such as tree cutting methods *e.g.* (Gutiérrez et al., 2007). Popular fuzzy clustering approaches (Krishnapuram et al., 2001) that allow memberships in several clusters - as opposed to strict clustering with unique memberships - were not considered for this study, because of the difficulty to efficiently prioritize and mine the complex cluster memberships from these methods in the downstream functional analysis steps. As an implementation of the hierarchical clustering algorithm, we used the hclust function (Murtagh, 1985) from the statistical programming environment R (Team, 2008). As distance measurement we used correlation coefficients and as cluster joining method complete linkage (see Material and Methods for more details). To obtain discrete clusters from the resulting dendrograms, we developed for this study a novel hierarchical threshold clustering (HTC) method. This method selects clusters in hierarchical clustering dendrograms based on a maximum tolerable distance between cluster members by applying an all-against-all distance test on all possible sub-trees, while maintaining unique cluster memberships. As threshold we chose for this step a minimum correlation coefficient of 0.6. This relatively conservative HTC setting ensures that all members of any given cluster share with all other members of the same cluster correlation coefficients between the selected cutoff of 0.6 and the highest possible value of 1.0. The exact cutoff value of 0.6 was chosen because it resulted in the highest enrichment of functionally related genes compared to alternative cutoffs settings. Additionally, other gene expression correlation studies have used the same or very similar cutoff values (Haberer et al., 2006; Team, 2008; Wei et al., 2006).

Applying the above strategy, we calculated four separate clustering data sets using both the Pearson and Spearman correlation coefficients, in their signed and absolute forms as distance measures. The following text will refer to the four methods as PCC, SCC, PCCa and SCCa, respectively. All four data sets were generated, because of their complementary performance characteristics. The clustering with absolute correlation values allows the identification of positively and negatively correlated gene expressions, whereas the sign-specific approach joins only positively correlated items into similarity groups. The rank-based Spearman approach is limited to identifying global similarities in expression profiles, while the Pearson approach is very sensitive in detecting both, global and local similarities. In particular, the latter detects local similarities with wide amplitude changes relative to the background, which can result in extreme cases in co-clustering of outliers. A consensus approach between several or all methods was not considered, because such a strategy would artificially deflate the cluster sizes and compromise the transparency of the results.

The distributions of the obtained numbers of clusters including their sizes from the four clustering methods are summarized in Figure 3.2. Because the sign removal increases the potential pool sizes of gene pairs with correlation values above a given cutoff, one would expect larger cluster sizes for the data sets with absolute correlation values compared to their signed counterparts. This trend can be observed in the many individual clusters , but the effect is not very pronounced in the global representation of Figure 3.2. These relative increases in cluster sizes are not as frequent as expected, because of two main reasons. First, the number of highly negatively correlated gene pairs is much smaller than the number of positively correlated gene pairs (data not shown, compare (Haberer et al., 2006; Team, 2008)). Second, the assignment of a negatively correlated gene to a cluster at an earlier stage of the hierarchical clustering process can prevent other potential members from joining the same cluster at a given cutoff level, if they do not share the required degree of correlation with the existing members. This is particularly the case in combination with a complete linkage joining method, that was chosen for this study to minimize the number of false positive members in the generated

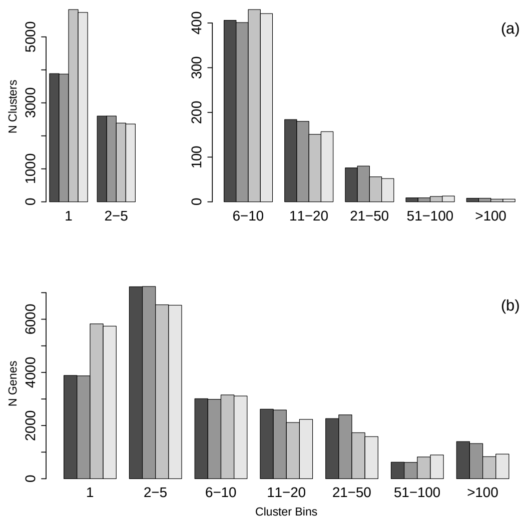


Figure 3.2: The numbers of clusters (a) and genes (b) are plotted for the cluster size intervals (bins) that are given along the abscissa. Each set of four bars, from left to right, contains the data for the clustering results using PCC, absolute PCC, SCC and absolute SCC values as distance measures, respectively.

clusters.

The most obvious differences among the four clustering data sets in Figure 3.2 are the numbers of singlet genes that do not join any clusters in the different methods. There are about 2000 fewer singlet genes in the Pearson than in the Spearman data sets. This is expected because the latter method tends to generate slightly lower correlation values on gene expression data. The subsequent text focuses on the clustering results from the distance method with the signed Pearson correlation coefficients (PCC).

3.2.4 Functional Categorization of Gene Expression Clusters

Gene expression clusters with highly enriched functions provide more conclusive information about the potential roles of their PUF encoding members than clusters with very heterogeneous compositions. To functionally annotate the obtained clusters and to select the most informative gene sets with over-represented gene functions, we performed enrichment analysis of Gene Ontology terms using the hypergeometric distribution as a statistical test (Falcon and Gentleman, 2007). This method computes the enrichment test for all ∼18,000 GO nodes of the three ontology networks and ranks the results by p-values (see Material and Methods). The results of this method are more comprehensive and informative than generalized functional categorization systems, like GO slim or high-level pathway classification systems. Clusters with fewer than 5 members were excluded from this analysis, because the predictive value of extremely small clusters is rather limited. It contains the data for 916 clusters composed of a total of 11,077 genes. To prioritize the clusters based on the obtained enrichment data, we applied two selection filters. First, each cluster of interest needed to contain at least one over-represented GO term in one of three ontologies (enrichment filter). Second, at least 20% of the cluster members had to be associated with this GO term in order to select clusters with relatively homogeneous compositions (uniformity filter). An overview of the number of clusters that meet these filter criteria is provided in Table 3.5. It contains the results for four different p-value cutoffs of the GO term enrichment filter ranging from 0.05 to 10^{-6}. The corresponding GO annotations for the prioritized cluster set, that passed the most stringent selection criteria of 10^{-6}, are listed in Table 3.4. For space and readability reasons, the table presents only the highest ranking GO term for each of the three ontologies. The following discussion of selected clusters is restricted to this most conservative data set (Table 3.4). It contains 66 clusters with a total of 1,279 genes that include 277 PUF genes derived from 53 clusters (see Table 3.5). Our focus on these clusters does not indicate that the other clusters of this study are biologically less important. This selection is mainly based on the assumption that clusters with uniform GO annotations are particularly informative for functionally associating PUF with PKF genes.

Depending on the stringency of the applied prioritization filters listed in Table 3.5, our combined clustering and GO term enrichment strategy associated 277-1,541 PUF genes to overrepresented GO annotations. In comparison to the GO annotations currently available for these PUF genes, our method associated 216-1050 of them to more specific GO terms in the MF category, 225-1089 in the BP category and 239-1096 in the CC category. The large number of PUF genes associated to functionally informative annotations demonstrates the great potential of our approach for guiding future experimental studies on these genes.

Based on enrichment p-values, the most highly over-represented functional categories in the obtained cluster set are the biological processes: ribosome assembly, photosynthesis pathways and cell wall metabolism (Table 3.4). This finding is largely in agreement with related gene co-regulation studies in Arabidopsis (Haberer et al., 2006; Team, 2008; Wei et al., 2006). With regard to ribosome assembly, 124 of the 410 GO annotated genes for cytosolic, plastidial and mitochondrial ribosome components appear in seven clusters (see Table 3.4, cluster IDs: 23, 32, 37, 39, 182, 239 and 299);

and 272 ribosomal genes appear in clusters with ≥5 members of the non-prioritized data set. While cluster 23 consists exclusively of genes annotated as ribosomal genes (GO:0005840, p-value: $1.2 * 10^{-64}$), the other six clusters are highly enriched in ribosomal genes, and they contain among others 16 PUF genes. Equally interesting is the observation that photosynthesis-related annotations are highly over-represented in five large clusters (cluster IDs: 4, 9, 45, 110 and 304). These clusters represent 51 of all the 121 genes that are currently annotated by the GO system as photosynthesis components (GO:0015979). Because both processes, photosynthesis as well as ribosomal activities, require the coordinated assembly of many proteins to large complexes and protein-protein interaction networks, it is not unexpected that their corresponding genes are tightly co-regulated. In alignment with the association hypothesis of this study, several of the PUF members in these functionally extremely uniform clusters may be involved in processes that are connected to the enzymatic or regulatory networks of photosynthesis and ribosomal activities.

CLID	CLSZ	PUF	Sample	P-value	Ont	GO Term
Reproduction						
115	20	5	2	3e-06	BP	GO:0010344: seed oilbody biogenesis
115	20	5	11	0.014	CC	GO:0016020: membrane
115	20	5	4	4.3e-07	MF	GO:0045735: nutrient reservoir activity
Carbohydrate metabolism						
95	21	4	4	5.9e-06	BP	GO:0006073: glucan metabolic process
95	21	4	8	1.8e-10	CC	GO:0005618: cell wall
95	21	4	4	1.9e-07	MF	GO:0005199: structural constituent of cell wall
131	18	1	6	1.1e-10	BP	GO:0006007: glucose catabolic process
131	18	1	7	1.8e-05	CC	GO:0005739: mitochondrion
131	18	1	2	1.3e-05	MF	GO:0004738: pyruvate dehydrogenase activity
248	11	2	3	9.3e-07	BP	GO:0005982: starch metabolic process
248	11	2	9	1.7e-07	CC	GO:0009507: chloroplast
248	11	2	5	0.003	MF	GO:0016740: transferase activity
300	11	3	3	1.7e-08	BP	GO:0005983: starch catabolic process
300	11	3	5	0.011	CC	GO:0044444: cytoplasmic part
300	11	3	2	0.0025	MF	GO:0016758: transferring hexosyl groups
548	7	0	3	2.3e-08	BP	GO:0006084: acetyl-CoA metabolic process
548	7	0	2	1.7e-06	CC	GO:0009346: citrate lyase complex
548	7	0	3	7.3e-09	MF	GO:0046912: transferring acyl groups
686	6	1	3	5.6e-08	BP	GO:0005982: starch metabolic process
686	6	1	2	0.0018	CC	GO:0005829: cytosol
686	6	1	2	3.1e-06	MF	GO:0001871: pattern binding
599	5	0	2	7.3e-06	BP	GO:0016138: glycoside biosynthetic process
599	5	0	2	0.19	CC	GO:0043231: intracellular membrane organelle
599	5	0	3	5e-07	MF	GO:0004497: monooxygenase activity
Nucleotide metabolism						
25	39	10	8	2.6e-07	BP	GO:0006259: DNA metabolic process
25	39	10	3	0.0045	CC	GO:0044427: chromosomal part
25	39	10	2	0.033	MF	GO:0003777: microtubule motor activity
29	37	5	13	2.1e-14	BP	GO:0006259: DNA metabolic process
29	37	5	6	5.3e-07	CC	GO:0005694: chromosome
29	37	5	15	1.2e-06	MF	GO:0003677: DNA binding
41	33	5	4	1.3e-05	BP	GO:0006399: tRNA metabolic process
41	33	5	21	1.8e-15	CC	GO:0009536: plastid
41	33	5	2	0.019	MF	GO:0004812: aminoacyl-tRNA ligase activity
Translation						
23	37	0	36	9.8e-45	BP	GO:0006412: translation
23	37	0	37	1.2e-64	CC	GO:0005840: ribosome
23	37	0	36	5.3e-65	MF	GO:0003735: structural constituent of ribosome

CLID	CLSZ	PUF	Sample	P-value	Ont	GO Term
32	35	3	31	2.7e-35	BP	GO:0006412: translation
32	35	3	33	2e-50	CC	GO:0030529: ribonucleoprotein complex
32	35	3	31	2.4e-52	MF	GO:0003735: structural constituent of ribosome
37	36	11	8	0.00097	BP	GO:0006412: translation
37	36	11	11	5.5e-08	CC	GO:0005739: mitochondrion
37	36	11	4	0.00026	MF	GO:0008135: translation factor activity
39	34	1	29	7.7e-32	BP	GO:0006412: translation
39	34	1	29	4.6e-46	CC	GO:0005840: ribosome
39	34	1	29	3.1e-48	MF	GO:0003735: structural constituent of ribosome
182	11	0	9	4.5e-10	BP	GO:0006412: translation
182	11	0	10	1.8e-16	CC	GO:0005840: ribosome
182	11	0	10	7.8e-18	MF	GO:0003735: structural constituent of ribosome
239	13	0	8	2.5e-08	BP	GO:0006412: translation
239	13	0	6	3.3e-08	CC	GO:0005840: ribosome
239	13	0	6	8.4e-08	MF	GO:0003735: structural constituent of ribosome
299	11	1	7	3.4e-07	BP	GO:0006412: translation
299	11	1	7	2.4e-11	CC	GO:0005840: ribosome
299	11	1	7	2.4e-11	MF	GO:0003735: structural constituent of ribosome
Lipid metabolism						
73	26	8	6	1.8e-14	BP	GO:0019915: sequestering of lipid
73	26	8	7	6.5e-09	CC	GO:0005576: extracellular region
73	26	8	4	1.6e-06	MF	GO:0045735: nutrient reservoir activity
279	10	2	2	9.3e-06	BP	GO:0019374: galactolipid metabolic process
279	10	2	2	0.01	CC	GO:0031967: organelle envelope
279	10	2	5	1.9e-07	MF	GO:0042578: phosphoric ester hydrolase activity
Transport						
47	34	8	2	0.00042	BP	GO:0045036: protein targeting to chloroplast
47	34	8	23	2.2e-17	CC	GO:0009536: plastid
47	34	8	8	1	MF	GO:0003674: molecular function (PUF term)
288	11	4	3	2.4e-07	BP	GO:0045036: protein targeting to chloroplast
288	11	4	4	1.1e-07	CC	GO:0009941: chloroplast envelope
288	11	4	2	0.016	MF	GO:0022804: transmembrane transporter activity
536	7	0	5	9.2e-06	BP	GO:0006810: transport
536	7	0	5	3.6e-09	CC	GO:0005794: Golgi apparatus
536	7	0	4	7.2e-05	MF	GO:0005215: transporter activity
708	6	1	3	4.7e-08	BP	GO:0006606: protein import into nucleus
708	6	1	3	6.4e-07	CC	GO:0005635: nuclear envelope
708	6	1	3	2.3e-06	MF	GO:0008565: protein transporter activity
765	5	1	2	3.2e-05	BP	GO:0006820: anion transport
765	5	1	2	2.1e-05	CC	GO:0005741: mitochondrial outer membrane
765	5	1	2	8.8e-07	MF	GO:0008308: voltage-gated ion channel activity
Biological process						
17	43	26	28	1.1e-08	BP	GO:0008150: biological process (PUF term)
17	43	26	23	1.5e-05	CC	GO:0005575: cellular component (PUF term)
17	43	26	26	2.7e-09	MF	GO:0003674: molecular function (PUF term)
Photosynthesis						
4	134	43	28	2.2e-37	BP	GO:0015979: photosynthesis
4	134	43	67	1.3e-89	CC	GO:0044436: thylakoid part
4	134	43	2	0.00058	MF	GO:0010242: oxygen evolving activity
9	88	18	6	2e-05	BP	GO:0015979: photosynthesis
9	88	18	47	4.2e-30	CC	GO:0009507: chloroplast
9	88	18	2	7e-04	MF	GO:0004045: aminoacyl-tRNA hydrolase activity
45	32	5	7	2.9e-10	BP	GO:0015979: photosynthesis
45	32	5	21	3.1e-16	CC	GO:0009507: chloroplast
45	32	5	5	1	MF	GO:0003674: molecular function (PUF term)
110	20	6	3	8e-05	BP	GO:0015979: photosynthesis
110	20	6	12	9e-10	CC	GO:0009507: chloroplast
110	20	6	2	0.00093	MF	GO:0004176: ATP-dependent peptidase activity
304	9	2	7	1.6e-15	BP	GO:0015979: photosynthesis
304	9	2	5	1.9e-11	CC	GO:0009523: photosystem II
304	9	2	3	5.1e-07	MF	GO:0046906: tetrapyrrole binding
428	8	2	2	0.024	BP	GO:0006091: generation of metabolites and energy
428	8	2	6	5.1e-07	CC	GO:0005739: mitochondrion
428	8	2	2	1.8e-06	MF	GO:0004449: isocitrate dehydrogenase activity
555	5	1	3	1.2e-06	BP	GO:0015979: photosynthesis
555	5	1	3	3.8e-10	CC	GO:0009502: photosynthetic electr. transport chain
555	5	1	3	3.3e-06	MF	GO:0009055: electron carrier activity
923	5	2	3	5.2e-08	BP	GO:0009853: photorespiration

CLID	CLSZ	PUF	Sample	P-value	Ont	GO Term
923	5	2	3	3.9e-08	CC	GO:0030964: NADH dehydrogenase complex
923	5	2	2	0.0047	MF	GO:0003735: structural constituent of ribosome
Cell organization and biogenesis						
77	24	5	6	4.2e-15	BP	GO:0009834: secondary cell wall biogenesis
77	24	5	3	0.011	CC	GO:0031225: anchored to membrane
77	24	5	6	1.7e-05	MF	GO:0016757: transferring glycosyl groups
108	18	7	2	0.00084	BP	GO:0009831: cellulose and pectin modification
108	18	7	18	1.3e-09	CC	GO:0016020: membrane
108	18	7	2	0.0076	MF	GO:0008289: lipid binding
349	9	0	7	1e-15	BP	GO:0009664: cellulose and pectin biogenesis
349	9	0	6	0.00028	CC	GO:0012505: endomembrane system
349	9	0	7	7.6e-19	MF	GO:0005199: structural constituent of cell wall
953	5	1	2	9.7e-07	BP	GO:0010020: chloroplast fission
953	5	1	2	0.029	CC	GO:0009507: chloroplast
953	5	1	4	0.044	MF	GO:0005488: binding
Secondary metabolism						
12	73	13	3	0.0076	BP	GO:0046148: pigment biosynthetic process
12	73	13	34	1.3e-18	CC	GO:0009536: plastid
12	73	13	3	0.00059	MF	GO:0003746: translation elongation factor activity
143	17	2	4	8.7e-08	BP	GO:0046148: pigment biosynthetic process
143	17	2	8	1.4e-05	CC	GO:0009536: plastid
143	17	2	5	0.0023	MF	GO:0016491: oxidoreductase activity
347	10	2	3	1.2e-07	BP	GO:0009686: gibberellin biosynthetic process
347	10	2	4	0.95	CC	GO:0005575: cellular component (PUF term)
347	10	2	5	1.8e-10	MF	GO:0016706: oxidoreductase activity
432	8	1	5	6e-13	BP	GO:0009813: flavonoid biosynthetic process
432	8	1	2	0.00017	CC	GO:0009705: membrane of vacuole
432	8	1	2	0.00023	MF	GO:0016706: oxidoreductase activity
600	5	0	2	9.3e-07	BP	GO:0009718: anthocyanin biosynthetic process
600	5	0	2	0.63	CC	GO:0005575: cellular component (PUF term)
600	5	0	4	0.00049	MF	GO:0016740: transferase activity
Response to stimulus						
68	22	3	7	5.4e-07	BP	GO:0006952: defense response
68	22	3	11	0.02	CC	GO:0016020: membrane
68	22	3	5	1.1e-06	MF	GO:0004888: transmembrane receptor activity
85	23	9	10	6.7e-18	BP	GO:0009408: response to heat
85	23	9	10	0.21	CC	GO:0005575: cellular component (PUF term)
85	23	9	2	0.043	MF	GO:0005516: calmodulin binding
90	22	5	6	1.6e-09	BP	GO:0009408: response to heat
90	22	5	5	0.04	CC	GO:0005634: nucleus
90	22	5	3	0.00052	MF	GO:0051082: unfolded protein binding
346	10	3	2	0.03	BP	GO:0009628: response to abiotic stimulus
346	10	3	9	6.2e-08	CC	GO:0009536: plastid
346	10	3	3	0.57	MF	GO:0003674: molecular function (PUF term)
356	9	9	8	1.1e-15	BP	GO:0009733: response to auxin stimulus
356	9	9	3	0.021	CC	GO:0043231: intracellular membrane-bound organelle
356	9	9	9	7.7e-05	MF	GO:0003674: molecular function (PUF term)
480	8	1	3	3.8e-05	BP	GO:0006979: response to oxidative stress
480	8	1	7	1.1e-08	CC	GO:0005739: mitochondrion
480	8	1	2	7.2e-05	MF	GO:0046933: hydrogen ion transporting ATP synthase
586	7	1	3	4.2e-05	BP	GO:0009737: response to abscisic acid stimulus
586	7	1	2	0.00013	CC	GO:0008287: serine/threonine phosphatase complex
586	7	1	3	5.5e-07	MF	GO:0015071: protein phosphatase type 2C activity
748	5	0	3	6.5e-08	BP	GO:0009404: toxin metabolic process
748	5	0	4	0.01	CC	GO:0005737: cytoplasm
748	5	0	3	6.9e-08	MF	GO:0004364: glutathione transferase activity
912	5	0	4	7.3e-08	BP	GO:0006457: protein folding
912	5	0	3	1.8e-06	CC	GO:0009532: plastid stroma
912	5	0	3	1.2e-06	MF	GO:0051082: unfolded protein binding
Physiological process						
36	34	6	15	0.0011	BP	GO:0043170: macromolecule metabolic process
36	34	6	11	2e-08	CC	GO:0043228: non-membrane-bound organelle
36	34	6	7	3.5e-06	MF	GO:0003735: structural constituent of ribosome
81	24	8	3	0.0011	BP	GO:0051188: cofactor biosynthetic process
81	24	8	14	6.7e-08	CC	GO:0009536: plastid
81	24	8	8	1	MF	GO:0003674: molecular function (PUF term)
130	15	1	3	3.8e-07	BP	GO:0010119: regulation of stomatal movement
130	15	1	2	1	CC	GO:0005575: cellular component (PUF term)

CLID	CLSZ	PUF	Sample	P-value	Ont	GO Term
130	15	1	5	0.041	MF	GO:0016787: hydrolase activity
134	17	7	12	1.1e-20	BP	GO:0006511: ubiquitin-dependent catabolic process
134	17	7	12	1.7e-28	CC	GO:0000502: proteasome complex
134	17	7	7	6.5e-08	MF	GO:0008233: peptidase activity
199	13	1	9	3.9e-07	BP	GO:0009058: biosynthetic process
199	13	1	6	5.6e-12	CC	GO:0044445: cytosolic part
199	13	1	6	2.1e-08	MF	GO:0003735: structural constituent of ribosome
224	12	3	2	0.045	BP	GO:0044249: cellular biosynthetic process
224	12	3	8	1e-07	CC	GO:0009507: chloroplast
224	12	3	2	0.043	MF	GO:0003723: RNA binding
293	11	3	2	0.00012	BP	GO:0042775: ATP synthesis coupled electr. transport
293	11	3	9	3.8e-11	CC	GO:0005739: mitochondrion
293	11	3	3	2.1e-05	MF	GO:0015078: hydrogen ion transmembr. transporter
366	8	1	3	7.9e-06	BP	GO:0006457: protein folding
366	8	1	6	4.5e-10	CC	GO:0005783: endoplasmic reticulum
366	8	1	2	0.0016	MF	GO:0031072: heat shock protein binding
406	9	1	4	1e-06	BP	GO:0006511: ubiquitin-dependent protein catabolic
406	9	1	4	6.2e-09	CC	GO:0000502: proteasome complex
406	9	1	3	0.00063	MF	GO:0008233: peptidase activity
520	7	0	2	3e-06	BP	GO:0006121: mitochondrial electron transport
520	7	0	2	3.4e-06	CC	GO:0045273: respiratory chain complex II
520	7	0	3	4.9e-07	MF	GO:0016627: oxidoreductase for CH-CH groups
728	6	0	6	5.5e-12	CC	GO:0005783: endoplasmic reticulum
728	6	0	2	0.0051	MF	GO:0008233: peptidase activity
790	5	1	3	8.6e-06	BP	GO:0006511: ubiquitin-dependent catabolic process
790	5	1	3	1.1e-08	CC	GO:0005839: proteasome core complex
790	5	1	3	0.00012	MF	GO:0008233: peptidase activity
895	5	0	5	0.0099	BP	GO:0008152: metabolic process
895	5	0	5	2.2e-07	CC	GO:0005739: mitochondrion
895	5	0	2	9.5e-08	MF	GO:0004774: succinate-CoA ligase activity
943	5	2	2	0.029	BP	GO:0009058: biosynthetic process
943	5	2	4	4.7e-07	CC	GO:0005783: endoplasmic reticulum
943	5	2	2	0.44	MF	GO:0003674: molecular function (PUF term)

Table 3.4: The GO annotations for the most conservative cluster prioritization filter from Table 3.5 are provided. The three filtering criteria for selecting the presented clusters are described in the previous legend. Based on space and readability considerations, only the highest ranking GO term within each ontology is included here. As a result of our prioritization criteria, every cluster listed has at least one GO term assigned that meets both, the enrichment (p-value $\leq 10^{-6}$) and uniformity ($\geq$20%) criteria. If an ontology did not contain a GO term passing these filters then the candidate with the lowest p-value was chosen. GO slim terms are used as table subtitles to organize the clusters based on a general biological process classification schema. The different columns provide the identifiers of each cluster (CLID), the number of genes (CLSZ), the number of PUF genes, the number of genes matching a given GO term (Sample), the Bonferroni corrected p-value of the hypergeometric distribution test (P-value), the ontology type (Ont) and the corresponding GO Term, respectively.

Interestingly, our method also identified a cluster (ID 77) that is highly enriched in cell wall-related annotations (e.g. GO:0009834, p-value: $4.2 * 10^{-15}$), such as cellulase synthase genes. A very similar cluster of genes was recently described and experimentally verified by two groups (Persson et al., 2005; Brown et al., 2005) who specifically mined public expression data for genes that are co-regulated with the cellulose synthase genes *CESA4*, *7* and *8*. In addition, comparable results were described by Jen et al. (2006). This example demonstrates that our genome-wide expression clustering approach generates biologically meaningful data. An additional interesting cell wall-related cluster is cluster 349 that contains eight genes for proline-rich extensin domain proteins.

