NTNU
Innovation and Creativity

Flexible Discovery of Modules with Distance Constraints

Øystein Lekang

Master of Science in Computer Science
Submission date: June 2006
Supervisor: Arne Halaas, IDI
Co-supervisor: Geir Kjetil Sandve, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Many authors argue that finding single transcription factor binding sites is not enough to be able to make predictions with regard to regulation in eukaryotic genes, as is the case with simpler prokaryotes. With eukaryotes combinations of transctiption factors must be modeled as a composite motif or module. In some cases even with a restriction on distance between individual sites, or within the module. Create a module discovery tool capable of using both deterministic patterns and position weight matrices as input, that can impose restrictions on distance. Use the tool for module discovery and evaluate the results.

Assignment given: 19. January 2006
Supervisor: Arne Halaas, IDI

Abstract

Currently, whole genomes have been sequenced and the next big challenge is to map the functional elements. The essence of this challenge is finding specifics in a vast amount of information, and it is therefore natural for the fields of biology and information technology to cooperate in creating solutions to meet it. The field of motif discovery has in resent years seen a shift of attention from single motifs to sets of motifs that cooperate in regulating the expression of a gene. This has resulted in a new and more challenging objective. Instead of finding single transcription factor binding sites (TFBS) we now have to tackle the problem of finding collections of TFBSs modelled as composite motifs (modules). The objective of this project is to create a tool that can use Position Weight Matrices as input, is able to impose distance restrictions on the composite motifs generated, and lends itself to further development.

To validate the functionality of the tool two large datasets where used. The first was from the TransCompel database and contains combinations of motifs known to span a short distance. With this dataset evaluation was performed on a per nucleotide position basis. The other was ChIP-CHIP-data from yeast (*Saccharomyces*), where the presence of modules was evaluated.

With TransCompel data the tool achieved a relatively low but positive correlation. It was also able to discern a higher correlation with distance restricted. The low correlation is mainly caused by problems related to finding matches for the single motifs in the sequences.

On yeast data the expectations where not hi, given that the dataset is based on ChIP-CHIP experiments. This also causes the dataset to lack positional information, and the evaluation was performed on presence alone. The tool achieved a positive correlation on these data, and found an indication that distances may be involved.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years the research into discovery of transcription factor binding sites (TFBS) har recieved much attention. Lately the focus has been mainly to the idèa of modules. That is, a collection of TBFS in the upstream region of a gene that together affect regulation of that gene. A multitude of applications have been created for the purpose of finding TFBS. Modules have been modelled by a wide variety of models, ranging from complex combinatorial and hidden markov models to simple set models. Many such models are equipped with some kind of restraint on distance. This is done under the assumption that physical nearness may be a factor if the motifs affect regulation together. Distance restraints are most common where the composite motif is modelled as a tuple of single motifs. In that case distance restraints can be imposed on the distance between motifs or spanning the whole module.

In this project the distance restriction is a maximum window. The module is modelled as a set, so as not to introduce complexity that could impose restrictions other then that on distance. This way results are directly connected to the distance restraint with as few other factors as possible. To make the tool flexible with regard to what data it can use, it is equipped with capability to use both deterministic patterns (regular expressions) and probabilistic position weight matrices.

Chapter 2

Biology

This chapter describes the biological background. It starts with a short introduction to some of the biological consepts concerning regulation. After this it describes some of the wetlab tools and methods used to find functional elements in the Deoxyribonucleic acid (DNA). These are presented because they are often the methods used as input, premise or control in for computerised methods.

2.1 Biological background and consepts

For computer scientists DNA is a string over the alphabet $\sum$. Figure 2.1 displays the double helix of DNA. Each strand with it's nucleotides coresponding to the nucleotide on the other strand. It is included here to visualise the complexity of the actual systems, compared to the models used to understand and analyse them.