The majority of the clusters in our data set contain one or more PUF genes

Filter	Clusters	Genes
None	916 (794)	11,077 (2,884)
0.05	519 (429)	6,262 (1,541)
0.01	373 (301)	4,893 (1,126)
0.001	212 (170)	3,315 (744)
1e-06	66 (53)	1,279 (277)

Table 3.5: The amount of clusters and genes are provided for different cluster prioritization filters that were applied to the GO term enrichment data. The values in parentheses represent the corresponding number of clusters containing PUF genes and the number of PUF genes in these clusters, respectively. The first row contains the counts for the unfiltered data set that considered only clusters with $\geq$5 members. The subsequent rows refer to the counts after applying the following two-component filter with four different stringency settings. (1) To select clusters with enriched GO terms, the clusters had to contain one or more over-represented GO terms in at least one of the three ontologies based on the Bonferroni corrected p-values of the enrichment analysis. The four different p-value cutoffs used for this filter are given in the first column. (2) In addition, $\geq$20% of the cluster members needed to be associated with the selected GO term in order to favor functionally homogeneous clusters.

(Table 3.5), but only a few of the larger clusters consist predominantly of PUF genes. Cluster 17 represents an exception to this rule. The 43 members of this cluster contain 26 PUF genes, and its characterized members show no clear enrichment of specific functions. Based on the high abundance of PUF genes in the entire data set ($\sim$32%), PUF gene enriched clusters occur much less frequent than those enriched in PKF genes; and clusters consisting exclusively of PUF genes are entirely absent (Table 3.5). One explanation for this difference could be that the expression of most PUF genes is controlled by the same regulatory networks as many PKF genes. If this is the case, PUF genes are more likely to appear in expression clusters together with PKF genes than without them.

Our method also identified clusters that are enriched in abiotic stress response annotations. For instance clusters 85 and 912 are highly enriched in heat stress-related genes (GO:0009408, p-values: $6.7 * 10^{-18}$, $1.8 * 10^{-6}$). Interestingly, 10 of the 23 members in the cluster 85 were identified by the subsequent DEG analysis of this study, as genes that respond specifically to heat stress and to a much lesser extent to other types of abiotic stresses. Based on the available co-expression data, the 9 PUF genes of this cluster are now excellent candidates for discovering novel gene functions involved in heat stress response pathways. Additionally, this example illustrates that the two chosen approaches of this study, expression clustering and DEG analysis, complement and confirm each other. The hypoxia cluster 203 is another interesting abiotic stress cluster (Fukao and Bailey-Serres, 2004). This cluster does not appear in the most stringently prioritized data set (Table 3.4), because it did not pass the applied uniformity filter. Nevertheless, it is enriched in hypoxia-responsive genes (cluster ID 203, GO:0001666, p-value: $2.0 * 10^{-5}$), and it contains several members that are involved in cellular respiration processes, such as genes for the alcohol dehydrogenase ADH1 (AT1G77120), a pyruvate dehydrogenase (AT4G33070) and a hemoglobin-like oxygen binding protein that affects ATP levels under hypoxia (AT2G16060, (Hebelstrup et al., 2007)). Whether the five PUF genes of this cluster are also involved in hypoxia-response processes, can be addressed in experimental studies.

In conclusion, the combined clustering and gene function enrichment strategy allowed us to associate a considerable fraction of the PUF encoding gene pool with tightly co-expressed gene sets of known function. Depending on the chosen stringency settings, the approach allowed us to assign 277-1,541 PUF genes (Table 3.5) to more specific GO terms than those available in the latest GO annotation release for Arabidopsis.

3.2.5 Analysis of Differentially Expressed Genes (DEGs)

DEG analysis can identify groups of genes that exhibit expression changes in response to specific treatments or cellular changes. Because this information is not easily obtainable from clustering of global expression profiles, DEG analysis of publicly available expression data complements the previous approach by associating PUF with PKF encoding genes based on common differential expression responses to environmental changes, such as abiotic stresses. If a group of genes shares similar expression patterns across a wide spectrum of treatments then it is likely that certain members are involved in similar or connected response pathways to these perturbations. The association of genes with these response mechanisms can provide valuable information for future functional characterization experiments of PUF or PKF genes.

One of the main challenges of performing systematic DEG analyses on large and diverse gene expression data sets from public sources is the identification of the given design parameters to determine for each experiment set its biologically most meaningful analysis strategy. This step is extremely crucial, because every analysis needs to focus on the specific treatment factors of an experiment. The alternative of performing simply all possible comparisons will provide meaningless results for many experimental designs, because it would generate a large number of illegitimate contrasts between biologically incomparable samples. In order to define reasonable analysis strategies for public GeneChip® microarray expression data sets, all their replicates and the most useful sample comparisons need to be determined manually to provide the proper experimental design parameters to the downstream statistical methods for identifying DEGs. The MIAME and MGED Ontology annotations (Brazma et al., 2001; Whetzel et al., 2006) of the public microarray depositories provide the essential information about the experiments, but efficient facilities to completely automate the DEG analyses on a large scale are not available at this point.

To perform large-scale DEG analysis of public expression data, we chose for this study a human-supervised analysis strategy, in which we determined for each experiment set its optimum analysis parameters. The goal of this analysis was to identify all PUF and PKF genes that respond to specific or a wide range of conditions by enumerating their significant expression modulations in the corresponding experiment classes. For this, the available experiment annotations were manually evaluated and the most reasonable set of sample comparisons were recorded in an experiment definition table that contained all the required input parameters to control the downstream statistical DEG analysis in an automated manner. Typically, we chose for each experiment set a design strategy that focused the analysis on the primary treatment as the main experimental factor. Multifactorial analysis strategies were avoided as much as possible. For instance, when an experiment contained a stress treatment as the primary experimental factor and time or different tissue types as secondary factors, then we compared only samples from identical tissues that were collected at the same time points. Additionally, comparisons between different experiment sets were not considered to exclude unknown variables, such as sample handling differences between laboratories (Hong et al., 2006). It is important to stress here, that depending on the design of a given experiment and its available annotations, it is often difficult to select a single most meaningful analysis strategy. Thus, our chosen strategy may not provide a perfect solution for every experiment set, but it represents a practical and reasonable compromise for performing systematic DEG analyses on large expression data sets from public databases.

In total our large-scale DEG analysis survey included 333 comparisons between samples with 2-4 technical or biological replicates from 41 experiment sets of 6 experiment categories. Table 3.3 provides an overview of the corresponding sample and experiment sets. Since the abiotic stress category is by far the largest data set, containing 524 chip hybridizations of 254 biosamples (Kilian et al., 2007), the following description of our DEG results will be restricted to this most comprehensive treatment category (Table 3.6). The data for the other categories are provided in the online database of this project (see below). As the statistical method for identifying DEGs with the determined experiment analyses strategies, we used Linear Models for Microarray Data (LIMMA) from (Smyth, 2004, 2005) using in all cases as confidence threshold a false discovery rate (FDR) of ≤ 0.01 in combination with a minimum fold-change filter of 2.

Applying the above DEG analysis strategy, we were able to identify 269 PKF and 104 PUF genes that showed expression changes in the majority of the ten considered abiotic stress categories (Figure 3.3). This set of a total of 373 generic stress DEGs was determined by filtering the generated DEG data set for members that showed one or more significant expression changes in at least 80% of all stress categories. Interestingly, 95% of these DEGs also appear in the generated gene expression clusters of the previous analysis. The subsequent GO term enrichment analysis revealed that stress-related annotations are highly over-represented in this group of DEGs. About 48 of its members (13%) are associated with the GO term "response to stress" from the BP ontology (GO:0006950, p-value: $2.0 * 10^{-13}$). This enrichment indicates that our strategy has a high selectivity for identifying stress-response genes. Therefore, many PUF encoding genes in this data set may be directly or indirectly involved in generic stress response pathways. Among the different groups of identified stress responsive genes (see below and Figure 3.3), the generic stress DEG set represents by far the largest group. Similarly, other studies have shown that stress-regulated genes frequently exhibit expression changes to a wide range of different abiotic stress treatments rather than a refined subset of stresses (Rodriguez and Redman, 2005; Kilian et al., 2007). The group of generic stress DEGs contains 48 genes that are annotated as transcription regulators in the MF ontology (GO:0030528, p-value: $2.5*10^{-3}$). This enrichment emphasizes the central role of transcription factors for the control of many stress response pathways. Moreover, it opens the possibility that several of the 104 PUF genes of this data set may be involved in similar transcription control processes.

We also used the generated abiotic stress DEG data set for identifying genes that respond predominantly to a specific type of stress. These specific stress DEGs were defined as follows. Firstly, they had to show in 25% of all comparisons of a given stress type significant changes. Secondly, they had to exhibit at the same time at least four times as many changes than in the other nine stresses (Figure 3.3). This frequency-based filtering approach appeared to be more efficient for associating DEGs with specific stresses than overly strict filtering methods. This is the case because most stress response genes are not highly specific for a single type of stress (Kilian et al., 2007). As a result, strict filtering for genes responding only to a single stress will fail to identify any candidate genes in our comprehensive data sets. It is important to emphasize here that the chosen filtering approach is a practical compromise, but not a perfect solution to the problem of assigning DEGs reliably to different stress types.

Stress	Chips	Samples	Comp
Heat	68	34	17
Cold	48	24	12
Osmotic	48	24	12
Salt	48	24	12
Drought	56	28	14
Oxidative	48	24	12
Wounding	56	28	14
UV-B	56	28	14
Light	48	16	10
Genotoxic	48	24	12

Table 3.6: The table provides an overview of the different types of abiotic stress experiment sets (Stress) that were used in the DEG analysis of this study. The numeric columns contain the number of the analyzed GeneChip® microarrays (Stress), the number of the corresponding biosamples (Samples) and the number of the performed comparisons (Comp).

With the chosen frequency filter we were able to identify specific stress DEG sets within six of the ten treatment types (Table 3.6, Figure 3.3). The data sets for the stress treatments - light, oxidative and wounding stress - did not contain any genes that meet our filtering criteria, and the drought data set contained only a single member. The lack of specific stress DEGs in these data sets indicates that the genome-wide expression response patterns to these four stresses widely overlap with those from other stresses. For the remaining six treatment categories we identified in total 608 PKF and 206 PUF genes that responded predominantly to single stresses. The functional analysis of these specific stress DEG sets with our GO term enrichment approach showed no outstanding enrichment of specific gene functions. Instead, the results contained mostly moderately enriched GO annotations from a wide spectrum of molecular and biological processes. Similar to the generic stress data, the different groups of specific stress DEGs included various marker genes that are characteristic for stress-related gene sets. For instance, they contained many genes that are annotated with the GO term "response to stress" (see Figure 3.3). This term is significantly enriched in the heat stress data set (p-value: $1.3 * 10^{-2}$), while the other five treatment sets contain it with considerable, but not significantly enriched frequencies (p-values $\geq 5 * 10^{-2}$). In addition, the heat stress and genotoxic stress data sets showed the expected enrichment of genes that are associated with heat response and DNA repair processes, respectively (GO:0009408, p-value: $4.9 * 10^{-3}$ and GO:0006281, p-value: $6.1 * 10^{-5}$).

In summary, the above large-scale DEG study identified a comprehensive set of

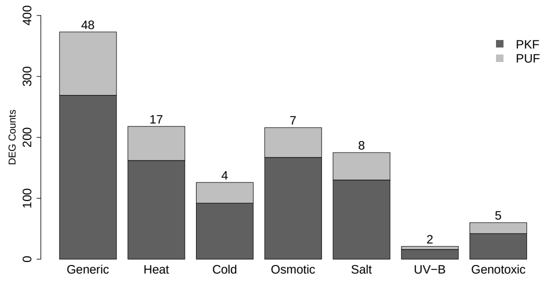


Figure 3.3: The number of PUF and PKF encoding genes are plotted that were identified as generic and specific stress DEGs. The values above the bars provide the corresponding numbers of genes that are currently annotated with the GO term "response to stress" (GO:0006950 in BP ontology). The different stress types are given along the abscissa. Genes responding to the majority of the 10 abiotic stresses were considered as generic stress DEGs (Generic), while those responding predominantly to a specific type of stress were classified as specific stress DEGs. The following filters were used for assigning genes to the two stress categories. (1) Generic stress-response genes are those that showed in at least 80% of all stress treatments one or more significant changes. (2) Whereas, specific stress-response genes are those that showed in $\geq$25% of all comparisons of a given stress significant changes, and exhibited there $\geq$4 times as many changes than in the other nine stresses. For both filters, the observed expression changes were only counted when they meet our confidence criteria of a FDR $\leq$0.01 and a fold change $\geq$2. The specific stress data for the four treatment sets - light, oxidative, drought and wounding - are not plotted here, because their data sets did not contain any DEGs that meet our specific stress criteria.

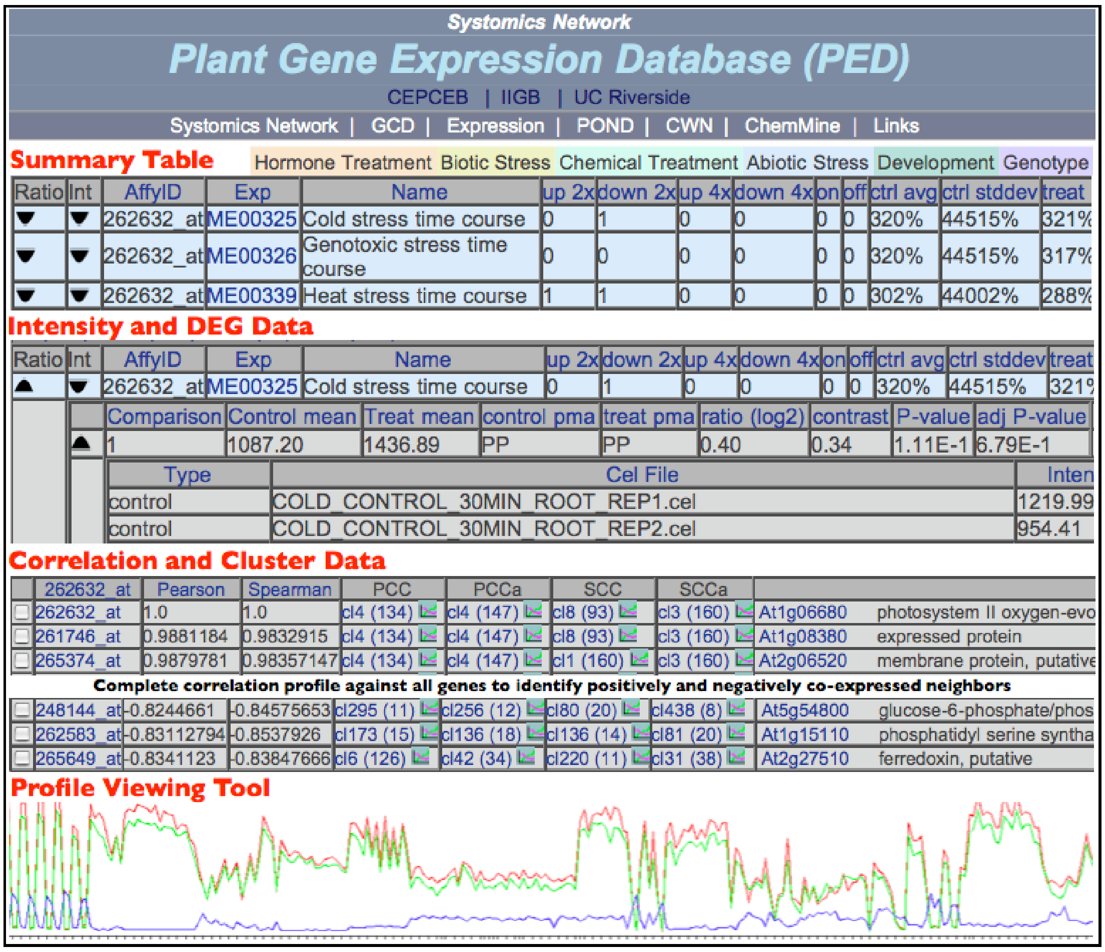
candidate PKF and PUF genes that are involved in generic and specific stress response pathways. These results suggest the existence of one or more abiotic stress response regulons in Arabidopsis similar to the environmental stress regulon (ESR) described in yeast (Gasch et al., 2000; Gasch, 2002). Furthermore, the generated data sets represent an important resource for other scientists, who are interested in addressing more specific questions relevant to abiotic stress research by querying the generated DEG information in alternative ways.

3.2.6 Plant Unknown-eome and Gene Expression Databases

To provide efficient access to the extensive data sets of this study, we have developed two publicly available online portals: the Plant Unknown-eome Database (POND, http://bio web.ucr.edu/scripts/unknownsDisplay.pl) and the Plant Gene Expression Database (PED, http://bioweb.ucr.edu/PED). The POND interface provides query and download options for the latest PUF sets from Arabidopsis. Their predictions are based on the three search methods used for this study: (1) BLASTP searches against the PKFs from Swiss-Prot, (2) HMM searches against the Pfam domain database and (3) retrieval of the 'unknown' annotations from the Gene Ontology system (MF).

The PED integrates our diverse co-expression data with a variety of online tools for user-friendly DEG analysis, cluster visualization and data mining (Figure 3.4). The aim of this service is not to duplicate or compete with the excellent web resources that are already available for array-based expression data from plants, such as GEO, Genevestigator, BAR, AtGenExpress, ATC, PageMan, CSB.DB and MetNet (Barrett et al., 2006; Grennan, 2006; Zimmermann et al., 2004, 2005; Toufighi et al., 2005; Schmid et al., 2005; Jen et al., 2006; Usadel et al., 2006; Steinhauser et al., 2004b; Yang et al., 2005). Instead PED complements the available resources by providing a subset of the publicly available Affymetrix expression data from Arabidopsis in pre-analyzed form using various statistical methods for DEG identification combined with expression cluster information for co-regulation analysis. To provide high-confidence data, the database is restricted to data sets with two or more replicates. The following text provides a brief overview of the most interesting features of the database.

All expression data in PED were normalized with the RMA and MAS5 algorithms (Irizarry et al., 2003; Qin et al., 2006). The incorporation of the expression values from both normalization methods increases the utility spectrum of the provided data sets. The quantile-based RMA method generates more accurate expression measures for weakly expressed genes, whereas the MAS5 scaling approach is more appropriate



Systomics Network

Plant Gene Expression Database (PED)

CEPCEB | IIGB | UC Riverside

Systomics Network | GCD | Expression | POND | CWN | ChemMine | Links

Summary Table

Hormone Treatment | Biotic Stress | Chemical Treatment | Abiotic Stress | Development | Genotype

Ratio	Int	AffyID	Exp	Name	up 2x	down 2x	up 4x	down 4x	on	off	ctrl avg	ctrl stddev	treat
▼	▼	262632_at	ME00325	Cold stress time course	0	1	0	0	0	0	320%	44515%	321%
▼	▼	262632_at	ME00326	Genotoxic stress time course	0	0	0	0	0	0	320%	44515%	317%
▼	▼	262632_at	ME00339	Heat stress time course	1	1	0	0	0	0	302%	44002%	288%

Intensity and DEG Data

Ratio	Int	AffyID	Exp	Name	up 2x	down 2x	up 4x	down 4x	on	off	ctrl avg	ctrl stddev	treat
▲	▼	262632_at	ME00325	Cold stress time course	0	1	0	0	0	0	320%	44515%	321

	Comparison	Control mean	Treat mean	control pma	treat pma	ratio (log2)	contrast	P-value	adj P-value
▲	1	1087.20	1436.89	PP	PP	0.40	0.34	1.11E-1	6.79E-1

Type	Cel File	Inten
control	COLD_CONTROL_30MIN_ROOT_REP1.cel	1219.99
control	COLD_CONTROL_30MIN_ROOT_REP2.cel	954.41

Correlation and Cluster Data

	262632_at	Pearson	Spearman	PCC	PCCa	SCC	SCCa		
☐	262632_at	1.0	1.0	cl4 (134) 📈	cl4 (147) 📈	cl8 (93) 📈	cl3 (160) 📈	At1g06680	photosystem II oxygen-evo
☐	261746_at	0.9881184	0.9832915	cl4 (134) 📈	cl4 (147) 📈	cl8 (93) 📈	cl3 (160) 📈	At1g08380	expressed protein
☐	265374_at	0.9879781	0.98357147	cl4 (134) 📈	cl4 (147) 📈	cl1 (160) 📈	cl3 (160) 📈	At2g06520	membrane protein, putative

Complete correlation profile against all genes to identify positively and negatively co-expressed neighbors

☐	248144_at	-0.8244661	-0.84575653	cl295 (11) 📈	cl256 (12) 📈	cl80 (20) 📈	cl438 (8) 📈	At5g54800	glucose-6-phosphate/phos
☐	262583_at	-0.83112794	-0.8537926	cl173 (15) 📈	cl136 (18) 📈	cl136 (14) 📈	cl81 (20) 📈	At1g15110	phosphatidyl serine syntha
☐	265649_at	-0.8341123	-0.83847666	cl6 (126) 📈	cl42 (34) 📈	cl220 (11) 📈	cl31 (38) 📈	At2g27510	ferredoxin, putative

Profile Viewing Tool

Figure 3.4: The outline illustrates important utilities of the database (URL: http://bioweb.ucr.edu/PED).

for comparisons between expression studies (Lim et al., 2007). The option to identify DEGs by statistical modeling is a very unique feature of this online service. For this, PED provides the results of experiment design-based expression changes from several statistical methods, such as LIMMA (Smyth, 2004, 2005). The corresponding experiment analysis strategies are available for online viewing and download. A combinatorial query page allows searching for DEGs by specific treatments and filtering by various quantitative values to obtain candidate gene lists with strategies that resemble typical microarray analysis routines. Furthermore, the expression intensity and DEG data in PED are fully integrated with a comprehensive set of gene co-expression data from correlation and cluster analyses. To identify for a gene of interest its most positively or negatively co-regulated neighbors, the interface contains a correlation tool that provides for every gene on the arrays the Pearson and Spearman correlation profiles against all other genes. Information on discrete expression clusters is combined with the correlation data. It contains the four separate HTC cluster data sets that were generated by this study using as distance measures the two correlation coefficients in their signed and absolute forms (see previous section). An expression profile plotting tool is available for evaluating the quality of expression clusters or visualizing the expression patterns for custom gene sets across all samples in the database. This utility offers convenient options for inspecting the vast number of expression clusters of this study efficiently. Extensive download options for imports into local spreadsheet programs are available on all query levels for intensity, DEG, correlation and cluster data.

While the backend of the database is based on PostgreSQL and the web interface is implemented in Java, the framework of data analysis and online tools is largely designed around R and BioConductor utilities (Team, 2008; Gentleman et al., 2005). The latter design feature will allow us to routinely add to PED's online services in the future additional useful tools from the wide spectrum of statistical data analysis packages that are provided by the R open source community.

3.3 Conclusion

We present here one of the most comprehensive gene co-regulation studies that are currently available for Arabidopsis. Our study is unique by focusing on the analysis on PUF genes and their systematic association with functional annotations of PKF genes. By applying a combination of genome-wide cluster and DEG analysis methods, we identified many interesting groups of potentially co-regulated genes from a wide range of biological processes and stress response pathways. This approach allowed us to assign

1,541 PUF genes to relative specific and functionally informative GO terms. These gene associations provide a valuable resource for guiding future functional characterization experiments of PUF and PKF genes. In addition, the developed large-scale expression data analysis methods and the associated database represent important components of a future open-source framework for other scientists who are interested in performing similar studies, or utilizing public gene expression resources more efficiently. Finally, users of the provided data sets should keep two limitations in mind. First, the generated associations are hypotheses and not final proofs of gene functions. Second, even the most careful statistical approaches for large-scale data can only reduce, but not fully eliminate errors in the decision making processes associated with the interpretation of microarray data.

3.4 Material and Methods

We now describe the details of each process used in the above discussion, as well as the specific data sets used and where they were obtained from.

3.4.1 Sequence Similarity and Domain Searches

We performed sequence similarity searches of the Arabidopsis proteome against the SwissProt database with the BLASTP program (Altschul et al., 1997) using an E-value of $1 * 10^{-6}$ as cutoff and the default settings for the remaining parameters. The Arabidopsis protein sequences were obtained from the TAIR site (version 7 release, ftp://ftp.arabidopsis.org/home/tair/Sequences) and the SwissProt sequences (Wu et al., 2006) were downloaded from the ExPASy site (release 54.4, ftp://ftp.expasy.org/databases/uniprot). To query only the functionally characterized protein space, we removed all entries annotated as sequences of unknown function from the SwissProt data set.

To identify protein domains of known function in the above Arabidopsis proteins, we performed domain searches against the hidden Markov models of the Pfam database (Bateman et al., 2004) with the HMMPFAM program (Eddy, 1996) using an E value of $1 * 10^{-2}$ as cutoff. The global models of the Pfam release 22 were used for these searches (ftp://ftp.sanger.ac.uk/pub/databases/Pfam/). We ignored matches against domains of unknown function in the post-processing of the search results in order to identify only candidate sequences with domains of known functions.

3.4.2 GO Analysis

The Arabidopsis gene-to-GO mappings from TAIR/TIGR were used for all GO analysis steps of this study. They were downloaded from the Gene Ontology site (10-12-2007 release, http://gene ontology.org). Direct assignments to the root node of each ontology were considered as unknown function annotations. These root assignments, in combination with the evidence code ND (No biological Data available), are the new official GO terms for sequences of unknown function. The former terms, molecular function unknown (GO:0005554), biological process unknown (GO:0000004) and cellular component unknown (GO:0008372), were discontinued by the consortium on 10-17-2006. In the subsequent GO term enrichment analysis steps, the new unknown annotations to the root were considered as artificial terminal annotations. This was necessary, because the root node is connected with all other genes in the GO network, which makes it impossible to obtain for the new unknown annotations meaningful enrichment data with most GO analysis approaches. This modification does not affect the results for any of the other GO nodes.

The hypergeometric distribution was used to test gene sets for the over representation of GO terms. To perform this test, we developed a set of modular functions using the R language for statistical computing for their implementation (Team, 2008). The corresponding GOHyperGAll script computes for a given sample population of genes the enrichment test for all nodes in the GO network, and returns raw and adjusted p-values. As an adjustment method for multiple testing, it uses the Bonferroni method according to Boyle et al. (2004). GOHyperGAll is based on the GOstats package (Falcon and Gentleman, 2007) from the BioConductor project (Gentleman et al., 2005), and it provides similar utilities as the hyperGTest function included in this package. The main differences of our method are that it simplifies the usage of custom gene-to-GO mappings, and it contains various utilities for efficiently analyzing large numbers of gene sets from cluster analyses in batch mode.

3.4.3 Microarray Analysis

A total of 1,310 Affymetrix raw data Cel files were downloaded from the AtGenExpress and GEO sites (Schmid et al., 2005; Barrett et al., 2006; Kilian et al., 2007). All of them are derived from the Affymetrix ATH1 gene GeneChip® microarray for Arabidopsis, and the corresponding samples contained at least two replicate samples. A summary of the utilized experiment sets is provided in Table 3.3. The required probe set-to-locus mappings for the ATH1 chip were obtained from TAIR (ftp://ftp.arabidopsis.org/home/tair/Microarrays/Affymetrix, version 2-5-2007). All ambiguous probe sets on this chip were treated in the gene enumeration steps of this study in the following manner: controls and probe sets matching no or several loci in the Arabidopsis genome were ignored in the downstream analysis steps. In addition, redundant probe sets that represent the same locus several times were counted only once.

The normalization of the raw data Cel files was performed in R using the MAS5 and RMA algorithms, that are implemented in the affy package form the BioConductor project (Irizarry et al., 2003, 2006; Qin et al., 2006). To allow in the DEG analysis comparisons between the different samples of an experiment set, the RMA normalization was performed in batches for entire experiments sets (Table 3.3). This batch normalization is only required for the quantile-based RMA approach, but not for the MAS5 scaling approach. The present call information of the non-parametric Wilcoxon signed rank test was computed with the affy package to estimate the amount of unexpressed genes (Liu et al., 2002a; McClintick and Edenberg, 2006). The obtained expression values from both normalization methods were uploaded to the PED database.

For the DEG analysis, the replicates and the most appropriate sample comparisons were determined manually for each experiment set. The generated analysis strategies were recorded in experiment definition tables. These tables were used to control the downstream DEG analysis steps in an automated manner by providing all information on replicates and sample comparisons to the statistical test methods. The actual analysis of DEGs was performed with the LIMMA package from (Smyth, 2004, 2005). The Benjamini & Hochberg method was selected to adjust p-values for multiple testing and to determine FDRs (Benjamini and Hochberg, 1995). As confidence threshold we used an adjusted p-value of ≤ 0.01 in combination with a minimum fold-change filter of 2. All DEG analyses were performed on both the MAS5 and RMA normalized data sets. While both DEG analysis results were uploaded to the PED database, only the RMA set is discussed in this study, because the RMA algorithm provides more accurate measurements on weaker expressed genes (Qin et al., 2006).

3.4.4 Cluster Analysis

The correlation and cluster analysis steps were performed in R on the MAS5 normalized expression data set. For this, the mean values from replicated biological measurements were combined in one large expression matrix. The RMA data were not used for cluster analysis, because they are less reliable for correlation studies than MAS5 data (Lim et al., 2007). The Pearson and Spearman correlation coefficients were calculated with the cor function in R. The obtained correlation coefficients were transformed into a correlation-based distance matrix after subtracting their values from 1. Four separate distance matrices were calculated for the Pearson and Spearman correlation coefficients in their signed and absolute forms. The matrices were passed on to the hclust function (Murtagh, 1985; Team, 2008) that performs agglomerative hierarchical clustering. Complete linkage was used as cluster joining method.

In order to obtain from hierarchical dendrograms discrete clusters, we developed a new hierarchical threshold clustering (HTC) method for this project. This method identifies sub-clusters in dendrograms based on a minimum tolerable similarity cutoff between all cluster members. This is achieved by applying an all-against-all similarity test for the clusters from all possible sub-trees. At the same time, unique cluster memberships are maintained and all items in the processed dendrogram are assigned to clusters with one or more members. As cutoff we used for this cluster selection procedure a correlation coefficient of $\geq$0.6. This cutoff was chosen because it resulted in the highest enrichment of functionally related genes compared to alternative cutoffs settings. As a result of this method, the members of every identified cluster shared with all other members of the same cluster correlation coefficients between 0.6 and 1.0.

Chapter 4

Gene Network Analysis Using Bayesian Clustering

The goal of this project is to identify gene regulatory networks in Arabidopsis by integrating different kinds of genomic data, such as expression and regulator data. In addition, we also try to identify the function of previously unknown genes, and find novel regulators. Our dataset consists of 26 thousand genes, for each gene we use 571 microarray experiments, and the promoter sequence data for each gene.

We take a Bayesian approach to this problem primarily because the data is very noisy and Bayesian methods have been shown to be effective in modeling this noise (Friedman et al., 2000). A secondary reason is that Bayesian models are very flexible, which allows us to easily add additional components to the model that represent different kinds of data.

We started with a Naive Bayes model, which is a simple way to model a clustering problem. In this case we are clustering the genes of Arabidopsis. The model has one hidden cluster variable and several independent (given the cluster) child variables. The child variables represent different features of the data, in this case we have child variables for each experiment, and we test two different methods of representing the promoter. Since the cluster variable is hidden, we use Expectation Maximization (EM) to learn its parameters.

In addition to this model, we also use Context Specific Independence (CSI) (Barash and Friedman, 2002) to better model the data. It is often the case that some features are only relevant to some clusters, and completely useless to other clusters. With CSI, we can find out which clusters depend on which features, then the useless clusters can be summarized with a single distribution. CSI both reduces the complexity of the model, and provides us with more information about the data. In order to learn this mapping, we use Structural Expectation Maximization (SEM).

4.1 Introduction

A gene network is a directed graph indicating which genes regulate which products. A product can be either something that promotes or inhibits the transcription of another gene, or it can be an end product itself, commonly a protein. An important first step in constructing this graph is identifying the regulators of each gene. The most common way of doing this is to start with a set of expression data from experiments done under different conditions and then look for groups of genes that were expressed in similar ways. Such groups are likely to be regulated by the same regulator. Other types of data can also be used to group genes together, such as GO identifiers and previously identified regulators. Ideally, a clustering algorithm should be able to make use of all of these types of data.

Once a clustering of genes is found one can search for common patterns in each cluster, which are good candidates for regulators. Additionally, if some of the genes in a cluster have a previously identified function we can predict that other cluster members share the same function.

When combining different kinds of data, and even between different experiments in the expression data, there are cases where some features will be irrelevant to the correct grouping of certain genes. For example, two genes may be very similar in all but one feature, but that feature may have nothing to do with what is truly common between the genes, so it is acceptable for it to be different. Most standard clustering algorithms, such as K-means or hierarchical clustering, do not deal with this case very well. In the above example, that pair would be penalized for differing in that one variable, when it should not be penalized. The algorithm presented here is able to make use of a wide variety of data types as well as identify which variables are relevant to each cluster, thus avoiding this problem.

4.2 Method

In this work we present two stages. In the first stage we implement a Naive Bayes clustering algorithm with CSI. In this stage we make use of expression data and previously identified regulators to cluster genes. In the second stage, we extend and modify this algorithm to make use of expression data and the raw promoter sequence of

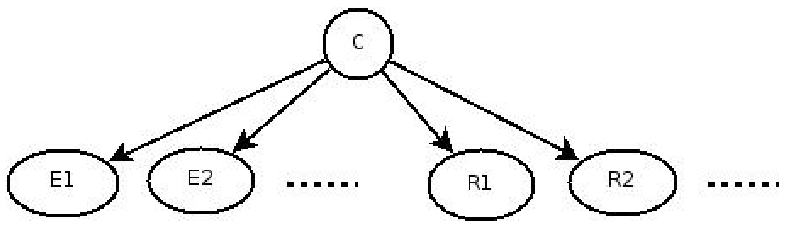


Figure 4.1: Naive Bayes clustering model with expression variables $E_1, \ldots, E_n$, and regulator count variables $R_1, \ldots, R_m$.

each gene. In addition to clustering the genes, the algorithm identifies a set of regulators for each cluster. Regulator identification and clustering are then repeated in an iterative fashion, each helping the other to find a better solution.

4.2.1 Clustering

We start with a simple naive Bayes model for clustering. We have a cluster variable, C, and feature variables, each dependant on C. In stage one we have two kinds of features, real valued expression level variables, $E_i, i \in [1, n]$, and discrete valued variables, $R_j, j \in [1, m]$, for the counts of each regulator in each gene. Each of these features are conditioned on the value of C. This model also assumes that each data variable is independent of all other data variables, given the cluster label. While this is not guaranteed to be true in practice, it is often a close approximation and keeps things simple and fast. The whole data set is denoted by D, and the mth row by $D[m]$. A specific value of a variable, such as r_1 in the mth row, will be denoted as $r_1[m]$.

C is modeled as a multinomial with parameters $\theta_1, \cdots, \theta_k$. Here k is the number of clusters, which must be set by hand. $P(E_i|C) \sim N(\mu_{E_i}, \sigma_{E_i})$ and $P(R_j|C) \sim \text{Multinomial}(\theta^1_{R_j}, \cdots, \theta^q_{R_j})$, where q is 4, each value representing either 0, 1, 2, or "more than two". All these variable are observed, except for C, which is a hidden variable and represents the cluster label. Expectation maximization (EM) is used to learn the parameters of C. Given an initial set of parameters though, we can define a clustering by assigning each sample to the most likely cluster:

$$\text{cluster(sample)} = \arg\max_{c \in C} P(c|e_1, \cdots, e_n, r_1, \cdots, r_m) \tag{4.1}$$

The goal of EM is to find a set of parameters for the cluster variable which maximizes the likelihood of the given dataset. This is done by iterating two steps. We start with a set of random parameter values for C and then compute the expected sufficient statistics as follows:

$$\bar{M}[c] = \sum_m P(c[m]|D[m]) \tag{4.2}$$

$$\bar{M}_e[e_i, c] = \sum_m e_i[m]P(c[m]|D[m]) \tag{4.3}$$

$$\bar{M}_{e^2}[e_i, c] = \sum_m e_i[m]^2 P(c[m]|D[m]) \tag{4.4}$$

$$\bar{M}[r_j, c] = \sum_m r_j[m]P(c[m]|D[m]). \tag{4.5}$$

A sufficient statistic is the smallest set of numbers needed to compute the parameters of a distribution. For example, a Gaussian distribution has two parameters, the mean and the variance. We can compute both of these values knowing just the number of data points, the sum of the values, and the sum of the squares of each value. Given these three values, we no longer need the original data to compute the parameters of a Gaussian. Since we don't actually have all the data in this case, we compute the expected value of the sufficient statistics, based on our current estimate of the distribution parameters. Given the sufficient statistics, we can compute the optimal parameters, in terms of likelihood, as follows:

$$\hat{\theta}_c = \frac{\bar{M}[c]}{\bar{M}} \tag{4.6}$$

$$\hat{\mu}_{e_i,c} = \frac{\bar{M}_e[e_i, c]}{\bar{M}[c]} \tag{4.7}$$

$$\hat{\sigma}_{e_i,c} = \frac{\bar{M}_{e^2}[e_i, c]}{\bar{M}[c]} - \hat{\mu}^2_{e_i,c} \tag{4.8}$$

$$\hat{\theta}^q_{r_j,c} = \frac{\bar{M}[r_j, c]}{\bar{M}[c]}. \tag{4.9}$$

Here $\hat{M}$ is the total number of pseudo counts, $\sum_c \hat{M}[c]$. Each iteration of these steps is guaranteed to not decrease the likelihood.

We also make use of Context Specific Independence (CSI) in this model. Without CSI, each data variable must store a distribution for each possible cluster label. However, it may be that a variable is really only relevant, or distinct, for some of the clusters, while its behavior is identical for all other clusters. Then it makes sense to only store distributions for the clusters for which the variable has distinct differences, and summarize all other clusters with a single distribution (Barash and Friedman, 2002). A second advantage of CSI is that it reduces the number of parameters that need to be learned, which leads to better generalization of the model. For example, if we had a regulator variable, R, and it usually only appears in sequences in clusters 1 and 3, we

	q=0	q=1	q=2	q=3
C=1	0.1	0.1	0.5	0.3
C=3	0.1	0.1	0.2	0.6
C=$*$	0.7	0.2	0.05	0.05

Table 4.1: Example Context Specific Independence (CSI) of a single regulator variable. Here only clusters 1 and 3 are relevant to this variable, while all other clusters are summarized by the $C = *$ distribution.

might a have set of distributions that look like Table 4.1, where $C = *$ is called the default cluster and is used as the distribution for every cluster label besides 1 and 3.

Which clusters to keep distributions for must itself be learned, for each variable. Because each variable in independent given the cluster, we only need to consider one variable at a time which searching for the best structure. The structure here is a binary string for each variable, with bit for each cluster label. A 1 bit signifies that a specific cluster distribution is stored for that cluster, while a 0 bit indicates that the default cluster should be used for that cluster. We thus have an exponential number of structures to search through, for each variable. Since the number of clusters can be large, an exhaustive search through this space is intractable. To solve this problem we perform a local search through the space of structures. To do this we need a score with which we can compare structures. Using the likelihood itself is not sufficient since that will always lead to a bit string with all ones. The goal of CSI is to simplify the structure by removing unnecessary distributions. Thus, we need to trade off the performance, measured by the likelihood, with the complexity, measured by the number of parameters needed.

We used the Bayesian Information Criterion (Schwarz, 1978), or BIC, score for this purpose, which is defined as

$$\text{Score}_{\text{BIC}} = \ln P(D|\mathcal{M}, \hat{\Theta}_{\mathcal{M}}) - \frac{1}{2}\ln(M)\text{Dim}(\mathcal{M}). \tag{4.10}$$

This is the difference in the log-likelihood of the data given the current model $\mathcal{M}$ and the current parameter estimate for that model $\hat{\Theta}_{\mathcal{M}}$, and the complexity of $\mathcal{M}$. The complexity is weighted by the log of the number of data points, M, used to learn the model. As the amount of data grows, the complexity term will grow more slowly than the log-likelihood, and so will exert less influence on the final score. In this way, more complex models will be accepted if there is enough data to justify the extra complexity. But where there is only a small amount of data available, the complexity term encourages simpler, more general models.

Using this score, we can search for better structures. We initialize each variable with a random structure and then run EM as described above. Then, for each variable, we perform a greedy search for a better structure. We first compute the BIC score of the current variable, and then try flipping each bit to see if the score improves. If a change is found to improve the BIC score for a variable, EM is run again for just that variable to re-estimate the parameters. The greedy search is then resumed. This process is iterated until no improvement in the structure is found. This is known as structural EM, or SEM.

The final likelihood expression of this model with CSI is

$$P(D|\mathcal{M},\hat{\Theta}_{\mathcal{M}}) = \left[\prod_m P\left(C = c[m]|\mathcal{M},\hat{\Theta}_{\mathcal{M}}\right)\right] \cdot \prod_{x_i \in X}\left[\mathrm{L}\left(x_i, *, \hat{\Theta}_{x_i|*}\right)^{Q_i} \prod_{l \in \mathcal{L}_i} \mathrm{L}\left(x_i, l, \hat{\Theta}_{x_i|l}\right)\right] \tag{4.11}$$

where

$$\mathrm{L}\left(x_i, l, \hat{\Theta}_{x_i|l}\right) = \prod_m P(X = x_i[m]|C = l, \mathcal{M}, \hat{\Theta}_{\mathcal{M}}) \tag{4.12}$$

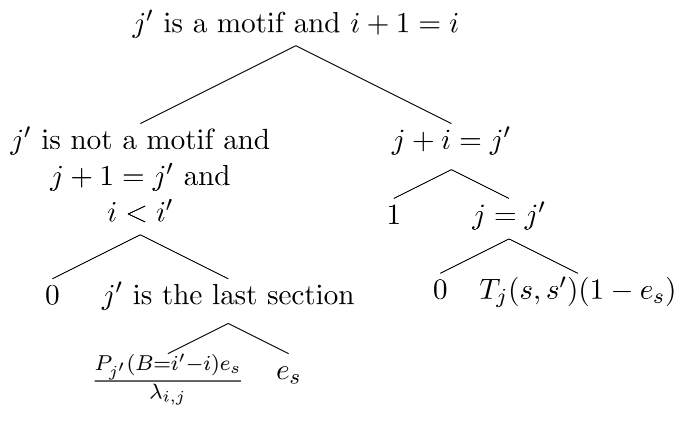
is the likelihood of variable x_i. Here, $\mathcal{M}$ represents the set of structures for all the variables, and $\hat{\Theta}_{\mathcal{M}}$ is the set of parameters for the whole model, estimated with structure $\mathcal{M}$. $\mathcal{L}_i$ is the structure for the ith variable, and $\hat{\Theta}_{x|l}$ indicates the parameters at veriable x given structure l. Finally, Q_i is the number of default clusters used by the ith variable.

During experimentation we found that the algorithm did not always make good use of all the cluster labels given to it. Some clusters would become arbitrarily small so that the probability of any data point belonging to it was extremely small. To solve this we tried two different methods of re-initializing such clusters. One method is to try to split the largest cluster. This is done by initialing the degenerate cluster to almost the same parameters as the largest cluster, but slightly perturb them. Then these two clusters would, hopefully, gradually fit two distinct parts of the data more closely. A second method is to instead set the parameters of the degenerate cluster to the average parameter values over all the cluster distributions. This should then allow the cluster to fit any part of the space more closely, as EM is iterated. Both of these methods are heuristic, of course, and provide no guarantees that the cluster will not become degenerate again. However, the goal is to help the algorithm out of local optima.

In the second stage we modified the model to make use of a more complex regulator model. In this model the regulator variables R_j are replaced by a single promoter variable, M. M is itself a Markov model which models the entire promoter sequence. This model consists of an alternating chain of two types of variables, a background variable and a regulator variable. The number of regulator variables is fixed. A background variable, B, models the length of the DNA sections between end points and regulators, $P(B|C) \sim N(\mu_B, \sigma_B)$. This essentially models where the regulators occur in the promoter sequence. Each regulator variable, R, is itself a hidden Markov model (HMM) (see Section 1.1.7.3). which models the regulator sequence. We can represent the state of M with a triple, $\omega = (i, j, s)$, where i is the current position in the promoter, j is the current state of the top level Markov model, and s is either 0 if j represents a background state, or else the state of the regulator HMM. The position, i, can advance by any amount when transitioning out of a background state, but must only advance one position at at time while in a regulator state, which includes advancing through the underlying HMM. The initial state distribution of this model is

$$\pi((i,j,s)) = \begin{cases} P_j(B = i) & j = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{4.13}$$

The model is required to start in the first background state, thus the probability of every state with $j \neq 0$ is 0. Given that $j = 0$, the probability of a state is equal to the probability that that background section has length i. The transition matrix is best described as a decision tree, where right branches are taken when the condition is true. The value of $\tau((i', j', s')|(i, j, s))$ is given by


j' is a motif and $i + 1 = i$
j' is not a motif and $j + 1 = j'$ and $i < i'$
$j + i = j'$
1
$j = j'$
0
j' is the last section
0
$T_j(s, s')(1 - e_s)$
$\frac{P_{j'}(B=i'-i)e_s}{\lambda_{i,j}}$
e_s

This is the probability of transitioning to state (i', j', s') given a current state of (i, j, s). There are only four non-zero terminal states here. If we are moving from a motif to a background state, that is, j' is not a motif, then we must also have that j increments by one and i increases by some non-zero amount. Then, if we are moving into the last section, we are forced to jump to the end of the sequence because we must account for the whole sequence. In this case we have an exit probability of e_s for exiting the current section while in HMM state s. Otherwise we get the probability of observing a gap of length $i' - i$ along with the exit probability $e_{j,s}$, dividided by $\lambda_{i,j} = \sum_l p(B = l_i - i)$, where l ranges over the valid next states of (i, j, s). If we are moving into a motif state we require that the position, i, only increment by one, in line with the requirements of the motif HMM model. If j incremented by one, then we just came from a background section. The first state of an HMM is the only possible place to transition to after a background section, so that has probability 1. Otherwise, if j stays the same, we are transitioning within the HMM, so we use T_j, the transition matrix associated with the HMM in section j. We also multiply T_j with the probability of not exiting early. We set $e_s = \frac{0.9}{2^{r-s}}$ where r is the number of states in each motif HMM.

In order to perform EM on this model we have the following sufficient statistics:

$$T_j(s, s') = \frac{\overline{\#}[\text{ transition from s to s' in regulator j }]}{\overline{\#}[\text{ being in state s in regulator j }]} \tag{4.14}$$

$$b_s^j(m) = \frac{\overline{\#}[\text{ see symbol m in state s in regulator j }]}{\overline{\#}[\text{ being in state s in regulator j }]} \tag{4.15}$$

$$N_{B_j} = \overline{\#}[\text{ being in background j }] \tag{4.16}$$

$$\text{sum}_{B_j} = \sum_l \sum_{k \in \text{ prev}(l)} (l_i - k_i)\xi(k, l)I[l_j = j] \tag{4.17}$$

$$\text{sum}^2_{B_j} = \sum_l \sum_{k \in \text{ prev}(l)} (l_i - k_i)^2\xi(k, l)I[l_j = j]. \tag{4.18}$$

T and b are the transition and observation counts needed for the HMM in the jth state. The remaining statistics are for the background length distributions. We sum first over all positions we could end in, l, then over all the positions we could have come from, denoted by prev(l). For each pair we have the length of the jump, times the number of times that jump occurred, represented by $\xi(k, l)$. Finally, only jumps ending in state j are included in the statistics for state B_j, so the indicator, I, filters these. To learn the best state sequence for a given sequence, we use the forward-backward algorithm. The forward message is defined as

$$\alpha((i, 0, 0)) = P_0(B = i)b(O_{0:i}) \tag{4.19}$$

$$\alpha(k) = \sum_{l \in \text{prev}(k)} \alpha(l)\tau_{l,k}b(l, k) \tag{4.20}$$

and the backward message is

$$\beta((i, J, 0)) = \begin{cases} 1 & k_i = end \\ 0 & k_p \neq end \end{cases} \tag{4.21}$$

$$\beta(k) = \sum_{l \in \text{next}(k)} \tau_{k,l}b(k, l)\beta(l). \tag{4.22}$$

4.2.2 Motif Discovery

As a first step towards discovering novel motifs, we developed a simple greedy search method as an addition to our existing model. In addition to the known regulators, we add N_u random regulator variables, each associated with a random DNA string of length l. We then run the above EM and SEM algorithms until convergence and then look at each random regulator variable to see if its structure indicates that it was used for some cluster. If a variable was used, then the regulator is extended until no more improvement is found. If a variable was not used, than it is replaced with another random DNA string up to 50 times, or until some improvement is found. If some improvement is found, the regulator is extended, as in the other case.

To extend a regulator, each of "C", "T", "G", and "A" are added and the improvement computed. If any of these actions improves the score, that addition is kept and the process is repeated until there is no more improvement. To compute the improvement of a new regulator, we must first count the number of times it occurs in each sequence in the dataset. Then the expected sufficient statistics are updated, as described above, and the new parameters are computed. To compute the difference in log-likelihood of the old regulator, r_1, compared to the new one, r_2, we keep r_1 as is, and add r_2 as an additional variable in the model, but with an empty structure. In this way we can pretend that r_2 has always been there, but it never affected the log-likelihood. We have two different structures then, $\mathcal{M}_1$ in which r_1 has its original structure $\mathcal{L}$ and r_2 has an empty structure, and $\mathcal{M}_2$ in which r_1 is given the empty structure, thus effectively removing it from the model, and r_2 is given structure $\mathcal{L}$. We can compute the ratio of these two likelihoods as

$$\frac{P(D|\mathcal{M}_2, \hat{\Theta}_{\mathcal{M}_2})}{P(D|\mathcal{M}_1, \hat{\Theta}_{\mathcal{M}_1})}.$$

Expanding this and canceling terms we are left with

$$\frac{\mathrm{L}\left(r_1, *, \hat{\Theta}_{r_1|*}\right)^k \mathrm{L}\left(r_2, *, \hat{\Theta}_{r_2|*}\right)^Q \prod_{l\in\mathcal{L}} \mathrm{L}\left(r_2, l, \hat{\Theta}_{r_2|l}\right)}{\mathrm{L}\left(r_2, *, \hat{\Theta}_{r_2|*}\right)^k \mathrm{L}\left(r_1, *, \hat{\Theta}_{r_1|*}\right)^Q \prod_{l\in\mathcal{L}} \mathrm{L}\left(r_1, l, \hat{\Theta}_{r_1|l}\right)}.$$

Taking the log and rewriting we end up with

$$(ll(r_1, \emptyset) + ll(r_2, \mathcal{L})) - (ll(r_1, \mathcal{L}) + ll(r_2, \emptyset)) \tag{4.23}$$

where $ll(V, S)$ is the log-likelihood of variable V with structure S. If r_2 is better, the first term will be larger and the difference will be positive. In this case r_2 would be accepted to replace r_1, otherwise we would retain r_1.

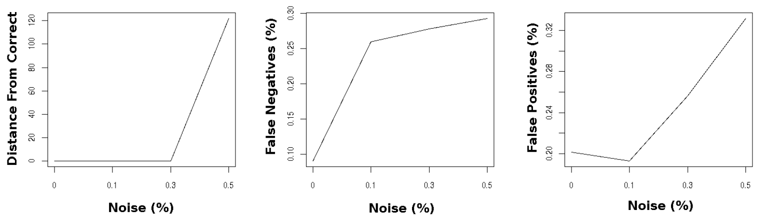


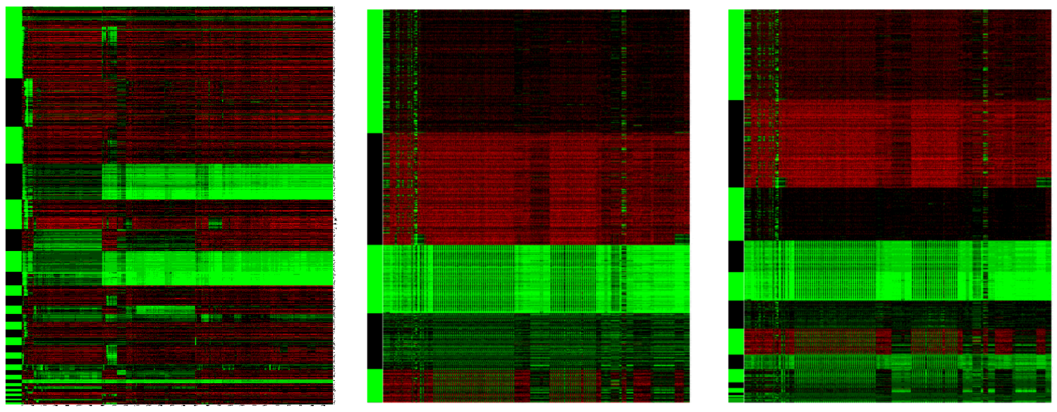
Figure 4.2: Performance of synthetic data. The first plot shows how different the learned clustering is from the correct clustering, in terms of the number of mismatched pairs. The second and third plot show the false positive and false negative rates of the regulator variables. This would be how many regulators were found that were not real regulators, and how many regulators were missed.

In the second stage of the model, this scheme is replaced by a single model, M, which models the entire promoter sequence, including the location and pattern of the regulators. With this model we don't need to perform a greedy search for specific regulators, we instead treat is as part of the rest of the model and perform EM to find the best set of parameters. This model also only works with a small number of regulators, around 2 or 3, which must be fixed before hand.