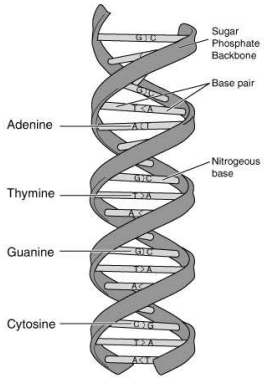


Figure 2.1: Double helix of DNA, displaying all 4 nucleotides[1]

2.1.1 Transcription

Transcription is the process whereby the DNA sequence in a gene is copied into messenger-RNA (mRNA). This process of copying the gene can be said to be performed by an

enzyme called *RNA-polymerase*. RNA-polymerase is attracted by signals bound to binding sites denoted by "promotor" and "enhanser" in figure 2.2. These signals are called transcription factors (see section 2.1.3). The transcription starts at a "transcription start site" (TSS) at the beginning of the coding region. There may be more than one TSS for each gene. Thereafter the nucleotides in the coding region are coupled with theire conterpart creating the mRNA. The mRNA may then go trough "splicing" where the introns are removed, and may be transported out of the nucleus to be translated into a proteine. These processes are however outside the scope of this project, as it's focus is on transctiption and the gene regulation ocuring through transcription factors.

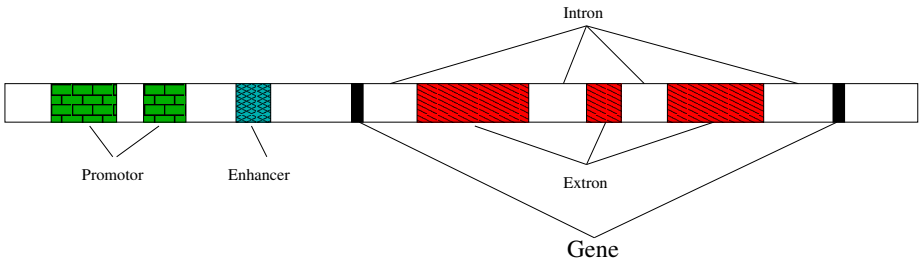


Figure 2.2: Coding area for a gene, including the promoter region.

2.1.2 Regulation

An important consept in this report is gene regulation. "Regulation of gene expression" or "gene regulation." Regulation is the amount or timing of introduction of a functional product of a gene. The product can be anything from mRNA to a proteine or even a modified proteine. In this text the term is used as a descriptor of the affinety to transcribe a given gene. An up regulated gene has higher likelihood of beeing transcribed and is transcribed more often.

2.1.3 Transcription factor binding sites

Transcription Factor is a discription of proteins that attach themselves to binding sites in a genes promotor region, and thereby regulate the transctiption of the gene. For this reason the discovery of such functional elements in the non-coding regions of DNA can be very valuable. The regulation of a gene is much more complex than this description implyes. Orientation and folding of the DNA are significant, among many other factors. Still the transcription factor binding sites are two of the most important functional elements in any genome, for understanding regulation. TFBSs are usually short, around 5-15 base pairs (bp), and are frequently degenerate sequence motifs (figure 3.1). As a result of this potential binding sites can occur frequently by chance in large genomes of higher eukaryotes. This makes the functional elements even harder to find.

2.2 Identifying TFBS

How the genome encodes information that specifies when and where a gene will be expressed is a major challenge in interpreting genome sequences. This process has to be started by identifying regions of the genome that contain regulatory information. Tradisionaly discovering binding information for TFs has been done through time consuming aproaces. These where either analysis that monitors the change in mobility when DNA and protein bind, fotprinting that identifies the region of DNA protected by protein, or reporter constructs. In reasont years, however much effort has been directed to develop technologies that has a high througput. Methods has been developed for this purpose for *in vitro*(SELEX) and *in vivo*(Microarray, ChIP-CHIP) identification.

These high throughput methods have made genome sequences available for a number of spicies and thereby opened the door for computational methods.

Chapter 3

Motif Discovery

Computerised methods for analysing biological data have been developing for years. Now that whole genomes have been sequenced they are more promising than ever. This chapter will start by describing how motifs usually are represented when treated *in silico*. Priority is lent to the topics that are used in this project. Then some general topics concerning motif discovery will be covered, along with a description of the tool "General Composite Motif Discovery" from witch this project stems. Lastly some evaluation values are listed for reference.