4.3 Results

To test the first stage of the model we generated some synthetic data from the model itself, mixing it with some percentage of random data. We first construct a random model with a set of random parameters and CSI structures. We then sample from this model to generate synthetic datasets. Then, since we know the correct model for the synthetic data we can test how well the algorithm is able to recover the correct model. For this test we used 5 clusters, 93 expression variables, and 25 regulator variables. We generated 50 different datasets, which 1000 samples each. For each dataset, we used the best result out of 10 restarts. The results are shown in Figure 4.2, for varying amounts of noise. We can see that the correct clustering is found perfectly when 30% or less of the data is noise. The false positive and false negative plots show how well the set of regulators was recovered. Overall, both values are good, but the false negative rate is adversely affected by noise more easily.

Tests on synthetic data can really only show that the algorithm is working. The real test is whether or not the model accurately represents real biological data. To test this aspect, we created two subsets of the real dataset. The data set used is

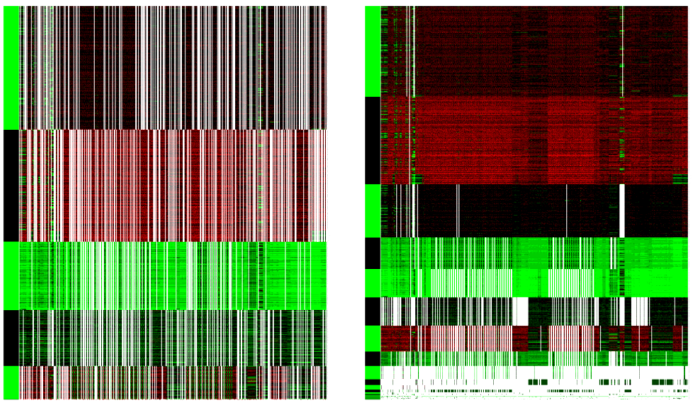


(a) Hierarchical (b) Bayesian (c) Refined

Figure 4.3: Subset 1, expression data. The green and black bars on the left denote cluster boundaries

the Arabidopsis thaliana genome, which consists of 26,000 genes. We used only the expression data from 571 experiments in these tests. In the first subset we selected 808 genes that other hierarchical clustering methods had performed well on, and the second dataset consisted of 1000 randomly selected genes. Both of these datasets are then clustered using a standard complete linkage, agglomerative hierarchical clustering algorithm, and then using our model, with and without the cluster refinement methods. The clustering results can be seen in Figures 4.3, 4.4, 4.5, and 4.6. The clusters are denoted by alternating green and black bars on the left side of each plot. Each row represents the expression data for one gene. Each column is a different experiment and the color of each cell represents the value of the expression level of a gene in a certain experiment. The color scale ranges from red to black to green, where black represents the average expression level over the entire dataset, shades of red are below average levels, and shades of green are above average levels.

It can be seen that the Bayesian clustering method creates a much smaller number of clusters, but they are much more uniform in expression levels. The cluster refinement method is able to increase the number of clusters, creating more specific clusters than the plain Bayesian method did. In the second subset, we can see that the hierarchical method creates a very poor clustering while the Bayesian method is still able to create a set of uniform clusters. The structures learned are shown in Figures 4.4 and 4.6. For each experiment (column), a white bar is shown for each cluster that had been set to the default distribution, thus indicating it was not helpful for that experiment. It can been seen that this situation did occur with some regularity, so the reduced number



(a) Bayesian (b) Refined

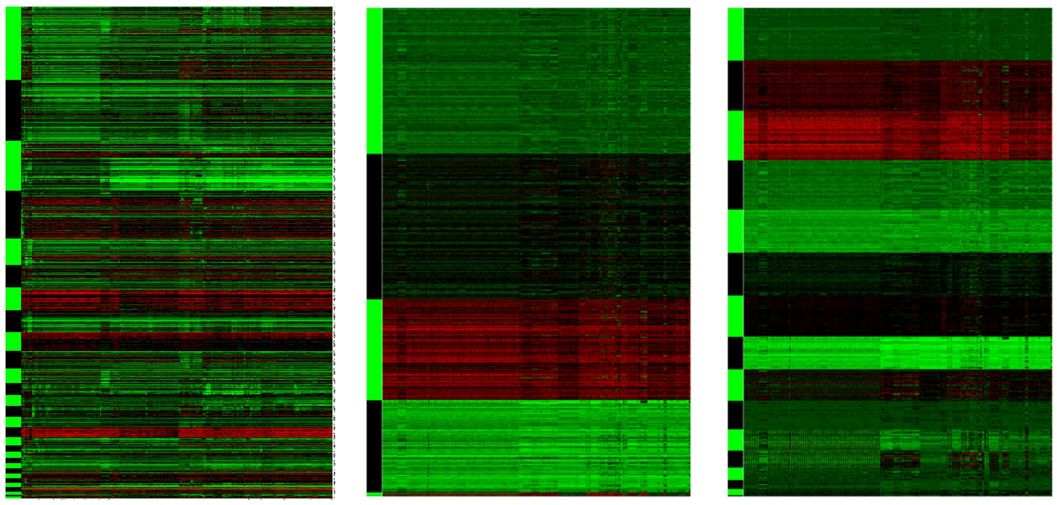
Figure 4.4: Subset 1, CSI structures

of parameters should make the learned model more robust and general.

We then compared these algorithms on the full dataset, including expression data and 237 previously known motifs. We set the number of clusters to 100. The results are shown in Figures 4.7 and 4.8. We again see a similar result as with the two previous data subsets. In Figure 4.9 we can see the results of the clustering on the regulator counts. Here the color red indicates a count of 0 and bright green indicates more than 2 occurrences. We can see that the regulator counts look quite similar across all clusters, although structures learned for each regulator do differ between clusters. This may indicate that this particular set of regulators were not useful in distinguishing among these clusters.

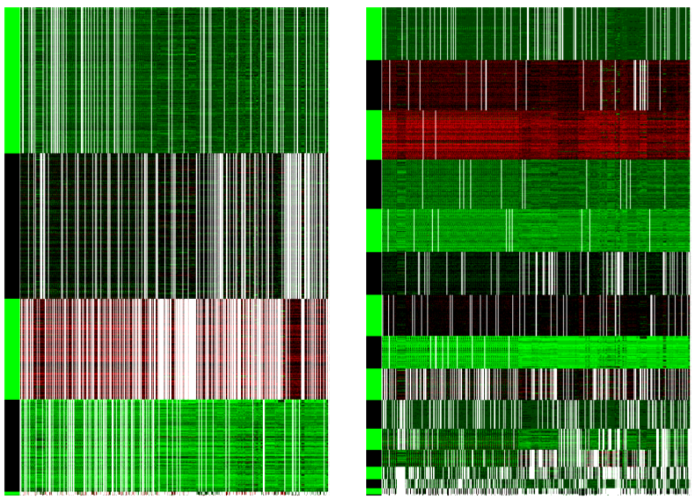
To test how well we could discover novel motifs, we generated some synthetic promoter data and injected regulator patterns into it. We generated 20 random motifs of length 7 and assigned each to a different cluster. Each promoter in each cluster then had one of its associated motifs inserted at a normally distributed position. Each promoter sequence was 500 base pairs long. We also inserted motifs at random among all promoters to simulate noise. Figure 4.10 shows the results of this experiment. In this case we can see that the regulator counts have distinct patterns in each cluster and we also found that almost all of the injected regulators where recovered.

This same dataset was used to test the second stage model, using only 2 motifs per cluster however. We will show the results of two clusters here. In the first cluster,



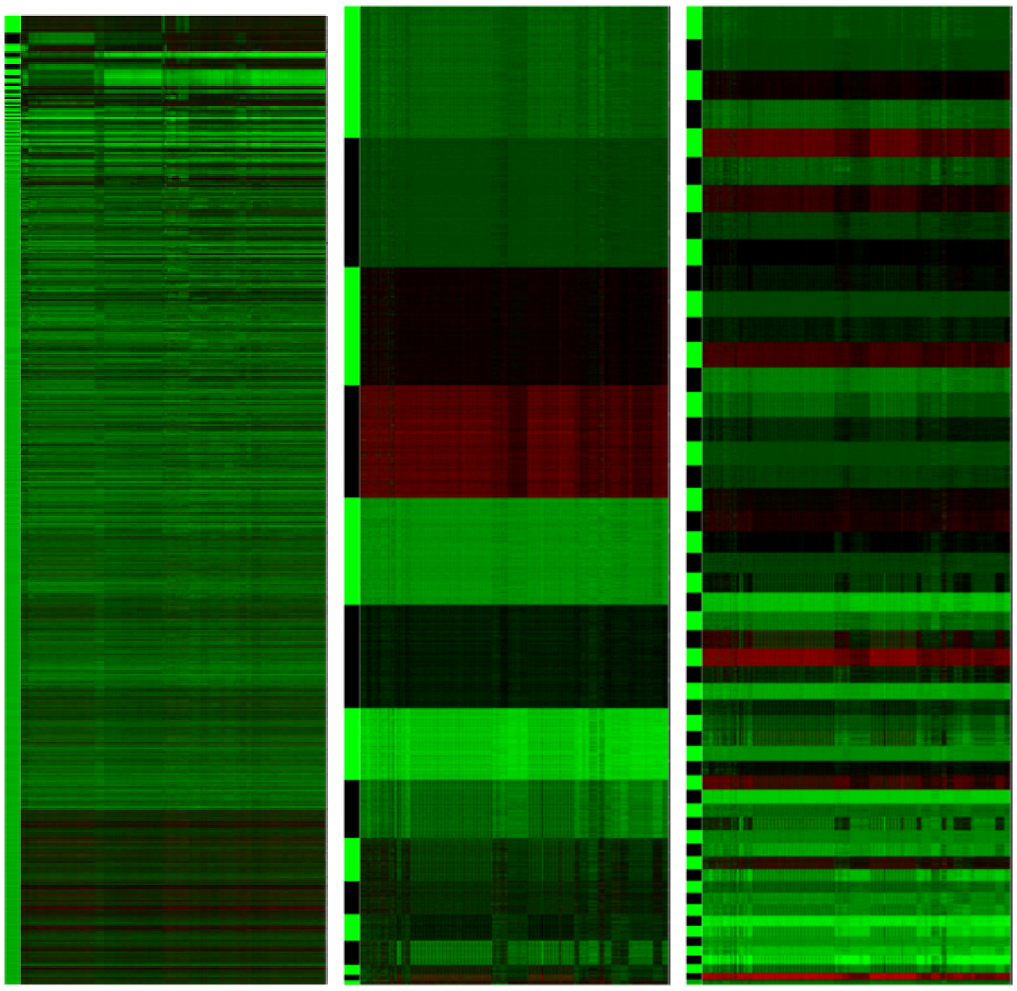
(a) Hierarchical (b) Bayesian (c) Refined

Figure 4.5: Subset 2, expression data



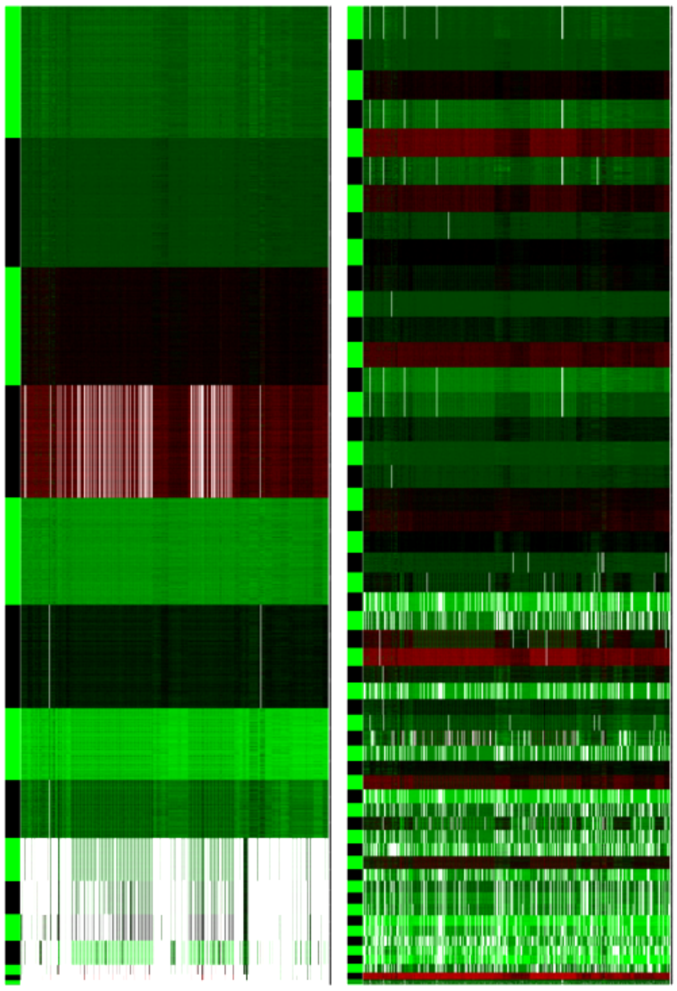
(a) Bayesian (b) Refined

Figure 4.6: Subset 2, CSI structures



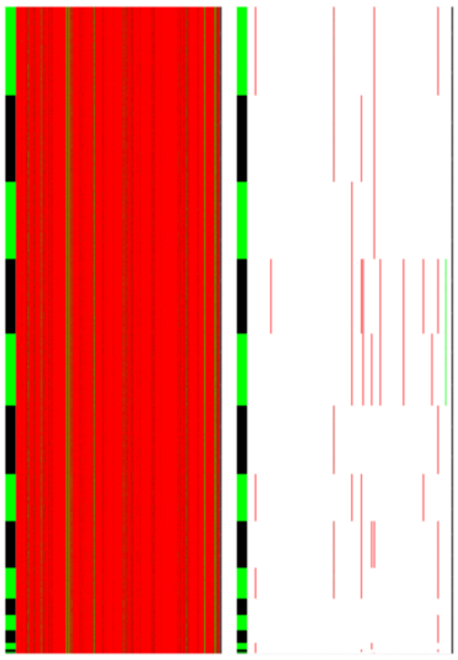
(a) Hierarchical (b) Bayesian (c) Refined

Figure 4.7: Full dataset, expression data



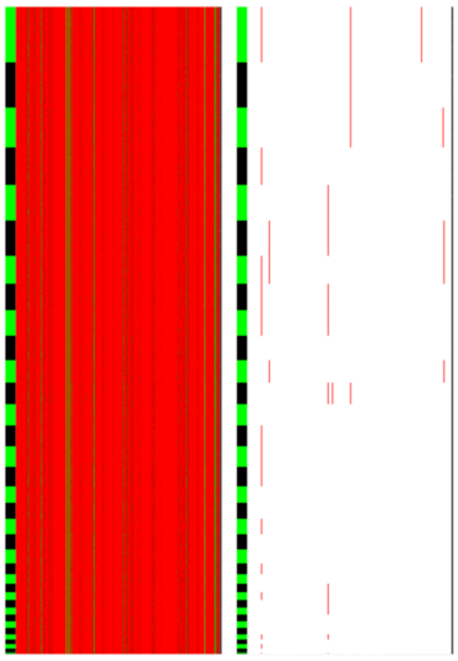
(a) Bayesian (b) Refined

Figure 4.8: Full dataset, CSI structures



(a) full (b) masked

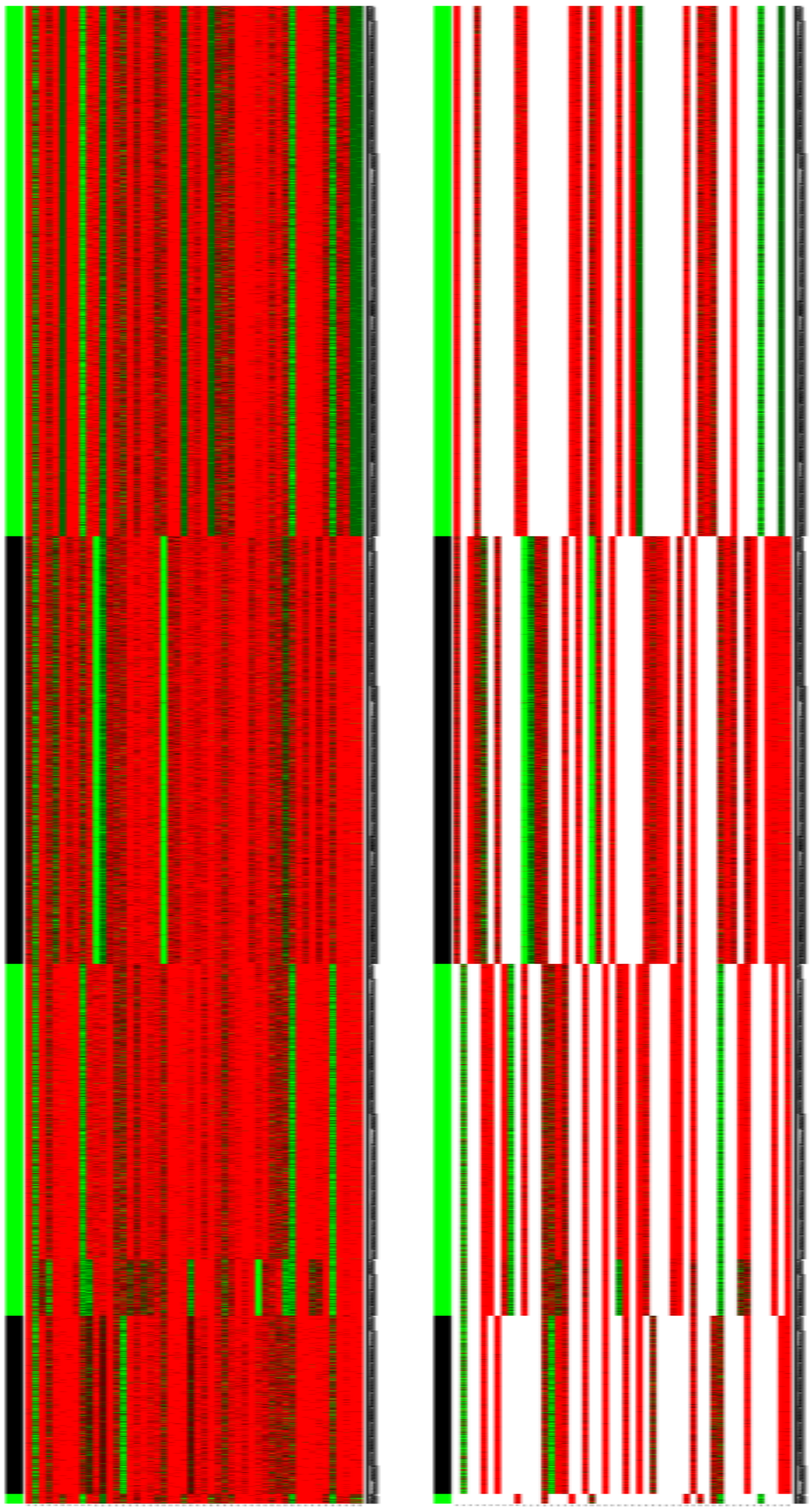
(c) Bayesian



(d) full (e) masked

(f) Refined

Figure 4.9: Clustering of regulator variables



(a) full (b) masked

Figure 4.10: Regulator clustering on synthetic data

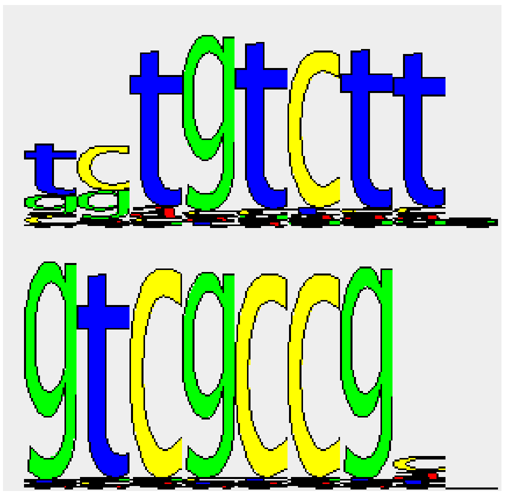


Figure 4.11: Motif discovery result on cluster 1

the two patterns inserted were "TGTCTTA" and "ATGGTCGCCG". The resulting regulator models are show in Figure 4.11. We can see that in both cases a significant part of the regulator pattern was recovered by the HMM. In the second cluster, the two patterns were "AAGCAAC" and "CCTTCAC". Figure 4.12 shows the resulting models. In this case the first pattern is recovered well, but the second one quite poorly. The performance of the model overall was very sensitive to the initial parameter settings. To initialize the model, we enumerated every 6 base pair pattern and looked for patterns which where over expressed in a cluster, according to a hyper-geometric test. The top two or three scoring patterns per cluster where used to initialize the regulator HMMs. While this was effective in synthetic data, as the above results show, results on Arabidopsis sequence data were not as good. The most likely reason for this is that higher order organisms, such as Arabidopsis, have much more complex promoter regions as well as more complex motif patterns and transcription processes. As a result, patterns are more difficult to find, and our model may have been too simple to capture these additional complexities.

4.4 Conclusion

In this work we have presented a flexible model which can make use of diverse data types to cluster genes. In addition, we have presented a novel method of discovering new motifs by iterating between clustering steps and motif discovery steps. For the

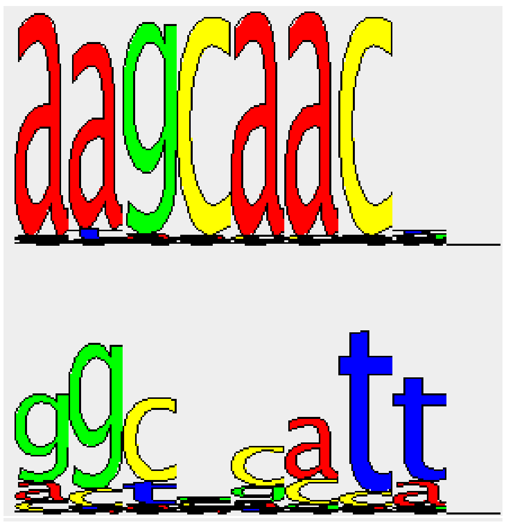


Figure 4.12: Motif discovery result on cluster 2

motif discovery step we tried both a simple greedy motif search which could find short exact motifs, and a full Bayesian model which could learn an HMM representation of discovered motifs. We also demonstrated a method to improve the utilization of given clusters, so that the desired number of clusters is actually used. Finally, both motif discovery methods where shown to be effective in discovering novel motifs in synthetic data.

Chapter 5

Predicting conserved protein motifs with Sub-HMMs*

Profile HMMs (hidden Markov models) provide effective methods for modeling the conserved regions of protein families. A limitation of the resulting domain models is the difficulty to pinpoint their much shorter functional sub-features, such as catalytically relevant sequence motifs in enzymes or ligand binding signatures of receptor proteins.

To identify these conserved motifs efficiently, we propose a method for extracting the most information-rich regions in protein families from their profile HMMs. The method was used here to predict a comprehensive set of sub-HMMs from the Pfam domain database. Cross-validations with the PROSITE and CSA databases confirmed the efficiency of the method in predicting most of the known functionally relevant motifs and residues. At the same time, 46,768 novel conserved regions could be predicted. The data set also allowed us to link at least 461 Pfam domains of known and unknown function by their common sub-HMMs. Finally, the sub-HMM method showed very promising results as an alternative search method for identifying proteins that share only short sequence similarities.

Sub-HMMs extend the application spectrum of profile HMMs to motif discovery. Their most interesting utility is the identification of the functionally relevant residues in proteins of known and unknown function. Additionally, sub-HMMs can be used for highly localized sequence similarity searches that focus on shorter conserved features rather than entire domains or global similarities. The motif data generated by this study is a valuable knowledge resource for characterizing protein functions in the future.

* Originally published in Horan et al. (2010), used with permission.

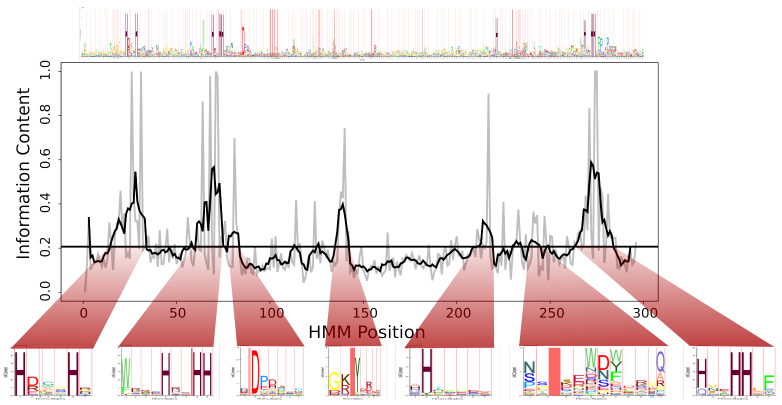


Figure 5.1: An example of the sub-HMM excision process is given for the fatty acid desaturase domain (PF00487). The light gray line is the KL-divergence of each position in the original HMM. The darker line is the result of smoothing with $s = 8$. The horizontal line is the threshold, set to the average KL-divergence. Each section of the curve with more than $l = 8$ positions above the threshold produces a sub-HMM. The details about the different parameters are given in the Method section.

5.1 Extracting Sub-HMMs

The proposed *protein sub-HMM* method starts with a profile HMM that has been trained on the multiple alignment of a protein family. We then extract sub-HMMs from the generated HMM. A robust scoring method is used to predict the presence of the sub-HMMs in any protein sequence of interest.

The first step is to find a small set of the most information-rich sub-HMMs (or smaller HMMs), which still captures most of the information present in the corresponding regions of a larger HMM built for a particular family. To extract the desired sub-HMMs we first compute the Kullback-Leibler divergence (or relative entropy) of each position in the large HMM, $h_i = D_{KL}(m_i||B)$, where m_i is the observation distribution of the match state at position i, and B is the background distribution. We normalize h so that each position has a value between 0 and 1, and then smooth the values as follows: $h_i^s = \frac{1}{s}\sum_{j=1}^{s} h_{i-j}$ for $i \geq s$. The smoothness of the curve is determined by parameter s, with higher values producing a smoother curve. Then we extract every sub-sequence of h^s that is always above some threshold t, and is at least l positions long. We always set t to be the average over h_i. An example is shown in Figure 5.1a.