3.1 Motif representation

This section gives a short introduction to the two most common representations of motifs. They are used in publicly available databases like Prosite[30], Transfac[42], and Jaspar[6].

3.1.1 Deterministic Patterns

The simplest kind of deterministic pattern is just a sequence of characters from the alphabet $\sum$. In terms of DNA this would be the alphabet $\{A, C, G, T\}$. On top of this there are multiple expansions like *IUPAC extensions* and *regular expressions* or more generally ambiguities, wildcards and gaps. Consensus strings are strings over the alphabet $\sum$ that captures the composition of most occurrences of the motif. This consensus string is often expanded either by regular expressions or IUPAC strings[2]. Regular expression subsets most used are braces for ambuigueties or a dot for "any character" while IUPAC has different symbols for the different combinations of bases/nucleotides. Figures 3.1.1 to 3.1.3 displays the different variants. The reg.exp. and IUPAC string are not completely equal since the IUPAC "N" equals the "." or [ACGT] in regular expressions.

Significance With string representations of motifs are usually logods score over a markov background. This assumes independence between nucleotides in varying degree depending on the order of the markov chain. The reason for using logarithms is purely computational. With logarithms the calculation is mainly addition instead of multiplication with probabilities, thereby fewer instructions. It also removes trouble with accuracy on small likelihoods.

TACGAT
TATAAT
TATAAT
GATACT
TATGAT
TATGTT

TATAAT

3.1.1: Consensus String

TATRNT

3.1.2: IUPAC string

TAT[AG][ACT]T

3.1.3: Regular Expression

A	– 2.81	1.33	– 2.81	0.41	0.79	– 2.81
C	– 2.81	– 2.81	1.44	– 2.81	1.44	– 2.81
G	0.51	– 2.81	– 2.81	2.00	– 2.81	– 2.81
T	0.73	– 2.81	0.73	– 2.81	– 1.17	0.97

3.1.4: Possition Weight Matrix

Figure 3.1: The most common motif representations

3.1.2 Position Weight Matrix

A Position Weight Matrix is a matrix of score values that gives a weighted match to any given substring of fixed length. In DNA each row will represent a nucleotide, and the columns are possitions in the substring. The PWM is created from an alignment matrix (count matrix) where the count of each nukleotide in each position when aligned is recorded. From the alignment matrix a probability matrix can be created (see equation 3.1). This is a matrix of the likelihood weighted on background residue. In this probability matrix all columns or "positions" $\sum_{i=1}^{A} f_{i,j} = 1$ since each position must have a character from the given alphabet A. Equation 3.2 results in the PWM. An example is shown in figure 3.1.4, it is created from the sequences in figure 3.1.1.

$$f_{i,j} = \frac{n_{i,j} + p_i k}{\sum_{i=1}^{A} n_{i,j} + k} \tag{3.1}$$

$$W_{i,j} = \ln \frac{f_{i,j}}{p_i} \tag{3.2}$$

$f_{i,j}$	corrected frequency of residue i at position j
A	alphabet size
p_i	prior/background probability for residue i
k	pseudo weight (arbitary)

The PWM can then be used to scan sequences for hits by using a sliding window, and calculating the score for each position of the window. The score is equivalent to the logarithm and the ratio between the product of the matrix frequencies and the product of the prior probabilities of the residues found in the sequence segment. This can also

be thought of as a logarithm of the ratio of the probabilities of being within or without the motif.