5.2 Scoring

Sub-HMMs can be matched and scored against protein sequences either as single models or as sets of models. When scoring a set of sub-HMMs against a protein sequence S, such as all sub-HMMs extracted from a Pfam HMM, we used a method based on a complete generative model. We hypothesize the entire protein sequence can be generated according to the following sampling semantics: First, choose the length of the sequence. Then, for each sub-HMM y, sample the starting location from a uniform distribution, and then sample a sequence from y and place it at the chosen starting point. After this is done for all the sub-HMMs, fill in the gaps with samples from the background distribution. This assumes that each of the sub-HMMs generates a portion of the protein sequence, while their order is not important. In addition, we ignore possible overlaps among sub-HMMs. We use the Viterbi algorithm to find, for each sub-HMM, the most likely hidden state sequence and position in S, using a local-local alignment. Let M be the length of S and Y the set of sub-HMMs. Then the resulting score is

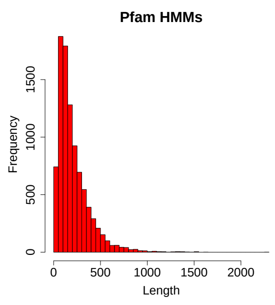
$$\text{final\_score}(S) = \sum_{y \in Y} \text{score}_y(S) - |Y| \log M \ . \tag{5.1}$$

Here $\text{score}_y(S)$ is the score from Equation 1.2 for HMM y. The term $|Y| \log M$ arises from the uniform distribution over positions at which any sub-HMM might begin.

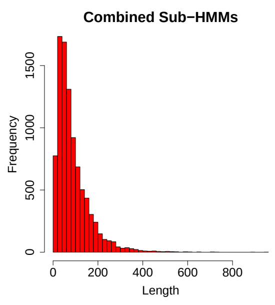
5.3 Data Set

Sub-HMMs were extracted from Pfam domain families using HMMER2 and HMMER3 models (Finn et al., 2010). Pfam 22.0 was used for all experiments, whereas Pfam 24.0 was mainly used in the performance comparisons with HMMER3. This is because Pfam has adopted HMMER3 models only very recently, and at the time this work was done many of its families had not been as rigorously tested and curated by experts in the field as in the earlier HMMER2-based releases.

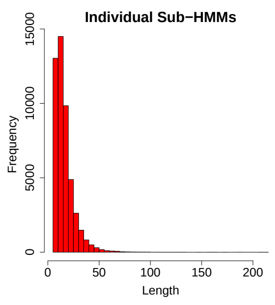
Using our new sub-HMM method, we extracted 48,535 sub-HMMs (Table 5.3) from the Pfam 22.0 database (Pfam-A, Pfam_ls). This database consisted of 9,318 domain profile HMMs with 2,990,695 unique protein sequences associated with at least one domain. Due to the presence of multiple domains in many sequences, the data set contained a total of 4,070,949 family memberships. The length distributions of the original Pfam HMMs and our sub-HMMs for all families are shown in Figure 5.2. As expected the sub-HMMs are much shorter than the original Pfam HMMs, with an average length of 17 residues compared to 210 residues, respectively. This has several



(a)



(b)



(c)

Figure 5.2: (a) The length distribution of Pfam HMMs is depicted in the form of a histogram. The Pfam HMMs consist on average of 210 positions, while it is only 90 positions for the combined set of sub-HMMs per Pfam HMM. (b) The length per domain model is computed by summing the lengths of the sub-HMM extracted from that model. (c) The length distribution of individual sub-HMMs is shown. In all cases the sub-HMMs were created with a minimum length setting of 8 and a smoothing factor of 8.

Name	Size	Description
Pfam proteins	2,990,695	Proteins in Pfam database
Pfam HMMs	9,318	Domains in Pfam database
DKFs	7,435	Pfam domains of known function
DUFs	1,883	Pfam domains of unknown function
Sub-HMMs	48,535	Sub-HMMs excised from Pfam domains
Sub-DKFs	39,217	Sub-HMMs excised from DKFs
Sub-DUFs	9,318	Sub-HMMs excised from DUFs

Table 5.1: The sizes of the different data sets used and generated by this study using Pfam 22.0.

advantages for the goals of this study. First, the sub-HMMs have a length distribution similar to the size of many known functional motifs, which is essential for predicting features with related properties (Hulo et al., 2006, 2008). Second, their shorter length reduces the computation time for scoring a protein. Finally, it reduces the number of parameters, which should improve the accuracy of the detector.

Subsequently, we performed several benchmark tests to determine the performance of the new sub-HMM method in identifying functionally relevant sequence features and searching for sequences sharing them. For this, we determined the presence of each Pfam HMM and our sub-HMMs in all protein sequences from the Pfam database by applying the scoring system described in Section 5.2.

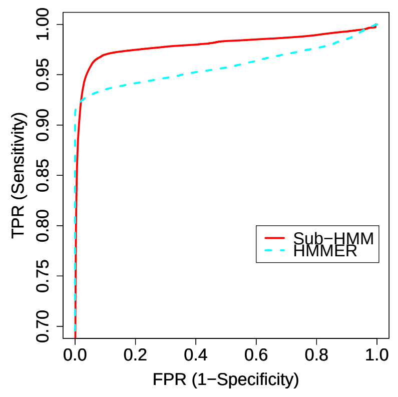


Figure 5.3: The true positive rates versus false positive rates are plotted in the form of ROC (Receiver Operating Characteristic) curves to compare the performance of the HMMER2 and sub-HMM methods. The full Pfam 22.0 data set was considered in this test.

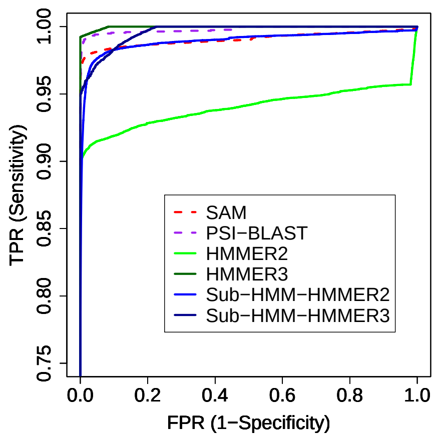
5.4 Search Performance

In order to compare the sensitivity and selectivity performance of the sub-HMM method with the widely used HMMER2 software, we tested how well each method could recover the members of each domain family from all proteins in the entire Pfam 22.0 database. We used the scores computed for each protein to generate an ROC (Receiver Operating Characteristic) curve for each method (Figure 5.3). This allowed us to compare the methods without choosing a fixed threshold, which is usually hard to define *a priori*. In this preliminary test, we used the original Pfam HMMs for the HMMER2 method, and the sub-HMMs extracted by our method from the same Pfam HMMs. As a test sample, all proteins in Pfam were used. This experimental design gives a slight advantage to both methods, because the Pfam HMMs are trained on a representative subset of proteins that overlaps with the total protein set in each family. Despite this limitation, the difference in performance is still meaningful due to the identical starting conditions for both methods. Figure 5.3 shows the resulting ROC curves for assembling all 9,318 families. The results show that the HMMER2 method has a higher sensitivity at false positive rates less than 0.02, but the sub-HMM method performs slightly better at higher false positive rates. Due to the much shorter profiles used by our method, it is expected to have a higher false positive rate when it is benchmarked against a test data set that is based on the family assignments of complete domain models.

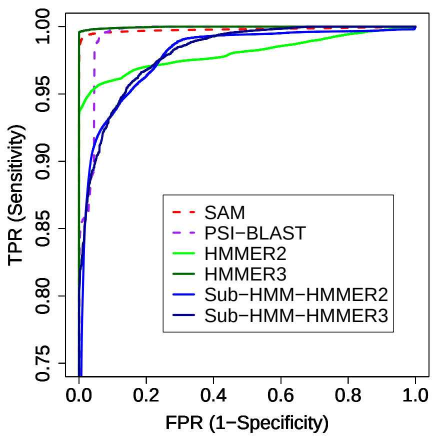
We also performed more rigorous comparisons of our method against HM-

MER2, HMMER3, SAM and PSI-BLAST (Altschul et al., 1997). Additionally, we tested our sub-HMM method with HMMER3 profile HMMs. In this case the sub-HMMs where excised from HMMER3 models and the HMMER3 search tool was used to map and score the individual sub-HMMs to the sequences. We then combined the scores as described in the Methods section. In the following text of this section, the sub-HMM experiments performed with HMMER2 and HMMER3 are referred to sub-HMM-HMMER2 and sub-HMM-HMMER3, respectively. In all tests we trained the models ourselves by randomly selecting 20% of the members from each protein family, but the training data were not included in the test data sets. HMMER2, HMMER3 and SAM use a multiple sequence alignment for the model building step. Since it was not our goal to test the alignment quality, we used the curated domain alignments provided by Pfam as input to all methods. Although SAM can create its own alignments, we forced it to use the alignments we provided to make this method more comparable to HMMER2 and HMMER3. For PSI-BLAST, we first created multiple sequence alignments for all the training data sets using CLUSTALW. Subsequently, we built PSSMs to search the test data set with PSI-BLAST. For all methods, we compared how well they could recover the remaining 80% in each protein family from the combined set of all test sequences. Due to computational resource constraints, it was not possible to test these methods on all Pfam families. Instead we created two smaller subsets of families, one composed of smaller families and one composed of larger families. The small family set contained 933 families randomly selected from Pfam 22.0 with of 10 to 100 members, while the large set contained 1002 families with more than 100 members. In addition, we tested the different methods on the HMMER3-based Pfam 24.0 data set. To maximize the comparability of the results, we selected only families that were available in both Pfam releases and fell into the same size categories. For the small set, we found 899 families in Pfam 24.0 but only 491 of them had less than 100 members. For the large set, 988 families were also available in Pfam 24.0 and all of them contained more than 100 members.

The ROC plots for all comparisons are shown in Figures 5.4 and 5.5. For the experiments with Pfam 22.0, the results indicate that the sub-HMM-HMMER2, sub-HMM-HMMER3 and PSI-BLAST methods perform better on the small family set than on the large one, while HMMER2, HMMER3 and SAM show an opposite performance trend. When comparing the six methods, both sub-HMM methods perform at least as well as HMMER2, whereas SAM, HMMER3 and PSI-BLAST show the best performance in assembling the families from both family size categories. Direct comparisons of the Pfam 22.0 and Pfam 24.0 results indicate that HMMER3, PSI-BLAST, SAM, sub-



(a) Small



(b) Large

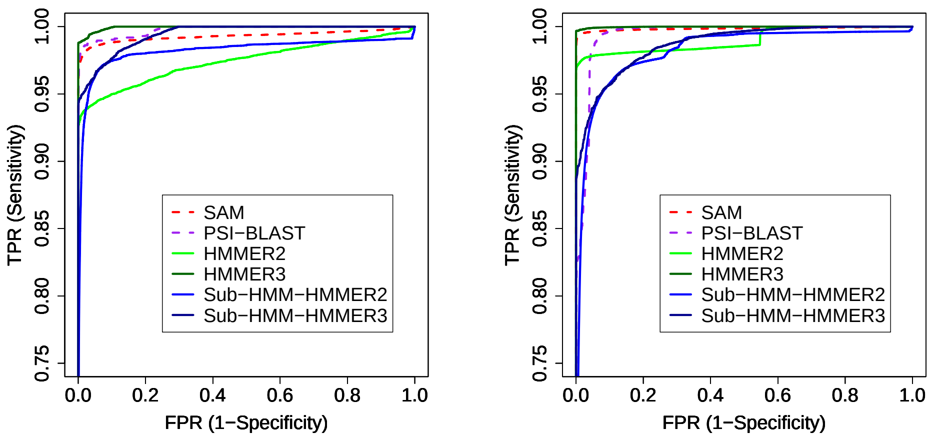
Figure 5.4: The performances of sub-HMM-HMMER2, sub-HMM-HMMER3, HMMER2, HMMER3, SAM and, PSI-BLAST on the Pfam 22.0 data set are compared in the form of ROC curves (compare Figure 5.3). The first test (a) considers smaller families with 10 to 100 members, whereas the second one (b) considers large families with more than 100 members.

HMM-HMMER2 and sub-HMM-HMMER3 perform very similarly on the small family set, while HMMER2 improves slightly. These trends are almost identical for the large family set, except that sub-HMM-HMMER3 performs better on this data set.

Since our method is designed to find short sequence similarities, it is expected to have a lower selectivity (higher false positive rate) than the other methods when reassembling family relationships that are based on longer domain similarities. In fact, such a performance characteristics on known family data sets is required for discovering novel conserved fragments in sequences that do not necessarily belong to the same domain family. The latter is the main utility feature of the sub-HMM method.

5.4.1 ROC Comparisons

For the PSI-BLAST tests, the training sets were aligned with CLUSTALW (Thompson et al., 1994) and then a PSSM was generated using blastpgp with just one round of searching. The test data was then scored by blastpgp using the trained PSSM as a starting point and running for up to 6 rounds. For each sequence, we recorded the maximum log-odds score from all the rounds. For the SAM tests, we extracted the



(a) Small (b) Large

Figure 5.5: Performance comparisons with Pfam 24.0. The first test (a) considers families used for Figure 5.4 that were also present in Pfam 24.0 for the small (a) and the large sets (b).

aligned training data from the Pfam database and used them to train the models, forcing SAM to use the given alignments rather than create its own. These models were then used to classify the test data. In the case of HMMER2 and HMMER3, we trained models with hmmbuild and hmmcalibrate (HMMER2 only) using the same alignments as for the SAM tests. In all cases, HMMER2 tests were performed with HMMER2 models and HMMER3 tests with HMMER3 models. We then used these models to classify the test data with hmmsearch. If multiple domains were found in one sequence, the result from the best scoring one was used. For the sub-HMM method, we used the aligned training data to build HMMER2 and HMMER3 models, and then extracted sub-HMMs from them. We then used our hmmsearch implementation to score each sequence according to our model. For all tests, the training sets consisted of a random selection of 20% of the sequences from each Pfam family, while the test database contained the union of the remaining sequences. The ROC curves where computed with the ROCR library (Sing et al., 2005) using the concatenation of all the scores for each method. Log-odds scores were used for all methods to obtain comparable results. In the case of SAM, we used reverse log-odds scores (Karplus et al., 2005).

5.5 Cross Validation with PROSITE and CSA

Next, we determined how well the sub-HMM method performed in identifying known motifs that are likely to be of functional relevance. This was addressed by comparing the extracted sub-HMMs from the Pfam 22.0 database with the hand curated conserved protein motifs from the PROSITE database. If the sub-HMMs are enriched in functionally relevant candidates, then one would expect a high degree of overlap with the motifs from the PROSITE database. This should be the case because the PROSITE motifs are derived from a comparable protein knowledge space as the sub-HMMs generated by this study. The overlaps were determined by comparing the matching positions of the two fragment data sets in their corresponding protein family sequences. For counting overlaps, we used relatively conservative filtering criteria: the two fragment models had to have 50% of their matching protein sequences in common and the overlaps had to occur in least 95% of the common protein members. In addition, we consider a sub-HMM to match only if it has a score of 0 or higher. Furthermore, we compute the probability of this event happening by chance and require that it be less than 0.01.

According to these comparisons, 1,054 of the 48,535 sub-HMMs overlapped with 937 of the 1,303 (72%) PROSITE motifs by at least 10% of the length of the shortest fragment. The probability of finding $\geq$937 matches just by chance was estimated to be $< 1.6 * 10^{-6}$ (see Section 5.5.1 for details). Of these 1,303 PROSITE motifs, 958 were associated by Pfam with one or more of its protein families. The number of matching families for varying percent overlaps is shown in Table 5.2. An example of a matching pair is shown in Figure 5.6.

A similar test was performed for the catalytic residue annotations from the Catalytic Site Atlas (CSA) (Porter et al., 2004). This is a database of active site residues from enzymes represented in the Protein Data Bank (PDB). Due to their functional importance, most of these residues are highly conserved within protein families. In our tests, we considered only those sites which are supported by the literature and also mapped to protein domain regions in the Pfam data set. This left us with 4147 sites mapping to 642 proteins. Subsequently, we counted how many sub-HMMs overlapped with these sites and found that 847 sub-HMMs overlapped with CSA residues. These corresponded to 2903 active sites from 546 proteins. Thus, our sub-HMM data set contained 70% of these active sites. The probability of observing $\geq$2903 overlaps among the two data sets just by chance is $< 1.5 * 10^{-18}$.

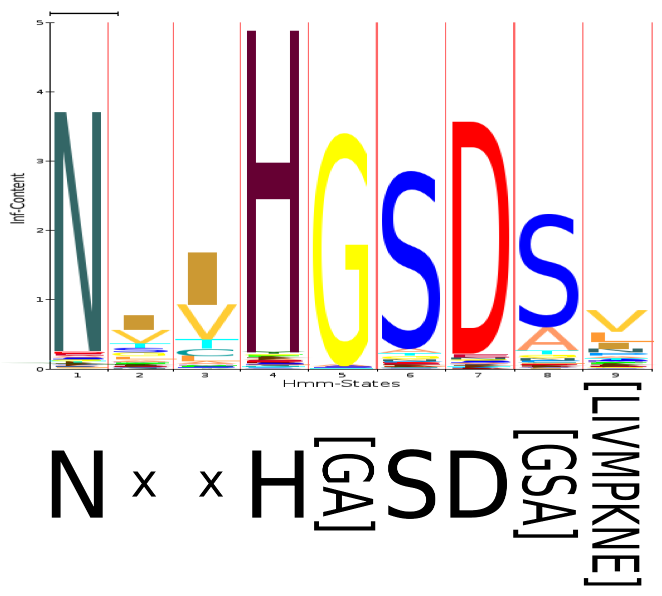


Figure 5.6: An example of a sub-HMM matching with a PROSITE pattern. In this case, the sub-HMM (PF00334.10-3, top) extracted from domain PF00334.10 matched against PS00469 (bottom) with a p-value of less than 10^{-315}.

Overlap	Sub-HMMs	PROSITE	TP	TPR
10%	**1,054**	**937**	**562**	**0.58**
25%	1,023	932	558	0.58
50%	965	912	549	0.57
75%	849	827	495	0.51
90%	720	716	423	0.44
100%	620	624	366	0.38

Table 5.2: The numbers of sub-HMMs are listed that overlapped with PROSITE motifs. The first column provides the relative overlap among the two feature types. The second and third columns contain the number of overlapping sub-HMMs and PROSITE motifs, respectively. The details of the filter settings used in these comparisons is given in the Result and Discussion section. The column TP contains the number of true positives that we identified out of the 958 PROSITE families annotated by Pfam 22.0. The last column TPR gives the corresponding true positive rate.

Match Type	Sub-HMMs	Pfam HMMs	OL PROSITE/Sub-HMM
sub-DKF $\rightarrow$ DKF	28,794	6,571	689
sub-DKF $\rightarrow$ DUF	21,615	1,751	502
sub-DUF $\rightarrow$ DKF	6,798	5,487	0
sub-DUF $\rightarrow$ DUF	5,070	1,516	0

Table 5.3: The table lists the numbers of sub-DKFs and sub-DUFs which matched in addition to their source families other DKF and DUF families. A sub-HMM is considered to have matched a Pfam 22.0 family if it scores greater than 0 on more than 50% of its members. The last column contains the counts of sub-HMMs that also overlapped with PROSITE motifs.

The considerable agreement of our method with the PROSITE and CSA data sets indicates that the sub-HMM method is efficient in identifying many of the known functionally important residues in protein families. Therefore, it is reasonable to assume that the novel conserved regions, identified by this study, are a useful resource for characterizing the functional hotspots in protein sequences of known or unknown function in the future.

5.5.1 Comparison Method

The overlaps of sub-HMMs and PROSITE motifs were computed by matching them against the domain sequences in each Pfam family. The PROSITE matches were determined with ps_scan (Gattiker et al., 2002). To minimize the compute time of these overlap comparisons, we considered only those Pfam and PROSITE sets (families) which had at least 50% of their sequences in common. Among these, at least 95% of the matches had to overlap by variable lengths specified in Table 5.3. The overlaps with the CSA data set were computed similarly. Due to the short length of the active sites, their positions had to be completely contained in the sub-HMM matches. The probability of a sub-HMM matching with a PROSITE motif by chance was computed as follows.

Let q_{ij} be the probability that by chance a query fragment of length F_j overlaps a PROSITE fragment of length P_j on a protein of length S_j in PFAM family i by a fraction of at least x, assuming both fragments were placed uniformly at random:

$$q_{ij} \leq \min\left(1, \frac{P_j + F_j - 2x\min(P_j, F_j) + 1}{S_j - F_j + 1}\right) \quad (5.2)$$

Then we want to compute the probability, D_i, that a certain number of overlaps occurs between a PFAM family i and a PROSITE family. In particular, given that at least 50% of the members of either family lie in the intersection, we want the probability that 95% of the sequences in the intersection have an overlapping fragment.

Let F be a PFAM family and P be a PROSITE family. We define $\mathcal{R}$ as the set of all subsets of $F \cap P$ which contain at least 95% of the intersection:

$$\mathcal{R} = \{R | R \subset F \cap P \wedge |R| \geq 0.95n\} \tag{5.3}$$

where $n = |F \cap P|$. Let $p_{ij} = \{q_{ij} | j \in F \cap P\}$, then

$$D_i = \sum_{R \in \mathcal{R}} \left(\prod_{j \in R} p_{ij} \prod_{j \in (F \cap P) \setminus R} 1 - p_{ij} \right) \tag{5.4}$$

Since this would require enumerating every set in $\mathcal{R}$, this would take too long to calculate, so we approximate it with an upper bound. Let $j^* = \text{argmax}_j p_{ij}$ and $R^* = \text{argmin}_R(|R|)\ \forall R \in \mathcal{R}$. Then we have

$$\begin{aligned}
D_i &\leq \sum_{R \in \mathcal{R}} \prod_{j \in R} p_{ij} && (5.5)\\
&\leq \sum_{R \in \mathcal{R}} p_{ij^*}^{|R|} && (5.6)\\
&\leq |\mathcal{R}| p_{ij^*}^{|R^*|} && (5.7)\\
&= \left(\sum_{k=\lceil 0.95n \rceil}^{n} \binom{n}{k} \right) p_{ij^*}^{\lceil 0.95n \rceil} && (5.8)
\end{aligned}$$

This bound is often too loose in practice however. This is because for large values of p_{ij^*}, the last term in equation 5.4 makes that term very small, whereas the corresponding term in our bound would still be large. Therefore, we adopt a method of removing these large outliers to get a tighter bound.

$$\begin{aligned}
D_i &= \sum_{R \in \mathcal{R}} \left(\prod_{j \in R} p_{ij} \prod_{j \in (F \cap P) \setminus R} 1 - p_{ij} \right) && (5.9)\\
&= \sum_{R \in \mathcal{R}} \left(\prod_{\substack{j \in R \\ j \leq n'}} p_{i(j)} \prod_{\substack{j \in R \\ j > n'}} p_{i(j)} \prod_{j \in (F \cap P) \setminus R} 1 - p_{ij} \right) \forall n' \in [1, n] && (5.10)\\
&= \min_{n' \in [1,n]} \sum_{R \in \mathcal{R}} \left(\prod_{\substack{j \in R \\ j \leq n'}} p_{i(j)} \prod_{\substack{j \in R \\ j > n'}} p_{i(j)} \prod_{j \in (F \cap P) \setminus R} 1 - p_{ij} \right) && (5.11)
\end{aligned}$$

To simplify the notation, we re-write this in terms of the following sets:

$$U = F \cap P \tag{5.12}$$

$$U_- = \{x | x \in U \wedge p_{ix} \leq p_{in'}\} \tag{5.13}$$

$$U_+ = U \backslash U_- \tag{5.14}$$

$$\mathcal{R}_- = \{S | S \subset U_- \wedge |S| \geq n' - 0.05n\} \tag{5.15}$$

$$\mathcal{R}_+ = 2^{U_+} \tag{5.16}$$

These essentially divide U and $\mathcal{R}$ into their corresponding sets for elements less than $p_{in'}$ and elements greater than $p_{in'}$. Now we can rewrite Equation (5.11) as:

$$\leq \min_{n' \in [1,n]} \sum_{S^- \in \mathcal{R}_-} \sum_{S^+ \in \mathcal{R}_+} \left(\prod_{j \in S^-} p_{ij} \prod_{j \in S^+} p_{ij} \prod_{j \in U_- \backslash S^-} 1 - p_{ij} \prod_{j \in U_+ \backslash S^+} 1 - p_{ij} \right) \tag{5.17}$$

$$= \min_{n' \in [1,n]} \sum_{S^- \in \mathcal{R}_-} \left(\prod_{j \in S^-} p_{ij} \prod_{j \in U_- \backslash S^-} 1 - p_{ij} \right) \sum_{S^+ \in \mathcal{R}_+} \left(\prod_{j \in S^+} p_{ij} \prod_{j \in U_+ \backslash S^+} 1 - p_{ij} \right) \tag{5.18}$$

$$= \min_{n' \in [1,n]} \sum_{S^- \in \mathcal{R}_-} \left(\prod_{j \in S^-} p_{ij} \prod_{j \in U_- \backslash S^-} 1 - p_{ij} \right) \tag{5.19}$$

The last step follows because the last sum in Equation (5.18) is over every subset of U_+, so it sums to 1.

We can then bound this expression as follows:

$$\min_{n' \in [1,n]} \sum_{S^- \in \mathcal{R}_-} \left(\prod_{j \in S^-} p_{ij} \prod_{j \in U_- \backslash S^-} 1 - p_{ij} \right) \leq \min_{n' \in [1,n]} \sum_{S^- \in \mathcal{R}_-} \prod_{j \in S^-} p_{ij} \tag{5.20}$$

$$\leq \min_{n' \in [1,n]} \sum_{S^- \in \mathcal{R}_-} (p_{in'})^{|S^-|} \tag{5.21}$$

$$\leq \min_{n' \in [1,n]} \sum_{k = n' - \lfloor 0.05n \rfloor}^{n'} \binom{n'}{k} p_{in'}^k \tag{5.22}$$

In equation (5.22), we replace the sum in the previous equation with a sum over the possible sizes of R. For each size, the binomial term gives the number of sets of size k, and the last term gives the probability of a set of size k.

We use the Hoeffding bound (Hoeffding, 1963) to upper-bound the likelihood of finding a certain number of PROSITE or CSA overlaps with our sub-HMM data set by chance (that is, if the sub-HMM data set had instead been chosen at random). The

Hoeffding bound states that if the random chance of any single test matching is p, then the probability of m or more matches in M tests is less than $e^{-2M\epsilon^2}$ where $\epsilon = \frac{m}{M} - p$.

For the PROSITE comparisons, matches are only considered if the prior probability is less than 0.01, therefore, $p = 0.01$. We found $m = 1,054$ overlaps out of a total set of $M = 48,535$ sub-HMMs. This yields a p-value (by the Hoeffding bound) of less than $1.6 * 10^{-6}$ for the probability of our sub-HMMs matching PROSITE models at this level by chance.

For the CSA comparison, each site is only a single amino acid. We restrict the comparisons to only those sequences containing annotated CSA sites. There are $M = 95,076$ amino acids matching our sub-HMMs, of which $m = 2,903$ are annotated by CSA. There are a total of $261,857$ amino acids, of which $4,147$ are annotated by CSA. Therefore, $p = \frac{4147}{261857}$, and we obtain (again with the Hoeffding bound), a p-value of less than $1.5 * 10^{-18}$ for the probability of our sub-HMMs overlapping these CSA-annotated amino acids by chance.

5.6 Discovery of Conserved Fragments in Protein Families

To evaluate the utility spectrum of sub-HMMs for conserved feature discovery, we tried to see which other families sub-HMM performed well in, excluding the family a sub-HMM was excised from. We used Pfam 22.0 for this test. To define a match between a sub-HMM and a family, we required a sub-HMM to match at least 50% of the sequences in each Pfam family with a log-odds score of 0 or higher. Table 5.3 shows how many sub-HMMs from Pfam domains of unknown function (DUFs) matched Pfam families of known function (DKFs) and vice versa. A sub-DUF is defined as a sub-HMM that was extracted from a DUF, whereas a sub-DKF was extracted from a DKF. Interestingly, the sub-DKFs shows considerable overlaps with the PROSITE data set, whereas the sub-DUFs do not overlap with PROSITE at all (last two rows in Table 5.3). The latter is expected because PROSITE focuses on motifs from functionally characterized proteins. This also indicates that our sub-DUF data sets contains many novel conserved and potentially functional motifs that are not represented in PROSITE.

A similar approach was used for constructing networks of Pfam 22.0 families by their common sub-HMM matches. The obtained clusters in this network showed many similarities to the clusters from the Pfam clan database, but also significant differences (Finn et al., 2006). The Variation of Information (VI) coefficient (Meilă, 2007) for the two network sets was 0.275. This score has a range from 0 to log(9318) = 9.1, with lower scores indicating more similar clusterings. Two small sub-graphs of the sub-HMM

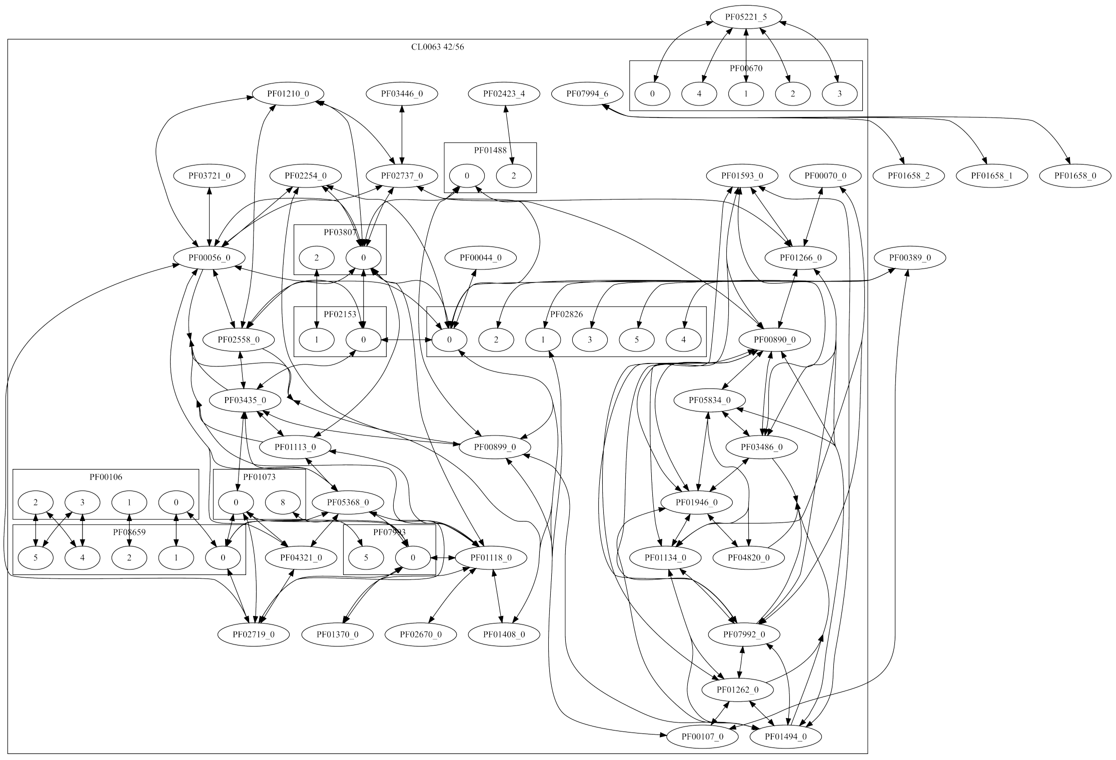


Figure 5.7: The graph shows an example of a Pfam clan and the corresponding sub-HMM network. The sub-HMM method clusters Pfam domains by conserved fragments. In the given example, the results from both methods agree very well. The Pfam clan membership is indicated by the large box labeled CL0063. The oval nodes with a PF* label represent domain families for which only one sub-HMM was created. The rectangular boxes labeled with a PF* number represent that domain family and nodes inside are sub-HMMs created from that family.

based domain network are shown in Figures 5.7 and 5.8. The box in Figure 5.7 encloses those families which are part of a clan according to the Pfam database. In this case the sub-HMM-based grouping of families agrees almost perfectly with the corresponding Pfam clan assignment. In contrast to this, Figure 5.8 gives an example of a new cluster of domains predicted by our method. Such differences in the results of the two methods are expected, because the Pfam clans are assembled with a profile HMM to profile HMM alignment method (Madera, 2008) that is fundamentally different from our sub-HMM method.

The large number of sub-HMMs matching different Pfam domains indicates the usefulness of our sub-HMM approach for discovering short sequence features that are conserved among different protein domains. Due to their high conservation, an important functional role for many of these features can be expected. Many of the

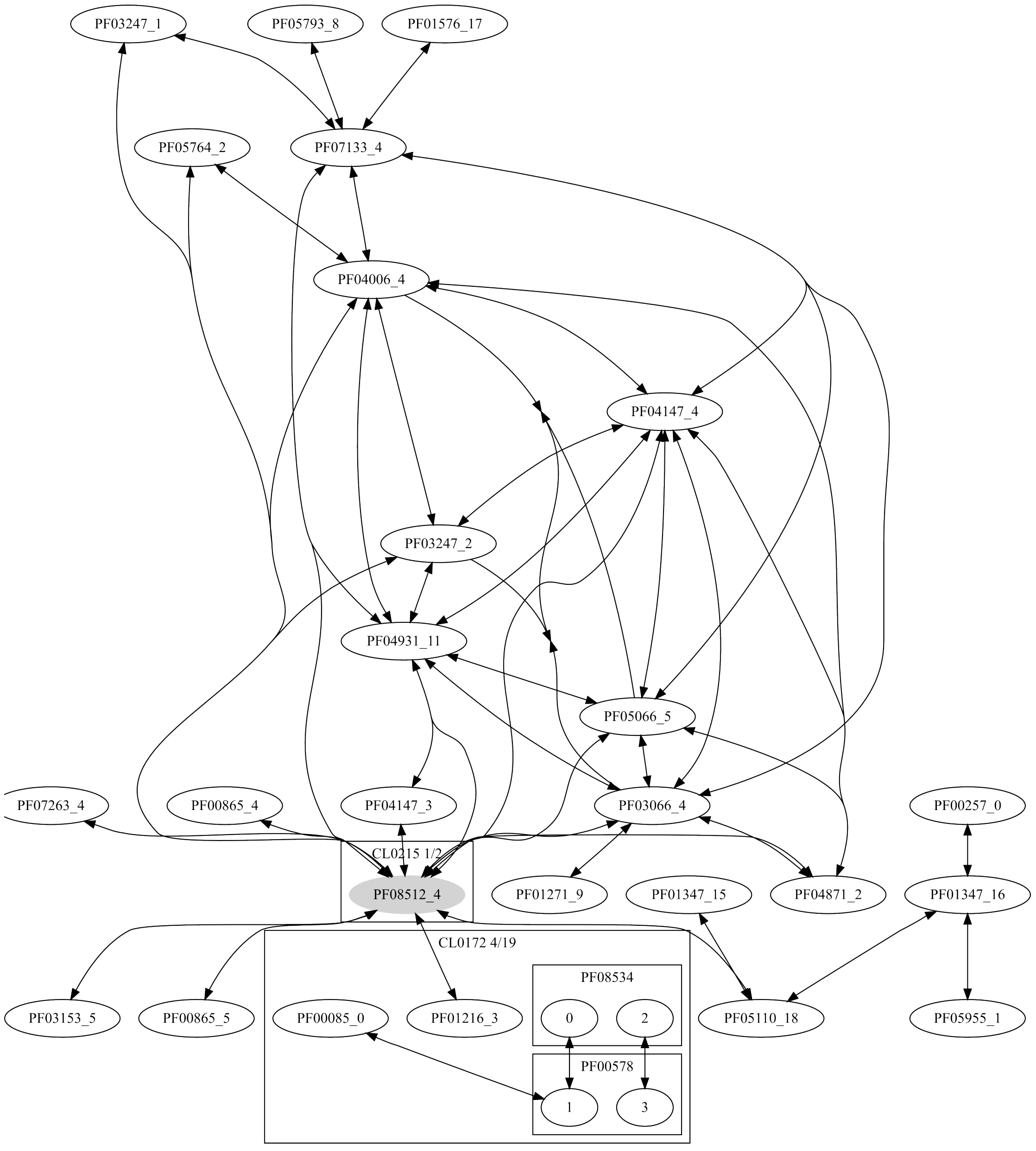


Figure 5.8: An Example is given for a novel Pfam domain cluster that could be predicted by the Sub-HMM method. The dark nodes indicates a domain of unknown function (DUFs). The other symbols in the graph are explained in the legend of Figure 5.7.

sub-DKFs will be useful for assigning potential functions to DUFs.

5.7 Conclusion

We have developed a simple but effective method for identifying the most highly conserved residues in protein sequences in a fully automated manner. Its design strategy is highly practical and versatile by making efficient use of a well-established bioinformatic infrastructure, such as existing domain databases and profile HMM search tools. In addition, the conserved patterns, identified by this study, are useful for characterizing proteins of unknown function by associating them with those of known function by their common sub-HMMs. Furthermore, the sub-HMM search method appears to be a very effective tool for finding sequences that share only very short sequence similarities with a sensitivity performance similar to HMMER2. The possibility to ignore the order of different sub-HMM matches in sequences is another advantage, which will allow the identification of more complex similarity arrangements among otherwise unrelated sequences.

Chapter 6

Sub-HMM Clustering

Sub-HMMs offer additional insight into the most important sub-sections of a more general family HMM, but they may not be the best representation of the true, defining patterns of a given family of protein sequences. Models representing extracted sub-HMMs where forced to be built on sequence fragments occurring at a specific location in the sequence which is dictated by the structure of the family HMM. Each fragment from a sequence used to build the extracted region of the model may not be the best fragment among all possible fragments on a given sequence however. Further, a family may be properly defined by a set of sub-HMMs, but an individual sub-HMM might also occur within other families. By finding these links, we can take advantage of the additional sequence data to improve the quality of models built on small families. In this chapter we demonstrate methods to address both of these concerns and show that search performance can be improved by taking advantage of them.

The first problem is addressed through a process we call retraining, in which the sub-HMM is allowed to match anywhere on each sequence and a new model is trained from the resulting fragments. This processes is iterated until no more movement occurs. See section 6.2 for details. The second problem is really a specific case of a more general problem of clustering HMMs, sub-HMMs or otherwise, in order to identify similar patterns. The sequence data associated with each model in a sub-HMM cluster can then be combined to train a new, more robust, model to replace all members of the cluster. Additionally, clustering will reduce the total number of unique sub-HMMs, which will make functional analysis easier and may also reveal interesting links between families. An in depth functional analysis is not pursued in this work however.

6.1 Notation

We first introduce some notation. Let $\mathcal{F}_j$ be the set of sequences associated with the jth family. Let A_{ij} be the matrix of scores of sub-HMM M_i scored on the jth family. That is,

$$A_{ij} = \sum_{S \in \mathcal{F}_j} \mathrm{lo}(M_i, S) \tag{6.1}$$

where lo() is the log odds score. Let $f(i)$ be the index of the family that M_i was extracted from. Finally, c_i is the complexity of sub-HMM M_i.

6.2 Retraining

This process improves the quality of individual sub-HMMs. For each sub-HMM, M_i, we find the matching fragment on each $S \in \mathcal{F}_{f(i)}$ and create a new alignment from these fragments by using `hmmalign` to align the fragments back to M_i. From this new alignment, a new model M_i' is trained and the process is iterated until the set of extracted fragments in one step is identical to the set of fragments in the previous step, or some maximum number of iterations is reached. This process allows extracted sub-HMMs to find their best matching location, which may be different than the location they were forced into by the original HMM. This is in line with our hypothesis that the order and location of sub-HMMs is not as important as the presence of a certain set of sub-HMMs.

We found that there are many cases where the sub-HMM matches some sequences with a poor score, and the resulting sequence fragments then corrupt the alignment, and thus the new model. To avoid this, we weight each sequence by the probability that the given sub-HMM is really present in the given sequence. That is, we compute $P(D_{ij}|S_j)$, where D_{ij} is a hidden variable which indicates whether or not M_i is present in S_j.. Ideally, we would then use these probabilities for each sequence as weights and tell the alignment program to treat sequences with lower weights as less important, and vice versa. We where unable to find an alignment algorithm which did this satisfactorily however, so we ignored these weights during the alignment step. The model builder though was able to use these weights to build a model which was more heavily influenced by sequences with larger weights and less corrupted by poor scoring, low weight sequences. Our results showed that this was still helpful, despite having a possibly corrupted alignment.

6.2.1 Estimating the Contribution of Each Sub-HMM

In the retraining step we want to compute

$$
\begin{aligned}
P(D_{ij}|S_j) &= \frac{P(S_j|D_{ij},M_i)P(D_{ij})}{P(S_j|D_{ij},M_i)P(D_{ij}) + P(S_j|\neg D_{ij},M_i)P(\neg D_{ij})} && (6.2)\\
&= \frac{\frac{P(S_j|D_{ij},M_i)}{P(S_j|\neg D_{ij},M_i)}P(D_{ij})}{\frac{P(S_j|D_{ij},M_i)}{P(S_j|\neg D_{ij},M_i)}P(D_{ij}) + P(\neg D_{ij})} && (6.3)\\
&= \frac{2^{\mathrm{lo}(M_i,S_j)}P(D_{ij})}{2^{\mathrm{lo}(M_i,S_j)}P(D_{ij}) + (1-P(D_{ij}))}. && (6.4)
\end{aligned}
$$

In order to compute this value, we need to know $P(D_{ij})$. We estimate this value by using EM in addition to the retraining procedure. We initialize $P(D_{ij})$ to be 0.75, then the first step of retraining computes the weights for each sequence fragment. Given these weights, we can re-estimate $P(D_{ij})$:

$$
\begin{aligned}
P(D_{ij}) &= \sum_{S_j \in \mathcal{F}_{f(i)}} P(D_{ij}|S_j)P(S_j) && (6.5)\\
&= \sum_{S_j \in \mathcal{F}_{f(i)}} P(D_{ij}|S_j)\frac{1}{|\mathcal{F}_{f(i)}|} \quad \text{assuming each } S_j \text{ is equally likely} && (6.6)\\
&= \frac{1}{|\mathcal{F}_{f(i)}|}\sum_{S_j \in \mathcal{F}_{f(i)}} P(D_{ij}|S_j) && (6.7)
\end{aligned}
$$

This new value is then used in the next step of retraining.

6.3 Clustering

Clustering the sub-HMMs would first reduce the number of sub-HMMs to a more manageable number. Secondly, it would reduce the number of parameters while increasing the amount of data available per parameter for estimation, which should lead to better generalization of the model. Clustering the sub-HMMs would also provide evidence that cluster members are functionally related.

There are several different approaches to clustering sub-HMMs. The most straightforward is to first compute some distance between each pair of sub-HMMs and then use a standard clustering algorithm. The distance function could be based on comparing models directly, or indirectly, by looking at the strings they match. However, since what we really want is to improve the search performance while reducing the total model complexity, it makes sense to optimize this directly by using the Minimum Description Length (MDL) criterion (Grunwald, 2005). This score provides the optimal way to trade-off between performance and complexity.

6.3.1 Distance Based Clustering

6.3.1.1 Comparing Models

One way to cluster sub-HMMs is to compute some distance score between sub-HMMs, using only the models themselves. We used the PRC (Madera, 2008) tool for this purpose. This tool creates an alignment between two profile HMMs and then computes a log odds score for the alignment. It currently only works reliably for local-local alignments, although we would prefer to use a global-global alignment. It also produces asymmetrical scores which we overcome by summing the scores from both directions.

Because the alignments from PRC are local-local, it is often the case that a short sub-HMM can match to a very long sub-HMM, even though most positions are quite different. This creates a problem when aligning extracted fragments from sub-HMMs of diverse lengths. The fragments themselves are often of very diverse lengths, and the alignments are very poor. Because of this problem, we did not pursue this method any further.

6.3.1.2 Comparing Matches

A second way to compare sub-HMMs is to compute a distance score based on which sequences the models matched. If two models tend to match the same strings, then they are more similar than two models which rarely match the same strings. This method better simulates a global-global comparison between sub-HMMs since we are looking at complete matches, so it thus avoids the alignment problem caused by PRC. However, it requires matching every sub-HMM on every sequence, which is a very time consuming process, and the time required will grow in proportion to the size of the dataset.

For each model, we score it on every sequence in the database and record the start position, the length of the match, and the score. This gives a vector of triples for each model:

$$S_k = \{(s_1, l_1, c_1), \ldots, (s_i, l_i, c_i), \ldots, (s_n, l_n, c_n)\}$$

where S_k stands for the kth model and $S_{k_{s_i}}$, $S_{k_{l_i}}$, and $S_{k_{c_i}}$ stand for the start, length, and score, respectively, of the kth model matched on the ith sequence of the database. These vectors are then used to compute distances between models. The different functions we tried are:

- Euclidean

$$d_E(k,l) = \sum_{i=1}^{n} \sqrt{(S_{k_{s_i}} - S_{l_{s_i}})^2 + (S_{k_{l_i}} - S_{l_{l_i}})^2 + (S_{k_{c_i}} - S_{l_{c_i}})^2}$$

- Location

$$d_L(k,l) = \sum_{i=1}^{n} \sqrt{(S_{k_{s_i}} - S_{l_{s_i}})^2 + (S_{k_{l_i}} - S_{l_{l_i}})^2}$$

- Length Normalized

$$\text{Norm}(i,j) = \left(\frac{s_i}{\max(l_i,l_j)}, \frac{l_i}{\max(l_i,l_j)}, \frac{c_i}{l_i + l_j}\right)$$

$$d_{\text{LN}}(k,l) = \sum_{i=1}^{n} d_E\left(\text{Norm}(S_k,S_l), \text{Norm}(S_l,S_k)\right)$$

- Family Limited Location

$$d_{\text{FL}}(k,l) = \sum_{i \in \mathcal{F}_{f(k)} \cup \mathcal{F}_{f(l)}} \sqrt{(S_{k_{s_i}} - S_{l_{s_i}})^2 + (S_{k_{l_i}} - S_{l_{l_i}})^2}$$

- Family Limited Overlap Count

$$d_{\text{FLO}}(k,l) = \frac{\sum_{i \in \mathcal{F}_{f(k)}} \mathcal{I}(\neg\text{overlaps}(k,l,0.5))}{|\mathcal{F}_{f(k)}|} + \frac{\sum_{i \in \mathcal{F}_{f(l)}} \mathcal{I}(\neg\text{overlaps}(k,l,0.5)}{|\mathcal{F}_{f(l)}|}$$

d_E is of course just the sum of Euclidean distances in three dimensional space. d_L is the sum of Euclidean distances in two dimensional space, where we just ignore the score dimension. For d_{NL}, we first normalize each triple and then just use d_E. The length of the longest sequence in scaled to 1 and the start and length values are rescaled to match. The score is set to the score per unit length of both sequences. These modifications should remove the effect of the sequence length on the distance value.

d_{FL} is just d_L in which we only consider sequences belonging the family of either model. Here, $\mathcal{F}_k$ is the set of sequence indices in the family from which sub-HMM k was extracted. This function effectively compares the performance of one model on the sequences of the model which it may be merged with, thus removing any noise caused by other sequences. Ideally, two similar models should perform similarly on any sequence, for better or worse, but in practice, we found that this is not always the case.

d_{FLO} is a stricter version of d_{FL} in which we count the number of times an overlap of 50% or more does not occur in a sequence, for each sequence in each models family. This value is divided by the size of the family to remove any dependence on the family size.

6.3.1.3 Cluster Algorithm

Given a distance matrix between all pairs of sub-HMMs, we used a hierarchical agglomerative complete linkage clustering algorithm (Johnson, 1967) to cluster the models. We then cut the tree at different heights to produce clusterings of different sizes.

6.3.1.4 Merging

Given a cluster of sub-HMMs, we want to create a new sub-HMM which summarizes all the cluster members. For each sub-HMM in a cluster, we extract the matching protein fragments from that sub-HMMs family members. These fragments are then aligned, weighted and used to build a new HMM, as described in section 6.2.1. The same iterative process is used, with one exception. On the first iteration, we cannot use `hmmalign` because we don't have one unique HMM that all fragments came from. Instead, we use clustalw to create the first alignment. On subsequent iterations, `hmmalign` is used with the HMM from the previous iteration.

6.3.2 MDL Based Clustering

In this method we combine the merging step with the clustering step, so models are merged as we go. Having the merged model at intermediate steps provides more information about which merges will produce the best result. It is based on a hierarchical clustering (HC) algorithm. The ideal algorithm would start off computing the improvement in MDL score obtained by merging, for every pair of sub-HMMs. It would then merge the pair with the largest improvement and then update the MDL improvement values between this new model and all other models. At some point all the improvements would be negative, indicating that merging any pair of models will make the overall MDL score worse, so we stop at that point. This provides a natural clustering, unlike normal HC, which will continue merging things until it runs out of things to merge.

The MDL score balances the likelihood of a model against is complexity. Given a model M_i, trained on n sequences and having N_p free parameters, we have:

$$\begin{aligned} \mathrm{MDL}(M_i, \mathcal{F}_{f(i)}) &= \sum_{S \in \mathcal{F}_{f(i)}} lo(M_i, S) - \frac{1}{2} N_p \lg n && (6.8) \\ &= A_{if(i)} - c_i. && (6.9) \end{aligned}$$

The complexity term here is for the most general case. For any model with N_p parameters which need to be estimated, the count for each parameter cannot exceed the number of data points, n, so we can encode each count in $\lg(n)$ bits. In the case of profile HMMs, this complexity term overestimates the true complexity in most models. Many of the distributions in an HMM have no support from the data so we end up with a large number of 0 counts. To encode these counts efficiently we used arithmetic encoding.

In arithmetic encoding, we first compute the empirical distribution of the counts to be encoded, then we used this distribution to encode the counts. The best compression rate is achieved when the computed distribution closely matches the actual counts. In the case of profile HMMs, we found that the best performance was achieved by using 9 different count distributions, one for insert counts, one for match counts, and 7 more for the 7 different transition types. Each of these model sections can have very different count distributions. For example, empirically, it is common to have very little support for the insert emission distributions at most positions, resulting in a large number of 0 counts. In the match emission distributions however, we have much more support from the data so we see a very wide range of counts. Similarly, some transitions occur very seldom and so have more 0 counts than is seen in other transition distributions. By using a different count distribution for each of these sections, we ensure that each one fits the data as close as possible. This gives us the best compression rate for encoding the counts. However, we must also encode the count distributions themselves, and encoding 9 distributions instead of just 1 will certainly add to the complexity. In practice we found that using 9 count distributions was still a net gain in the compression rate. Our final complexity term is

$$c_i = \sum_{s \in \mathrm{SEC}} \left(2\lg(M_s) + \sum_{j=0}^{M_s} 2\lg(\#_s(j)) + \sum_{j=0}^{N_s} \lg(p_s(n_j)) \right) \tag{6.10}$$

where SEC is the set of 9 sections described above, N_s is the number of counts in that section that need to be encoded, and M_s is the index of the last non-zero value in p_s.

The difference between the complexity term for the general case, and the complexity we compute with arithmetic encoding will shrink as the amount of data available to train the model on grows. At the same time, the log odds term will start to dominate the complexity term as the amount of data grows. Thus, computing a more accurate complexity value is most beneficial for small families. We also tried a simpler encoding scheme that sends just 1 bit if a distribution has no support, and sends

$$1 + \sum_{i=1}^{20} \lg \left(\frac{k_i}{\sqrt{w}} + 1 \right)$$

bits for supported distributions. Here, k_i is the count and w is the sum of the weights of each sequence in the data set. This proved to be inferior to arithmetic encoding in almost every case though.

The improvement in MDL of the merged model is computed as:

$$\mathrm{MDL}(M_{i,j}, \mathcal{F}_i \cup \mathcal{F}_j) - (\mathrm{MDL}(M_i, \mathcal{F}_i) + \mathrm{MDL}(M_j, \mathcal{F}_j)) \tag{6.11}$$

where $M_{i,j}$ is the model obtained by merging M_i and M_j. This is the difference in MDL score between using the two original models to score the sequences from their respective families, and using the merged model to score the union of sequences. The two original models should have a higher likelihood since they are more specific to their respective sets of sequences, but the total complexity will be higher since there are two models. The merged model may have a lower likelihood, but also a lower complexity. If the two original models where sufficiently similar, the likelihood of the merged model should not decrease too much, resulting in an higher MDL score.

The problem with the ideal algorithm is that computing the merged model between every pair of models is too expensive. To merge two models, say M_i and M_j, they must first both be scored against each sequence in their respective families. Then an alignment of all the resulting fragments must be created, and a new model built. Finally, this process is iterated a small number of times to let the model find the best final position. A single merge takes $O(nTN)$ time, where n is the number of sequences to be scored from both families, T is the average sequence length, and N is the average model length. Then we must perform $O(|Y|^2)$ merges. In our case, $|Y| \approx 1000$, $n \approx 100$, $T \approx 300$, and $N \approx 17$. This gives us around 10^{11} operations, which is not practical.