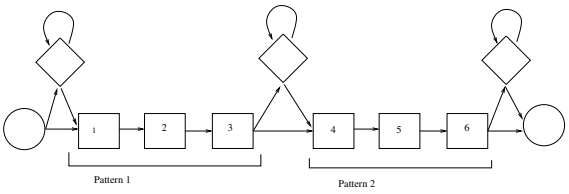
Significance for PWMs are calculated in multiple ways. One common solution is to calculate the *Information Content* of the matrix (see equation 3.3, w is the length of the matrix).[11]

$$\begin{aligned} f_{i,j} &= \frac{n_{i,j} + p_i k}{\sum_{i=1}^{A} n_{i,j} + k} \\ I &= \sum_{j=1}^{w} \sum_{i=1}^{A} f_{i,j} \ln \frac{f_{i,j}}{p_i} \end{aligned} \tag{3.3}$$

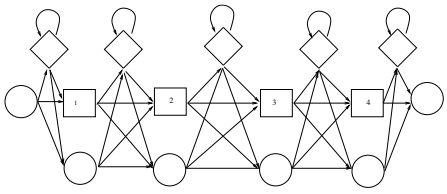
Another way is to use the background likelihood, or p-value. This is most often done by matching the matrix on either a real background [37] or one generated from a markov model [4, 37]. It can also be calculated directly from the background model[41]. See section 3.6 for a more detailed description of significance.

3.1.3 Other Models

There are multiple other models, specially in machine learning approaches. An example Hidden Marker Models (HHM). A HMM is a statistical model that assumes the system modelled is a Markov process with unknown parameters. In a Markov process the probability distribution of future states, depends solely on the current state. The challenge is to determine the hidden parameters, from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. Figure 3.2.2 is a commonly used HHM topology for sequence analysis[3, 35]. It allows for insertions and deletions, or gaps and mismatches. Figure 3.2.1 displays the topology of meta-MEME, where only insertions are allowed and only between single motifs.



3.2.1: Meta-MEME HHM



3.2.2: Profile HHM

Figure 3.2: Composite motif model from meta-MEME, and a general HHM for profiles

3.2 Discovering Transcription Factor Binding Sites

Pattern discovery and classification have been in use for a long time, even in a biological context. This has resulted in a wide variety of methods for computer sciences being applied to the field. Table 3.1 list some approaches.

Machine Learning	
Neural Networks	Blekas et al. [9] [8]
	Gulukota et al. [17]
Support Vector Machine	Eveillard et al. [14]
Hidden Markov Model	Krogh et al. [3] [20]
	Meta-MEME [16]
Heuristics	
Gibbs sampling	AlignAce [19]
	BioProspector [25]
Expectation Maximization	MEME [7]
Greedy algorithm	CONSENSUS [18]
Exact	
Enumeration	MOTIF [40]
	Pratt [22]
	TEIRESISAS [38]

Table 3.1: Tools and algorithms for motif discovery

3.2.1 Prior Knowlidge and Limiting Searchspaces

The way one approaches motif discovery is largely depended on what, and how much *a priori* knowlidge that is available. One approach is to use the whole genome as the search space, and search for patterns in upstream regions that are statistically overepresented. Alternatively one can combine computerized discovery with biological methods to limit the search space to "co regulated" genes or genes that we suspect may be regulated in a similar way. This way the search is confined to what we suspect to be a positive set. Another way to benefit from prior knowlidge is that of RepeatMasker[5]. In the mid 1960's scientists discovered that many genomes contain stretches of highly repetitive DNA sequences. These are largely belived to be nonfunctional. RepeatMasker is a program that removes repeats from the genome sequence, and thereby limiting the searchspace. Evolution can also help with the process. *Phylogenetic footprinting* is a method to identify regions of sequence conservation between genomes. The strategy is based on the assumption that evolution causes the important regulatory sites to be contained in these conserved regions. The technic has been described as searching for "islands of conserved sequences in seas of less conserved noncoding sequence"[24].

3.3 Modules or Composite Motifs

Computational discovery of single motifs have proven profitable, particularly when applied to sequences from prokatyotic organisms and yeast. Some cases however provide a rate of false positives that remains to high. Non-coding regions are much longer in

humans or mice compared to yeast. Kellis et. al.[27] states that yeast genomes are about 30% coding whereas the human genome is about 2% coding, and that intergenic regions of yeast consist of about 15% regulatory elements, while in human about 3% of intergenic regions are regulatory elements. This indicates the difficulty of extrapolating classic motif discovery techniques to higher eukaryotes. In addition it is believed that transcriptional regulation require the combinatorial interplay of multiple factors [10, 37, 23, 29]. See figure 3.3 for a simple conceptual picture.

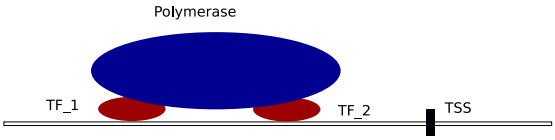


Figure 3.3: Conceptual picture of multiple Transcripion Facors regulating the transcription of a gene. In this case RNA polymerase binds to the facors TF_1 and TF_2, and they must therefore both be there to have an effect on regulation.

Sinha et. al. divides composite models into four groups[39]. These are the *set model* witch is built on the assumption that the occurrence of a set of simple motifs is all that is needed for regulation. This is the least complex of the model categories but is believed to be close to a biological reality, with higher eukaryotes. The *tuple model* consists of ordered tuples of single motifs, with distance constraints on the gaps. The constraints on distance are usually tight intervals or fixed lengths. *Hidden Markov Models*(HHM) is the third of the group. HHMs consist of a number of states, with wich an emission- and a transition probability distribution is associated. This enables walking through the states in the HHM emitting a nucleotide in each emission-state. (see figure 3.2.1) The last group is the *Boolean* that model composite motifs by boolean combinations of counts, distances or locations of simple motifs. i.e. given the simple motifs $m1$ and $m2$ a composite motif might be "(count($m1$) > 2) or ((count($m2$) >= 1) and (count($m1$) = 1) and (dist($m1, m2$) < d))." Which would specify the motif as wither more than 2 of $m1$ or one $m1$ and at least one $m2$ with a limited distance between $m1$ and $m2$.

Aproaches

With composite motif discovery one can either discover *de-novo* composite motifs directly, or use single motifs as input and combine them to composites. Some tools do both the novel single motif discovery and then use combinations of these for composite building.

The *BioProspector* program by Liu et. al. [25] is an example of finding the composite directly. It finds a tuple of two motifs with a ranged gap between them. It takes as input the width of the two blocks for the motifs, and a gap range. The algorithm then uses a modified Gibbs sampling strategy to find the composite.

3.3.1 Distance Restrictions in Modules

Many kinds of distance restrictions are in use. Of these strict or flexible gaps[25, 21, 13] and distance windows. Alternatively one could provide some function on the distance between motifs that premiers motifs separated with some distance and penalizes evenly as the distance grows or shrinks.

3.4 General Composite Motif Discovery

GCMD[15] is a tool that combines reg.exp motifs into composites. It belongs *set* model as occurrence is modeled by a bit for each sequence. This enables the tool to use these bitsets that represent occurrences in sequences to combine motifs through a logical "and" operation. The algorithm uses negative log likelihood for significance, thereby making maximation the optimal for both significance and support. It keeps track of the best composites in a pareto front, and uses the current front as a basis for pruning. Although this algorithm must be said to be a member of Sinhas *set* model it does however have the option to enforce order on the motifs in a composite. The worst case time complexety of GCMD is $\binom{n}{c}$, where n is the number of input motifs and c is the desired number of components. In practice the pruning arrangement makes the algorithm run faster.

GCMD is the model on witch this project is built. The algorithm is to be reimplemented with support for position weight matrices, and the ability to impose distance restraints on the composites it produces. The chosen distance model is a window defining the maximum length of the composite. The choice is made to keep the flexibility of a set model, and since the algorithm should be able to run multiple distances simultaneously it is not so much a restriction as a expansion of output information.