6.3.2.1 MDL Exemplar

We thus tried a few simplifications of the ideal algorithm. The first is called MDL Exemplar. In this algorithm, instead of creating a merged model for each pair, we choose one member of the pair, the exemplar, and treat it as if it was the merged model. That is, this one exemplar is used to score the sequences from both respective families. We let $f(k)$ be the set of family indices that model k currently represents, which will change as the algorithm progresses. It is initialized so that each model represents the family it was extracted from. Then the improvement of using M_i as the exemplar can be computed as

$$\begin{aligned}
&\mathrm{MDL}(M_i, \mathcal{F}_{f(i)} \cup \mathcal{F}_{f(j)}) - \big(\mathrm{MDL}(M_i, \mathcal{F}_{f(i)}) + \mathrm{MDL}(M_j, \mathcal{F}_{f(j)})\big) && (6.12)\\
&= \big(A_{if(i)} + A_{if(j)} - c_i\big) - \big((A_{if(i)} - c_i) + (A_{jf(j)} - c_j)\big) && (6.13)\\
&= A_{if(j)} - A_{jf(j)} + c_j && (6.14)
\end{aligned}$$

The final improvement used is the maximum of both exemplars:

$$\max(A_{if(j)} - A_{jf(j)} + c_j, A_{jf(i)} - A_{if(i)} + c_i) \qquad (6.15)$$

If M_i is chosen as the exemplar then we set $f(i) = f(i) \cup f(j)$. At the end, each cluster is represented by one of its members.

One thing to note is that the likelihood of an exemplar, trained on family $f(i)$, on $\mathcal{F}_{f(i)} \cup \mathcal{F}_{f(j)}$ should be smaller than a merged model, trained on $\mathcal{F}_{f(i)} \cup \mathcal{F}_{f(j)}$, on the same set. This is because the merged model has strictly more information about the sequences it will be later scored on. We can only say 'should' however, since the quality of the alignment will have a large effect on the model that is built and the addition of sequences $\mathcal{F}_{f(j)}$ can change the alignment substantially. The other thing to look at is the complexity of the merged model, M_{ij}. In the worst case, the alignments of i and j could be concatenated lengthwise resulting in a merged model whose length is the sum of the lengths of M_i and M_j. The complexity of M_{ij} in this case would be approximately $c_i + c_j$. This is highly unlikely though, and was never observed in our experiments. Generally, we can assume that the complexity of M_{ij} will be less than $c_i + c_j$. If the complexity of M_{ij} is less than $\min(c_i, c_j)$ than we would have that

$$\mathrm{MDL}(M_{ij}, D_{ij}) < \min\left(\mathrm{MDL}(M_i, D_{ij}), \mathrm{MDL}(M_j, D_{ij})\right)$$

where $D_{ij} = \mathcal{F}_{f(i)} \cup \mathcal{F}_{f(j)}$. We can use this fact to decide before hand if merging M_i and M_j will improve the MDL score. Without any prior knowledge, we can say that it is likely that the complexity term will improve in the merged model, but it is not clear what will happen to the likelihood without already knowing how similar the models are to each other. Thus, we cannot say much about the resulting MDL score either. However, if we know that one of the models performs well as an exemplar, then the above inequality tells us that the merged model will improve the MDL score under the given conditions. If neither exemplar has a better likelihood than that obtained using M_i and M_j, then we can't say anything about the performance of the merged model.

6.3.2.2 MDL

We can thus use the MDL Exemplar improvement score as a lower bound on the improvement obtained by using a merged model, in most cases. This leads to another version of the above algorithm. It starts off the same, but each time it choses a pair to merge, it creates a merged model and then keeps the model with the best MDL score, among the exemplars and the merged model. This model is then used in all future improvement calculations, which still use the MDL Exemplar method to compute improvements. Thus, the expensive merge step is only performed when we know it can improve things, instead of speculatively merging things.

Since there are still cases where the complexity of the merged model is sufficiently bad to offset the improvement in the likelihood, we would like to find a way to minimize the number of these cases. We found that most of these cases are caused by the model builder building a model that is too long. This is because the default HMMER2 model builder uses a different complexity term when it is deciding which columns of the alignment should be marked as match states. By modifying this complexity term to match what we use in our MDL score, we were able to reduce the frequency with which this problem occurs. See 6.3.2.3 for details.

At the end of this algorithm, each cluster is represented by a new sub-HMM which is the result of merging the cluster members together. This merged model has an advantage over the cluster merging done in section 6.3.1.4 in that the merging can take advantage of the clusters structure. In the unstructured case, fragments from each cluster member are aligned together and a new model is build from that. But multiple alignment algorithms can be very sensitive to the order in which things are aligned, and in the unstructured case we don't have any special information about the best order. In the structured case though, the order in which things are merged, as the algorithm runs, is also the best order to align the alignments associated with each pair in the merge. This results in the most similar sequences being aligned together first, which leads to a better overall alignment and thus a better model can be build from it. We used muscle (Edgar, 2004) to perform the alignment alignment alignment.

6.3.2.3 MDL Model Builder

The default HMMER2 model uses a dynamic programming method to decide which columns of the given alignment should correspond to match states in the resulting HMM. Once these columns are marked, the rest of the model construction process is trivial. The goal of this method is to optimize the likelihood of the data. Since this alone would result in marking every column, a penalty factor is included for the addition of each match state. This method works fine in general, however, it is not optimizing the MDL criteria that the rest of our method is trying to optimize. In particular, the penalty term used by HMMER2 does not consider the complexity of the model. To address this issue we implemented a new model builder which optimizes the MDL score directly.

The main challenge in this is estimating the resulting model complexity while making use of only column local data so that we can maintain the efficiency of dynamic programming. To do this we use arithmetic encoding, described above, to encode the counts of each column. Doing this however, requires that we already have a distribution over counts. We therefore create an initial model using the "fast" algorithm of HMMER2. In this algorithm, any column in which more than half of the symbols in that column are non-gaps is marked as a match position. We then use this marking to compute the distribution of counts which will then be used for the arithmetic encoding during the final model building step. The complexity for each column is then computed based on the size of the arithmetic encoding of the counts in that, as described above.

6.3.3 Sub-HMM Match Discovery

Given the complexity and the amount of time required to run the above MDL clustering algorithm we decided to develop a vastly simplified method with the goal of simply finding a few high quality sub-HMM to sub-HMM matches. This could produce the most important clusters relatively quickly. This method could also take advantage of the pre-filtering techniques of HMMER3 in order to vastly speed the rate at which sub-HMMs could be scored against the Pfam 24 database. The method works as follows. Every Sub-HMM is scored on every sequence in a database and only those matches which met a very strict score cutoff were kept. Pairs of sub-HMMs matching the same sequence were then clustered. We then use an MDL test on each cluster to see if merging the member sub-HMMs would result in some improvement or not.

6.3.3.1 Match Criteria

This test was only performed on PFAM 24 using HMMER3. The match criteria we used was the default HMMER3 criteria, that is, what ever matches HMMER3 reports with the default settings. Each sub-HMM was scored on every sequence and any reported matches were noted as a link between the sub-HMM and every family the matched sequence was a member of. We then look for cases where sub-HMMs from two different families match a sequence, or some fraction of sequences, of the same family. A match in this case means that, for each sequence, the two fragments matched by the sub-HMMs overlap by at least 10%. All such pairs are assigned a distance of 0 and all other pairs have a distance of 1. A standard hierarchical complete linkage clustering is then used to cluster the sub-HMMs.

To make the match test more strict, we used two different criteria. First is to require that $x\%$ of the family sequences be matched by both sub-HMMs. The second is that the fragments matched fall in similar locations on each sequence, according to the original family alignment. To measure this second criteria, we compute for each column of the alignment, the fraction of sequences which are part of the fragment. This is the entropy of that column. We then sum the entropies for each column and divide by the number of columns in the alignment. This gives us a number between 0 and 1, with 0 indicating a better alignment of the fragments. Let

$$A_{ij} = \begin{cases} 0 & \text{if position } i \text{ of sequence } j \text{ does not match} \\ 1 & \text{if position } i \text{ of sequence } j \text{ does match} \end{cases} \tag{6.16}$$

and let

$$p_i = \frac{1}{N_S} \sum_{j=0}^{N_A} A_{ij},$$

where N_A is the length of the alignment and N_S is the number of sequences in the alignment. Then the final entropy is

$$H = \frac{1}{N_A} \sum_{i=0}^{N_A} -p_i \lg p_i - (1 - p_i) \lg (1 - p_i).$$

6.3.3.2 MDL Test

We used an MDL test similar to that described above, but generalized for n sub-HMMs instead of just 2. Each member sub-HMM of a cluster C is used to extract its matching fragment from the sequences of a family. Then the MDL score is computed for the sub-HMM on those fragments. The sum of these scores is then compared to the score of the merged model M_C. If the merged model has a larger score then we accept the cluster as helpful. The final improvement is computed as

$$\mathrm{MDL}(M_C, \bigcup_{i \in C} \mathcal{F}_i) - \sum_{i \in C} \mathrm{MDL}(M_i, \mathcal{F}_i). \tag{6.17}$$

6.4 Results

We tested the search performance of retraining alone, as well as each different clustering method proposed above. The general test setup for these two tests was the same as that described in Section 5.4, with differences noted in each section below. We also tested our sub-HMM match discovery process on the entire PFAM 24 dataset, using HMMER3.

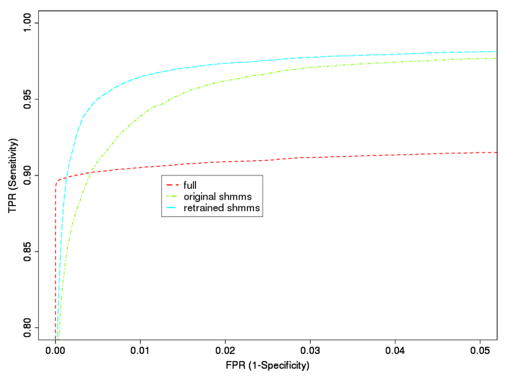
6.4.1 Improvements From Retraining

We found in practice that the retraining step alone can sometimes result in significant improvements in search performance. In Figure 6.1 we can see the results of retraining on several different datasets. In some cases it improves the performance, but in some cases it does not. Using HMMER2 on the small set we see that using the original sub-HMMs, we are able to improve the sensitivity somewhat, but the refinement procedure improves it even more, by about 5% at a specificity of 0.995. The small set on HMMER3 however does about 1% worse after retraining, while the large set does about 7% better at the same specificity. In general retraining was a useful thing to do, even when it doesn't improve things, it doesn't make them significantly worse either.

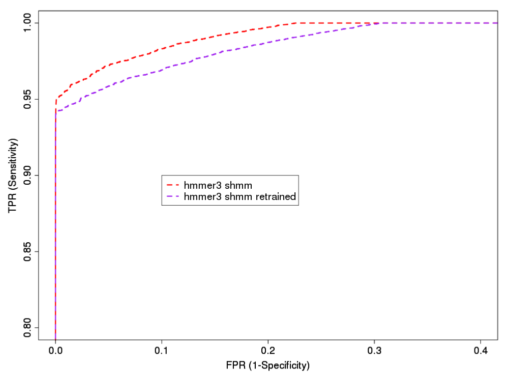
6.4.2 Sub-HMM Clustering

Each clustering method was used to cluster the set of sub-HMMs extracted from the PFAM 22 small set, about 4,953 sub-HMMs. For the Hierarchical Clustering (HC) clusterings, each cluster was used to create a single, merged representative model for that cluster, and all the cluster members were replaced with this representative. For the MDL Exemplar and MDL methods, the representative is output by the algorithm, so that is used in place of the cluster members. Then we use this new set of sub-HMMs to perform the search performance test, in the same way as described in Section 5.4. The results were compared against using the original, retrained sub-HMMs, without any clustering. We tested several variations of each clustering method. Figure 6.2 shows the best result from each method. For the HC methods, we could cut the cluster tree at different levels to create differing numbers of clusters. We found that about 3000 clusters produced the best trade off between fewer models, and good performance. As the number of clusters decreased, the performance also decreased.

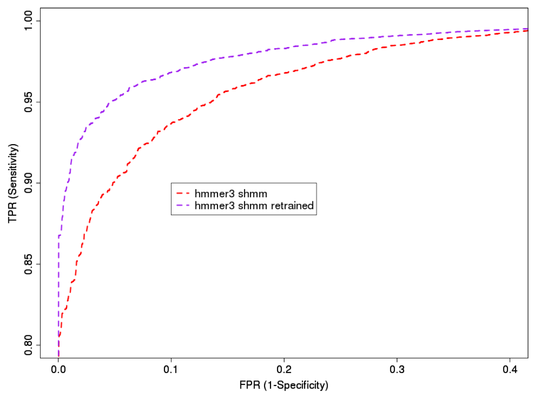
We can see that Euclid and Location work the best and about the same as each other. Limiting the comparison to just the families of the corresponding sub-HMMs, that is, the Family Limited methods, caused a higher false positive rate at first, but they met or exceeded the other methods after the false positive rate exceeded about 0.1. The MDL method performed reasonably well compared to the HC methods, though it



(a) HMMER2 Small



(b) HMMER3 Small



(c) HMMER3 Large

Figure 6.1: The improvement of retrained sub-HMMs over original sub-HMMs and full HMMs.

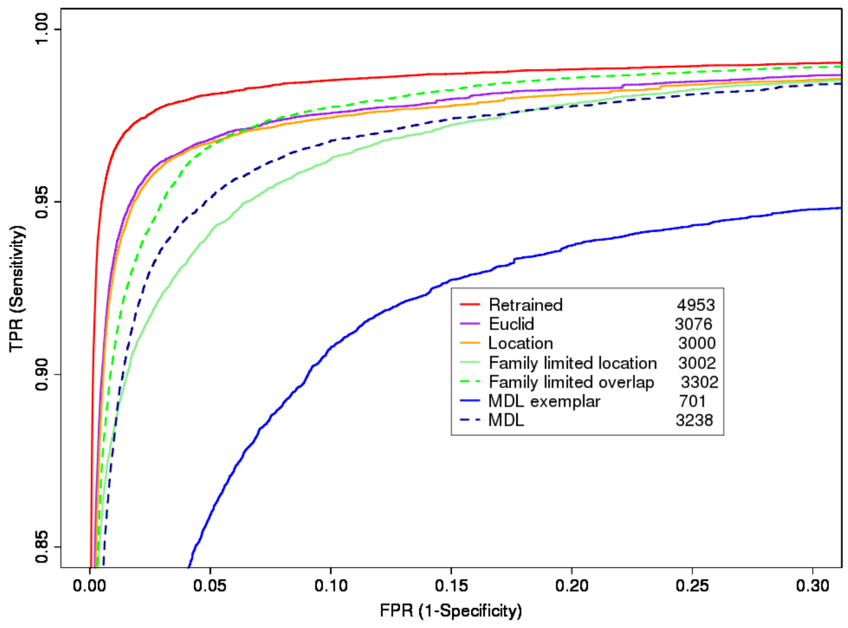


Figure 6.2: ROC curves of the search performance comparison of different clustering methods on the PFAM 22 small set. The number of clusters is given for each method.

used 200 more clusters and was still dominated by Euclid and Location. For the MDL methods, the number of clusters is determined by the algorithm. The MDL Exemplar method chose a rather small number of clusters, which led to its poor performance, although, for that few clusters, the performance is actually quite good. It was able to reduce the number of parameters needed by about 80%.

Overall, non of these methods did better than not performing any clustering at all, so in terms of search performance, they were not helpful. However, they did succeed in reducing the number of models and thus parameters needed by about 40% in most cases, with only a minor reduction in search performance. This has benefits in terms of speed and also in identifying groups of similar sub-HMMs, which is of interest on its own.

6.4.3 Match Discovery

In this test we first extracted 55,265 sub-HMM's from PFAM 24. We then used HMMER3 to score every sub-HMM against every protein in PFAM 24. The sub-HMMs were then clustered and merged, as described above. If we require at least 30% of family members to match and an alignment entropy less than 0.5, we found 18 matching pairs, 13 of which where considered helpful merges according to the MDL test. Relaxing the

% Family	Max Entropy	# Clusters	# Passed MDL	% Passed	# U-K
30%	0.5	18	13	72%	3
20%	0.5	30	22	73%	6
$> 0\%$	0.5	285	177	62%	47
$> 0\%$	1	391	254	64%	59

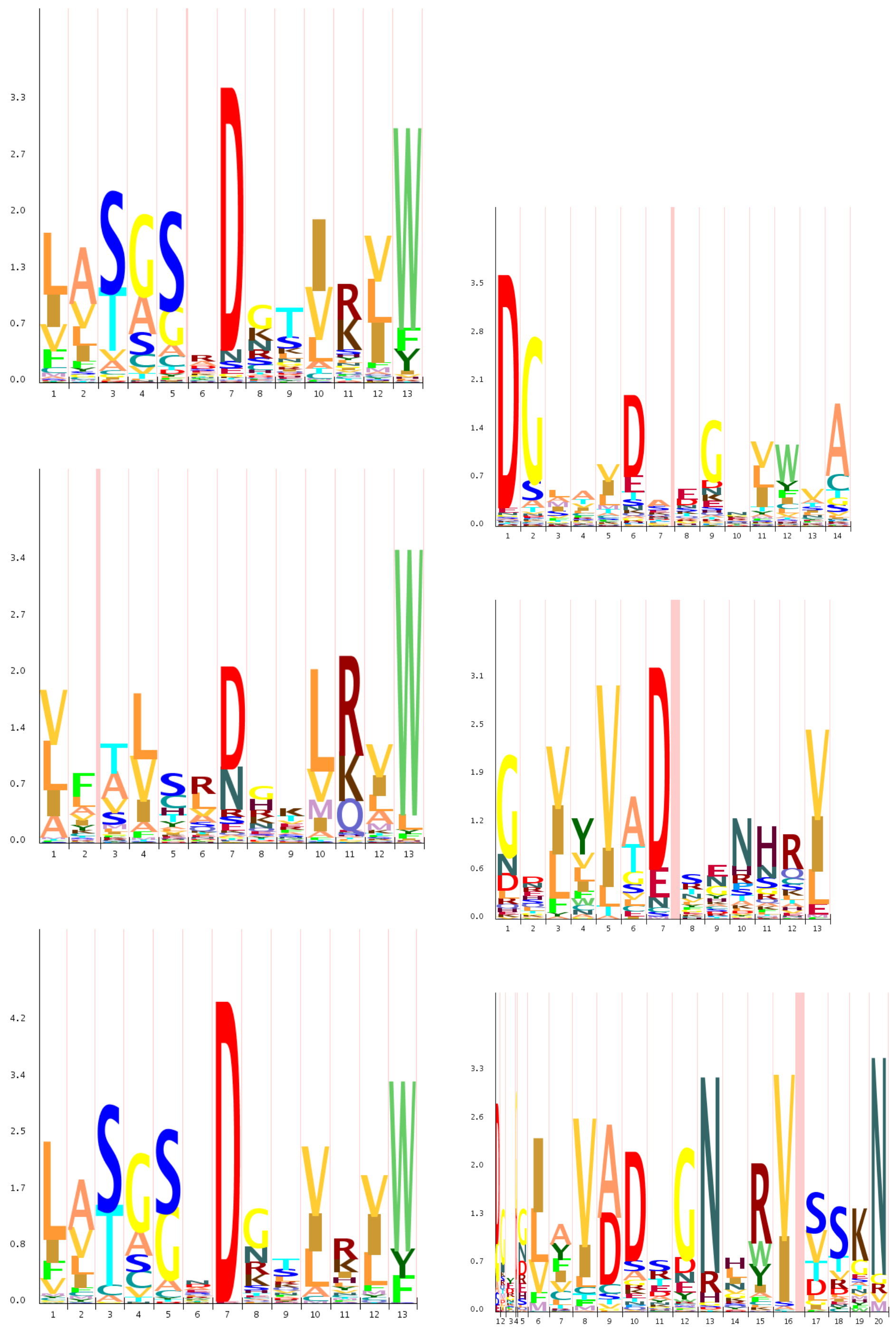
Table 6.1: Number of clusters found at different filter strictnesses and how many of them passed the MDL test. The first column is the fraction of the family members that must match. The last column is the number of clusters containing both known and unknown domains.

filter to 20% of family members, we found 30 clusters, of which 22 where beneficial. If we require only one sequence to match in a family, we find 285 clusters with 177 being helpful. Ignoring the filters all together, we found 391 clusters, 254 of which improved the MDL score when merged. Figure 6.3 shows and example of a good pair and a bad pair, according to the MDL test.

From these results we can see that this method was effective in finding groups of sub-HMMs which are similar enough to benefit from being merged. Requiring at least 20% of family members to match a sub-HMM improved the enrichment of beneficial clusters from 60% to 70%, but also greatly reduced the total number of clusters found. Since the total number of clusters found without the filters is still a manageable number, the filter restrictions are probably not necessary. Also interesting to note is that removing the entropy restriction did not noticeably change the fraction of beneficial clusters (62% $\rightarrow$ 64%), and thus may not be as important a factor. We also found that many clusters contained sub-HMMs from both known and unknown families, which indicates a potential functional link between these families. See Table 6.1.

6.5 Conclusion

We developed and tested several ways of improving individual sub-HMMs and clustering similar sub-HMMs together. We found that allowing variable placement of each sub-HMM, through retraining, produced a more accurate representation of the motif being modeled. This was shown through the improved search performance in most cases, when using retrained sub-HMMs. We developed several clustering methods in order to group similar sub-HMMs together. We found that a simple HC method based on the Euclidean distance of start, length pairs for each sub-HMM performed the best. The MDL clustering method was also able to create competitive clusterings, but at a



(a) Good Merge (b) Bad Merge

Figure 6.3: An example of merging two sub-HMMs. The top two logos are the cluster members and the bottom logo is the merged model. The MDL score indicates that (a), PF00400.25-0 and PF11715.1-4, is a good merge, while (b), PF08450.5-3 and PF01436.14-0, is a bad merge. Visually, this makes sense.

greater computational cost. The method was effective in deciding when a merge was helpful however. The test for matching up highly similar sub-HMMs was also effective in finding high quality clusters and the MDL test was able to filter out those clusters which would not be worth merging, from a search performance standpoint. These clusters, many of which contained both known and unknown members, may indicate additional relationships between families.

Chapter 7

Conclusion

In Chapter 3 we first establish that almost half of the protein coding genes in Arabidopsis are unknowns, at the time of the study, and explore ways of automatically assigning functions based on comparisons to other known genes. We present a process for assigning functions to unknown proteins in Arabidopsis and assign 1,541 previously unknown genes a function. The data used in this study was made available for searching through an online website.

Chapter 4 presents a model for clustering genes based on diverse data types and which could also learn novel motifs in the promoter regions of genes. Motif detection was done first with a simple greedy search among motif sequences, and then improved upon by using a Bayesian model to represent the entire promoter sequence. This method proved effective in finding novel motifs in synthetic data. In addition, we added several improvements to the naive Bayes clustering algorithm to ensure the best use of each available cluster label. This ensures that if the user asks for k clusters, the algorithm actually uses all k of them.

An automated method for extracting the most highly conserved regions in a set of protein sequences was presented in Chapter 5. These extracted motifs, or sub-HMMs, where shown to be similar to many hand curated motifs in PROSITE and also covered many known active sites documented in CSA. Sub-HMMs were also shown to be effective in searching for family members among a large database of proteins, despite using only half the number of model parameters and allowing re-arrangements of the sub-HMMs. We were also able to find links between different PFAM families by searching for the occurrence of sub-HMMs in families other than their own. These links may represent some functional similarities between these families, which is of particular interest since many of these links were between known and unknown domains.

Finally, in Chapter 6 we developed and tested several ways of improving individual sub-HMMs and finding similar sub-HMMs. We found that allowing variable placement of each sub-HMM, through retraining, produced a more accurate representation of the motif being modeled. This was shown through the improved search performance in most cases, when using retrained sub-HMMs. We developed several clustering methods in order to group similar sub-HMMs together. We found that a simple HC method based on the Euclidean distance of start, length pairs for each sub-HMM performed the best. The MDL clustering method was also able to create competitive clusterings, but at a greater computational cost. The method was effective in deciding when a merge was helpful however. The test for matching up highly similar sub-HMMs was also effective in finding high quality clusters and the MDL test was able to filter out those clusters which would not be worth merging, from a search performance standpoint. These clusters may indicate additional relationships between families.

We have established that many genes still have an unknown function and that assigning functions in an automated way will become increasingly important as new genomes are sequenced. In this work we have developed several novel tools and methods of associating unknown genes with known genes, and thus providing hypothetical functions for these unknown genes.

Bibliography

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, Oct 1990. doi: 10.1006/jmbi.1990.9999. URL `http://www.hubmed.org/display.cgi?uids=2231712`.

S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, Sep 1997. URL `http://www.hubmed.org/display.cgi?uids=9254694`.

W. Ao, J. Gaudet, W. Kent, S. Muttumu, and S. Mango. Environmentally induced foregut remodeling by pha-4/foxa and daf-12/nhr. *Science*, 305(5691):1743, 2004.

M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

T. Bailey and C. Elkan. Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization. *Machine Learning*, 21(1):51–80, 1995.

Y. Barash and N. Friedman. Context-Specific Bayesian Clustering for Gene Expression Data. *Journal of Computational Biology*, 9(2):169–191, 2002.

T. Barrett, D. Troup, S. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. Kim, A. Soboleva, M. Tomashevsky, and R. Edgar. Ncbi geo: mining tens of millions of expression profilesdatabase and tools update. *Nucleic acids research*, 35(suppl 1): D760, 2006.

A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy. The Pfam protein families database. *Nucleic Acids Research*, 32(Database issue):138–141, Jan 2004. doi: 10.1093/nar/gkh121. URL `http://www.hubmed.org/display.cgi?uids=14681378`.

R. A. Becker, J. M. Chambers, and A. R. Wilks. *The New S Language.* Wadsworth and Brooks/Cole, Pacific Grove, CA, 1988.

O. Bembom, S. Keles, and M. J. van der Laan. Supervised detection of conserved motifs in dna sequences with cosmo. URL `http://www.bepress.com/cgi/viewcontent.cgi?article=1260&context=sagmb`.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate - a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289–300, 1995.

T. Z. Berardini, S. Mundodi, L. Reiser, E. Huala, M. Garcia-Hernandez, P. Zhang, L. A. Mueller, J. Yoon, A. Doyle, G. Lander, N. Moseyko, D. Yoo, I. Xu, B. Zoeckler, M. Montoya, N. Miller, D. Weems, and S. Y. Rhee. Functional annotation of the Arabidopsis genome using controlled vocabularies. *Plant Physiology*, 135(2):745–755, Jun 2004. doi: 10.1104/pp.104.040071. URL `http://www.hubmed.org/display.cgi?uids=15173566`.

B. Boeckmann, A. Bairoch, R. Apweiler, M. C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, Jan 2003. URL `http://www.hubmed.org/display.cgi?uids=12520024`.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 115–123. Citeseer, 1996.

E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J. M. Cherry, and G. Sherlock. GO::TermFinder–open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics*, 20(18):3710–3715, Dec 2004. doi: 10.1093/bioinformatics/bth456. URL `http://www.hubmed.org/display.cgi?uids=15297299`.

A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, T. Gaasterland, P. Glenisson, F. C. Holstege, I. F. Kim, V. Markowitz, J. C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nature Genetics*, 29(4):365–371, Dec 2001. doi: 10.1038/ng1201-365. URL `http://www.hubmed.org/display.cgi?uids=11726920`.

D. M. Brown, L. A. Zeef, J. Ellis, R. Goodacre, and S. R. Turner. Identification of novel genes in Arabidopsis involved in secondary cell wall formation using expression profiling and reverse genetics. *Plant Cell*, 17(8):2281–2295, Aug 2005. doi: 10.1105/tpc.105.031542. URL `http://www.hubmed.org/display.cgi?uids=15980264`.

M. J. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, Jun 2004. doi: 10.1093/bioinformatics/bth078. URL `http://www.hubmed.org/display.cgi?uids=14871861`.

A. Dempster, N. Laird, D. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

R. Durbin. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

S. R. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, 6(3):361–365, 1996. URL `http://www.hubmed.org/display.cgi?uids=8804822`.

S. R. Eddy. A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Computational Biology*, 4(5), May 2008. doi: 10.1371/journal.pcbi.1000069. URL `http://www.hubmed.org/display.cgi?uids=18516236`.

R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004. doi: 10.1093/nar/gkh340. URL `http://nar.oxfordjournals.org/content/32/5/1792.abstract`.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, Dec 1998. URL `http://www.hubmed.org/display.cgi?uids=9843981`.

S. Falcon and R. Gentleman. Using GOstats to test gene lists for GO term association. *Bioinformatics*, 23(2):257–258, Jan 2007. doi: 10.1093/bioinformatics/btl567. URL `http://www.hubmed.org/display.cgi?uids=17098774`.

R. D. Finn, J. Mistry, B. Schuster-Böckler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. Sonnhammer, and A. Bateman. Pfam: clans, web tools and services. *Nucleic Acids Research*, 34 (Database issue):247–251, 2006. doi: 10.1093/nar/gkj149. URL `http://www.hubmed.org/display.cgi?uids=16381856`.

R. D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. Sonnhammer, S. R. Eddy, and A. Bateman. The Pfam protein families database. *Nucleic Acids Research*, 38 (Database issue):211–222, Jan 2010. doi: 10.1093/nar/gkp985. URL `http://www.hubmed.org/display.cgi?uids=19920124`.

G. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

M. Frith, U. Hansen, J. Spouge, and Z. Weng. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Research*, 32(1):189, 2004.

T. Fukao and J. Bailey-Serres. Plant responses to hypoxia–is survival a balancing act? *Trends in Plant Science*, 9(9):449–456, Sep 2004. doi: 10.1016/j.tplants.2004.07.005. URL `http://www.hubmed.org/display.cgi?uids=15337495`.

C. M. Gachon, M. Langlois-Meurinne, Y. Henry, and P. Saindrenan. Transcriptional co-regulation of secondary metabolism enzymes in Arabidopsis: functional and evolutionary implications. *Plant Molecular Biology*, 58(2):229–245, May 2005. doi: 10.1007/s11103-005-5346-5. URL `http://www.hubmed.org/display.cgi?uids=16027976`.

A. Gasch and M. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol*, 3(11):1–22, 2002.

A. P. Gasch. *The Environmental Stress Response: a common yeast response to environmental stresses*, volume 1 of *Topics in Current Genetics (series editor S. Hohmann)*, pages 11–70. Springer Verlag, Heidelberg, 2002.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11(12):4241–4257, Dec 2000. URL `http://www.hubmed.org/display.cgi?uids=11102521`.

A. Gattiker, E. Gasteiger, and A. Bairoch. Scanprosite: a reference implementation of a prosite scanning tool. *Applied Bioinformatics*, 1(2):107–108, 2002. URL `http://www.hubmed.org/display.cgi?uids=15130850`.

R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, New York, 2005. URL `http://www.bioconductor.org/pub/docs/mogr`.

F. Gibbons and F. Roth. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Research*, 12(10):1574, 2002.

T. Girke, J. Lauricha, H. Tran, K. Keegstra, and N. Raikhel. The Cell Wall Navigator database. A systems-based approach to organism-unrestricted mining of protein families involved in cell wall metabolism. *Plant Physiology*, 136(2):3003–3008, Oct 2004. doi: 10.1104/pp.104.049965. URL `http://www.hubmed.org/display.cgi?uids=15489283`.

M. Gollery, J. Harper, J. Cushman, T. Mittler, T. Girke, J. K. Zhu, J. Bailey-Serres, and R. Mittler. What makes species unique? The contribution of proteins with obscure features. *Genome Biology*, 7(7):R57, Jul 2006. doi: 10.1186/gb-2006-7-7-r57. URL `http://www.hubmed.org/display.cgi?uids=16859532`.

M. Gollery, J. Harper, J. Cushman, T. Mittler, and R. Mittler. POFs: what we don't know can hurt us. *Trends in Plant Science*, page in press, 2007.

A. K. Grennan. Genevestigator. Facilitating web-based gene-expression analysis. *Plant Physiology*, 141(4):1164–1166, Aug 2006. doi: 10.1104/pp.104.900198. URL `http://www.hubmed.org/display.cgi?uids=16896229`.

W. Grundy, T. Bailey, C. Elkan, and M. Baker. Meta-MEME: motif-based hidden Markov models of biological sequences. *Computer Applications in the Biosciences*, 13:397–406, 1997.

P. Grunwald. A tutorial introduction to the minimum description length principle. *Advances in Minimum Description Length: Theory and Applications*, pages 23–81, 2005.

R. A. Gutiérrez, L. V. Lejay, A. Dean, F. Chiaromonte, D. E. Shasha, and G. M. Coruzzi. Qualitative network models and genome-wide expression data define carbon/nitrogen-responsive molecular machines in Arabidopsis. *Genome Biology*, 8(1):R7, 2007. doi: 10.1186/gb-2007-8-1-r7. URL `http://www.hubmed.org/display.cgi?uids=17217541`.

G. Haberer, M. T. Mader, P. Kosarev, M. Spannagl, L. Yang, and K. F. Mayer. Large-Scale cis-Element Detection by Analysis of Correlated Expression and Sequence Conservation between Arabidopsis and Brassica oleracea. *Plant Physiology*, 142(4):1589–1602, Dec 2006. doi: 10.1104/pp.106.085639. URL `http://www.hubmed.org/display.cgi?uids=17028152`.

T. Hawkins and D. Kihara. Function prediction of uncharacterized proteins. *Journal of Bioinformatics & Computational Biology*, 5(1):1 – 30, 2007. ISSN 02197200. URL `http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=24943306&site=ehost-live`.

K. H. Hebelstrup, A. U. Igamberdiev, and R. D. Hill. Metabolic effects of hemoglobin gene expression in plants. *Gene*, 398(1-2):86–93, Aug 2007. doi: 10.1016/j.gene.2007.01.039. URL http://www.hubmed.org/display.cgi?uids=17555891.

L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring Expression Data: Identification and Analysis of Coexpressed Genes. *Genome Research*, 9(11):1106–1115, 1999. doi: 10.1101/gr.9.11.1106.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. ISSN 01621459. URL http://www.jstor.org/stable/2282952.

I. Holmes and W. J. Bruno. Finding regulatory elements using joint likelihoods for sequence and expression profile data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 202–210. AAAI Press, 2000. ISBN 1-57735-115-0.

F. Hong, R. Breitling, C. W. McEntee, B. S. Wittner, J. L. Nemhauser, and J. Chory. RankProd: a bioconductor package for detecting differentially expressed genes in meta-analysis. *Bioinformatics*, 22(22):2825–2827, Nov 2006. doi: 10.1093/bioinformatics/btl476. URL http://www.hubmed.org/display.cgi?uids=16982708.

K. Horan, J. Lauricha, J. Bailey-Serres, N. Raikhel, and T. Girke. Genome cluster database. A sequence family analysis platform for Arabidopsis and rice. *Plant Physiology*, 138(1):47–54, May 2005. doi: 10.1104/pp.104.059048. URL http://www.hubmed.org/display.cgi?uids=15888677.

K. Horan, C. Jang, J. Bailey-Serres, R. Mittler, C. Shelton, J. F. Harper, J. K. Zhu, J. C. Cushman, M. Gollery, and T. Girke. Annotating genes of known and unknown function by large-scale coexpression analysis. *Plant Physiology*, 147(1):41–57, May 2008. doi: 10.1104/pp.108.117366. URL http://www.hubmed.org/display.cgi?uids=18354039.

K. Horan, C. Shelton, and T. Girke. Predicting conserved protein motifs with subhmms. *BMC Bioinformatics*, 11(1):205, 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-205. URL http://www.biomedcentral.com/1471-2105/11/205.

J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *Journal of Molecular Biology*, 296(5):1205–1214, Mar 2000. doi: 10.1006/jmbi.2000.3519. URL http://www.hubmed.org/display.cgi?uids=10698627.

N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. De Castro, P. S. Langendijk-Genevaux, M. Pagni, and C. J. Sigrist. The PROSITE database. *Nucleic Acids Research*, 34 (Database issue):227–230, Jan 2006. doi: 10.1093/nar/gkj063. URL http://www.hubmed.org/display.cgi?uids=16381852.

N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, B. Cuche, E. de Castro, C. Lachaize, P. Langendijk-Genevaux, and C. Sigrist. The 20 years of PROSITE. *Nucleic Acids Research*, 36(Database issue):D245, 2008.

R. A. Irizarry, B. M. Bolstad, F. Collin, L. M. Cope, B. Hobbs, and T. P. Speed. Summaries of Affymetrix GeneChip probe level data. *Nucleic Acids Research*, 31(4): e15, Feb 2003. URL `http://www.hubmed.org/display.cgi?uids=12582260`.

R. A. Irizarry, L. Gautier, B. M. Bolstad, C. M. with contributions from Magnus Astrand, L. M. Cope, R. Gentleman, J. Gentry, C. Halling, W. Huber, J. MacDonald, B. I. P. Rubinstein, C. Workman, and J. Zhang. *affy: Methods for Affymetrix Oligonucleotide Arrays*, 2006. R package version 1.12.1.

C. H. Jen, I. W. Manfield, I. Michalopoulos, J. W. Pinney, W. G. Willats, P. M. Gilmartin, and D. R. Westhead. The Arabidopsis co-expression tool (ACT): a WWW-based tool and database for microarray-based gene expression analysis. *The Plant Journal*, 46(2):336–348, Apr 2006. doi: 10.1111/j.1365-313X.2006.02681.x. URL `http://www.hubmed.org/display.cgi?uids=16623895`.

O. Johnson and J. Liu. A traveling salesman approach for predicting protein functions. *Source Code for Biology and Medicine*, 1:3–3, 2006. doi: 10.1186/1751-0473-1-3. URL `http://www.hubmed.org/display.cgi?uids=17147783`.

S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34(Database issue):354–357, Jan 2006. doi: 10.1093/nar/gkj102. URL `http://www.hubmed.org/display.cgi?uids=16381885`.

K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998. URL `http://www.hubmed.org/display.cgi?uids=9927713`.

K. Karplus, R. Karchin, G. Shackelford, and R. Hughey. Calibrating e-values for hidden Markov models using reverse-sequence null models. *Bioinformatics*, 21(22):4107–4115, Nov 2005. doi: 10.1093/bioinformatics/bti629. URL `http://www.hubmed.org/display.cgi?uids=16123115`.

J. Kasturi and R. Acharya. Clustering of diverse genomic data using information fusion. *Bioinformatics*, 21(4):423–429, 2005. doi: 10.1093/bioinformatics/bti186. URL `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/21/4/423`.

J. Kilian, D. Whitehead, J. Horak, D. Wanke, S. Weinl, O. Batistic, C. D'Angelo, E. Bornberg-Bauer, J. Kudla, and K. Harter. The AtGenExpress global stress expression data set: protocols, evaluation and model data analysis of UV-B light, drought and cold stress responses. *The Plant Journal*, 50(2):347–363, Apr 2007. doi: 10.1111/j.1365-313X.2007.03052.x. URL `http://www.hubmed.org/display.cgi?uids=17376166`.

D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *AAAI/IAAI*, pages 580–587, 1998. URL `citeseer.ist.psu.edu/koller98probabilistic.html`.

R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi. Low-complexity fuzzy relational clustering algorithms for Web mining. *IEEE-FS*, 9:595–607, Aug. 2001. URL `citeseer.ist.psu.edu/article/krishnapuram01lowcomplexity.html`.

A. Kundaje, M. Middendorf, F. Gao, C. Wiggins, and C. Leslie. Combining sequence and time series expression data to learn transcriptional modules. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2:194–202, 2005. ISSN 1545-5963. doi: http://doi.ieeecomputersociety.org/10.1109/TCBB.2005.34.

R. Leinonen, F. G. Diez, D. Binns, W. Fleischmann, R. Lopez, and R. Apweiler. UniProt archive. *Bioinformatics*, 20(17):3236–3237, Nov 2004. doi: 10.1093/bioinformatics/bth191. URL `http://www.hubmed.org/display.cgi?uids=15044231`.

W. K. Lim, K. Wang, C. Lefebvre, and A. Califano. Comparative analysis of microarray normalization procedures: effects on reverse engineering gene networks. *Bioinformatics*, 23(13):282–288, Jul 2007. doi: 10.1093/bioinformatics/btm201. URL `http://www.hubmed.org/display.cgi?uids=17646307`.

W. M. Liu, R. Mei, X. Di, T. B. Ryder, E. Hubbell, S. Dee, T. A. Webster, C. A. Harrington, M. H. Ho, J. Baid, and S. P. Smeekens. Analysis of high density expression microarrays with signed-rank call algorithms. *Bioinformatics*, 18(12):1593–1599, Dec 2002a. URL `http://www.hubmed.org/display.cgi?uids=12490443`.

X. Liu, D. Brutlag, and J. Liu. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes. In *Pacific Symposium on Biocomputing*, volume 6, pages 127–138. Citeseer, 2001.

X. Liu, D. Brutlag, and J. Liu. An algorithm for finding protein–dna binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nature biotechnology*, 20(8):835–839, 2002b.

S. Ma, Q. Gong, and H. J. Bohnert. An Arabidopsis gene network based on the graphical Gaussian model. *Genome Research*, 17(11):1614–1625, Nov 2007. doi: 10.1101/gr.6911207. URL `http://www.hubmed.org/display.cgi?uids=17921353`.

M. Madera. Profile comparer: a program for scoring and aligning profile hidden Markov models. *Bioinformatics*, 24(22):2630–2631, Nov 2008. doi: 10.1093/bioinformatics/btn504. URL `http://www.hubmed.org/display.cgi?uids=18845584`.

J. N. McClintick and H. J. Edenberg. Effects of filtering by Present call on analysis of microarray experiments. *BMC Bioinformatics*, 7:49–49, 2006. doi: 10.1186/1471-2105-7-49. URL `http://www.hubmed.org/display.cgi?uids=16448562`.

M. Meilă. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007. ISSN 0047-259X. doi: http://dx.doi.org/10.1016/j.jmva.2006.11.013.

L. A. Mueller, P. Zhang, and S. Y. Rhee. AraCyc: a biochemical pathway database for Arabidopsis. *Plant Physiology*, 132(2):453–460, Jun 2003. doi: 10.1104/pp.102.017236. URL `http://www.hubmed.org/display.cgi?uids=12805578`.

F. Murtagh. *Multidimensional Clustering Algorithms*. COMPSTAT Lectures 4. Physica-Verlag, Wuerzburg, 1985. URL `http://astro.u-strasbg.fr/~fmurtagh/mda-sw/`.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48 (3):443–453, Mar 1970. URL `http://www.hubmed.org/display.cgi?uids=5420325`.

D. R. Nelson, M. A. Schuler, S. M. Paquette, D. Werck-Reichhart, and S. Bak. Comparative genomics of rice and Arabidopsis. Analysis of 727 cytochrome P450 genes and pseudogenes from a monocot and a dicot. *Plant Physiology*, 135(2):756–772, Jun 2004. doi: 10.1104/pp.104.039826. URL `http://www.hubmed.org/display.cgi?uids=15208422`.

J. Nikkila, P. Toronen, S. Kaski, J. Venna, E. Castrn, and G. Wong. Analysis and visualization of gene expression data using self-organizing maps. *Neural Networks*, 15(8-9):953 – 966, 2002. ISSN 0893-6080. doi: DOI:10.1016/S0893-6080(02)00070-9. URL `http://www.sciencedirect.com/science/article/B6T08-46XBH1X-3/2/301c3d00109d36483a7a42b5b10ee485`.

G. Pavesi, G. Mauri, and G. Pesole. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, 17(suppl_1):S207–214, 2001. doi: 10.1093/bioinformatics/17.suppl_1.S207. URL `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/17/suppl%_1/S207`.

S. Persson, H. Wei, J. Milne, G. P. Page, and C. R. Somerville. Identification of genes required for cellulose synthesis by regression analysis of public microarray data sets. *Proceedings of the National Academy of Sciences of the United States of America*, 102 (24):8633–8638, Jun 2005. doi: 10.1073/pnas.0503392102. URL `http://www.hubmed.org/display.cgi?uids=15932943`.

T. Plotz and G. Fink. Feature extraction for improved Profile HMM based biological sequence analysis. *Proc. International Conference on Pattern Recognition*, 2004.

T. Plotz and G. Fink. A new approach for hmm based protein sequence family modeling and its application to remote homology classification. *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on*, pages 1008–1013, July 2005. doi: 10.1109/SSP.2005.1628742.

C. T. Porter, G. J. Bartlett, and J. M. Thornton. The Catalytic Site Atlas: a resource of catalytic sites and residues identified in enzymes using structural data. *Nucleic Acids Research*, 32(suppl_1):D129–133, 2004. doi: 10.1093/nar/gkh028. URL `http://nar.oxfordjournals.org/cgi/content/abstract/32/suppl_1/D129`.

A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, May 2006. doi: 10.1093/bioinformatics/btl060. URL `http://www.hubmed.org/display.cgi?uids=16500941`.

L. X. Qin, R. P. Beyer, F. N. Hudson, N. J. Linford, D. E. Morris, and K. F. Kerr. Evaluation of methods for oligonucleotide array data via quantitative real-time PCR. *BMC Bioinformatics*, 7:23–23, 2006. doi: 10.1186/1471-2105-7-23. URL `http://www.hubmed.org/display.cgi?uids=16417622`.

L. R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-124-4.

R. Rodriguez and R. Redman. Balancing the generation and elimination of reactive oxygen species. *Proceedings of the National Academy of Sciences of the United States of America*, 102(9):3175–3176, Mar 2005. doi: 10.1073/pnas.0500367102. URL http://www.hubmed.org/display.cgi?uids=15728396.

M. Schmid, T. S. Davison, S. R. Henz, U. J. Pape, M. Demar, M. Vingron, B. Schölkopf, D. Weigel, and J. U. Lohmann. A gene expression map of Arabidopsis thaliana development. *Nature Genetics*, 37(5):501–506, May 2005. doi: 10.1038/ng1543. URL http://www.hubmed.org/display.cgi?uids=15806101.

R. Schwacke, A. Schneider, E. van der Graaff, K. Fischer, E. Catoni, M. Desimone, W. B. Frommer, U. I. Flügge, and R. Kunze. ARAMEMNON, a novel database for Arabidopsis integral membrane proteins. *Plant Physiology*, 131(1):16–26, Jan 2003. doi: 10.1104/pp.011577. URL http://www.hubmed.org/display.cgi?uids=12529511.

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17 Suppl 1:243–252, 2001. URL http://www.hubmed.org/display.cgi?uids=11473015.

E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, Jun 2003. doi: 10.1038/ng1165. URL http://www.hubmed.org/display.cgi?uids=12740579.

T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. Rocr: visualizing classifier performance in r. *Bioinformatics*, 21(20):3940–3941, Oct 2005. doi: 10.1093/bioinformatics/bti623. URL http://www.hubmed.org/display.cgi?uids=16096348.

T. Smith and M. Waterman. Identification of common molecular subsequences. *J. Mol. Bwl*, 147:195–197, 1981.

G. K. Smyth. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3(1), 2004. doi: 10.2202/1544-6115.1027. URL http://www.hubmed.org/display.cgi?uids=16646809.

G. K. Smyth. *Limma: Linear Models for Microarray Data. In: Bioinformatics and Computational Biology Solutions using R and Bioconductor. R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds)*, pages 397–420. Springer, New York, 2005.

D. Steinhauser, B. H. Junker, A. Luedemann, J. Selbig, and J. Kopka. Hypothesis-driven approach to predict transcriptional units from gene expression data. *Bioinformatics*, 20(12):1928–1939, Aug 2004a. doi: 10.1093/bioinformatics/bth182. URL http://www.hubmed.org/display.cgi?uids=15044239.

D. Steinhauser, B. Usadel, A. Luedemann, O. Thimm, and J. Kopka. CSB.DB: a comprehensive systems-biology database. *Bioinformatics*, 20(18):3647–3651, Dec 2004b. doi: 10.1093/bioinformatics/bth398. URL http://www.hubmed.org/display.cgi?uids=15247097.

Y. Sun and J. Buhler. Designing patterns and profiles for faster hmm search. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 6(2):232–243, 2009. ISSN 1545-5963. doi: http://dx.doi.org/10.1109/TCBB.2008.14.

S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–285, Jul 1999. doi: 10.1038/10343. URL `http://www.hubmed.org/display.cgi?uids=10391217`.

R. Team. R: A language and environment for statistical computing. *R Foundation for Statistical Computing Vienna Austria ISBN*, 3(10), 2008.

G. Thijs, M. Lescot, K. Marchal, S. Rombauts, B. De Moor, P. Rouze, and Y. Moreau. A higher-order background model improves the detection of promoter regulatory elements by gibbs sampling. *Bioinformatics*, 17(12):1113, 2001.

J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, Nov 1994. URL `http://www.hubmed.org/display.cgi?uids=7984417`.

K. Toufighi, S. M. Brady, R. Austin, E. Ly, and N. J. Provart. The Botany Array Resource: e-Northerns, Expression Angling, and promoter analyses. *The Plant Journal*, 43(1):153–163, Jul 2005. doi: 10.1111/j.1365-313X.2005.02437.x. URL `http://www.hubmed.org/display.cgi?uids=15960624`.

B. Usadel, A. Nagel, D. Steinhauser, Y. Gibon, O. E. Bläsing, H. Redestig, N. Sreenivasulu, L. Krall, M. A. Hannah, F. Poree, A. R. Fernie, and M. Stitt. PageMan: an interactive ontology tool to generate, display, and annotate overview graphs for profiling experiments. *BMC Bioinformatics*, 7:535–535, 2006. doi: 10.1186/1471-2105-7-535. URL `http://www.hubmed.org/display.cgi?uids=17176458`.

K. Vandepoele, T. Casneuf, and Y. Van de Peer. Identification of novel regulatory modules in dicot plants using expression data and comparative genomics. *Genome Biology*, 7(11):R103, Nov 2006. doi: 10.1186/gb-2006-7-11-r103. URL `http://www.hubmed.org/display.cgi?uids=17090307`.

D. Wang, J. F. Harper, and M. Gribskov. Systematic trans-genomic comparison of protein kinases between Arabidopsis and Saccharomyces cerevisiae. *Plant Physiology*, 132(4):2152–2165, Aug 2003. URL `http://www.hubmed.org/display.cgi?uids=12913170`.

H. Wei, S. Persson, T. Mehta, V. Srinivasasainagendra, L. Chen, G. P. Page, C. Somerville, and A. Loraine. Transcriptional coordination of the metabolic network in Arabidopsis. *Plant Physiology*, 142(2):762–774, Oct 2006. doi: 10.1104/pp.106.080358. URL `http://www.hubmed.org/display.cgi?uids=16920875`.

P. L. Whetzel, H. Parkinson, H. C. Causton, L. Fan, J. Fostel, G. Fragoso, L. Game, M. Heiskanen, N. Morrison, P. Rocca-Serra, S. A. Sansone, C. Taylor, J. White, and C. J. Stoeckert. The MGED Ontology: a resource for semantics-based description of microarray experiments. *Bioinformatics*, 22(7):866–873, Apr 2006. doi: 10.1093/bioinformatics/btl005. URL `http://www.hubmed.org/display.cgi?uids=16428806`.

C. J. Wolfe, I. S. Kohane, and A. J. Butte. Systematic survey reveals general applicability of guilt-by-association within gene coexpression networks. *BMC Bioinformatics*, 6:227–227, 2005. doi: 10.1186/1471-2105-6-227. URL `http://www.hubmed.org/display.cgi?uids=16162296`.

J. R. Wortman, B. J. Haas, L. I. Hannick, R. K. Smith, R. Maiti, C. M. Ronning, A. P. Chan, C. Yu, M. Ayele, C. A. Whitelaw, O. R. White, and C. D. Town. Annotation of the Arabidopsis genome. *Plant Physiology*, 132(2):461–468, Jun 2003. doi: 10.1104/pp.103.022251. URL `http://www.hubmed.org/display.cgi?uids=12805579`.

C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O'Donovan, N. Redaschi, and B. Suzek. The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Research*, 34(Database issue):187–191, Jan 2006. doi: 10.1093/nar/gkj161. URL `http://www.hubmed.org/display.cgi?uids=16381842`.

E. Xing, W. Wu, M. Jordan, and R. Karp. Logos: A modular bayesian model for de novo motif detection. *Journal of Bioinformatics and Computational Biology*, 2(1): 127–154, 2004.

Y. Yang, L. Engin, E. S. Wurtele, C. Cruz-Neira, and J. A. Dickerson. Integration of metabolic networks and gene expression in virtual reality. *Bioinformatics*, 21(18): 3645–3650, Sep 2005. doi: 10.1093/bioinformatics/bti581. URL `http://www.hubmed.org/display.cgi?uids=16020466`.

P. Zimmermann, M. Hirsch-Hoffmann, L. Hennig, and W. Gruissem. GENEVESTIGATOR. Arabidopsis microarray database and analysis toolbox. *Plant Physiology*, 136 (1):2621–2632, Sep 2004. doi: 10.1104/pp.104.046367. URL `http://www.hubmed.org/display.cgi?uids=15375207`.

P. Zimmermann, L. Hennig, and W. Gruissem. Gene-expression analysis and network discovery using Genevestigator. *Trends in Plant Science*, 10(9):407–409, Sep 2005. doi: 10.1016/j.tplants.2005.07.003. URL `http://www.hubmed.org/display.cgi?uids=16081312`.