3.5 Multiobjective Optimization Problem

When performing motif discovery two important values are the coverage or support and the significance of motifs. These values are in many respects opposites. Ta increase one there is usually a fall in the other. This kind of optimization problem is referred to as a Multiobjective optimization problem. The Multiobjective Optimization Problem (MOP) was defined by Osyczka[31] as the problem of finding

> *a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.*

More formatly this can be stated as follows [12]: The vector $\bar{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ denotes the optimal solutions. (Usually more than one) Find the $\bar{x}^*$ that will satisfy the m inequality constraints

$$g_i(\bar{x}) \geq 0 \; i = 1, 2, 3, \ldots, m \tag{3.4}$$

the p equality constraints

$$h_i(\bar{x}) = 0 \; i = 1, 2, 3, \ldots, m \tag{3.5}$$

and optimizes the vector function

$$\bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \ldots, f_k(\bar{x})]^T \tag{3.6}$$

where $\bar{x} = [x_1, x_2, x_3, \ldots, x_n]^T$ is the vector of decision variables.

The constraints of given of equations (3.4) and (3.5) represent the restrictions imposed on the decision variables and defines the *feasible region* F and any point $\bar{x} \in F$ defines a *feasible solution*.

If there is an $\bar{x}^*$ where

$$\forall_{x \in F}(f_i(\bar{x}^*) \geq f_i(\bar{x})) \quad [1] \tag{3.7}$$

then the problem is trivial an has one solution. This is however rarely the case. Normally there is no one solution where all the $f_i(\bar{x})$ has their maximum in F (see figure 3.4) at a common point x^*.

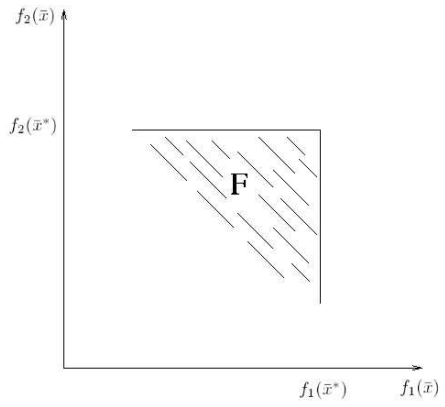


Figure 3.4: Ideal solution where all objectives has their maximum in one point, in a problem with two objective functions

Pareto Optimum is a consept formulated by Vilfredo Pareto[34], and is considered the birth of MOP. A vector $\bar{x}^*$ can be considered *Pareto Optimal* if there exists no other vector $\bar{x} \in F$ that increases some criteria without decreasing another criteria. More formally stated, a vector of decision variables $\bar{x}^* \in F$ is Pareto Optimal if there exists no $\bar{x} \in F$ where

$$\forall_{i \in I} f_i(\bar{x}) \geq f_i(\bar{x}^*)\ i = 1, 2, \ldots, k \tag{3.8}$$

or

$$\exists_j f_j(\bar{x}) > f_j(\bar{x}^*) \tag{3.9}$$

As already stated this, unfortunately, almost always gives more than one solution, but a set for *nondominated* solutions.

Pareto Front is the maxima in the Pareto sense. It is located in the boundary of the design region, or in the locus of the tangent points of the objective functions. In general this line or surface that contains these points is not easy to find. Figure 3.5 displays a bi-objective problem with the Pareto Front marked.

[1] In the litterature authors ussually prefer to use mimimum as optimal, so the $\geq$ would be a $\leq$.

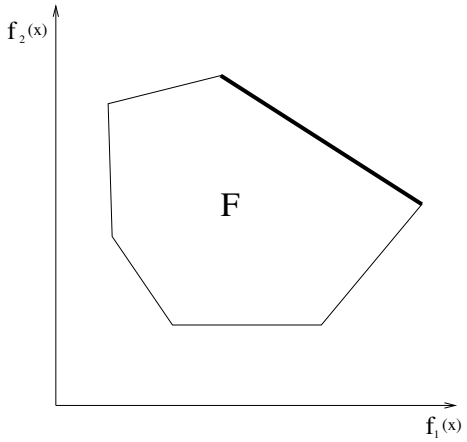


Figure 3.5: Problem with two objective functions to be maximized, the pareto front is marked with a bold line

3.6 Evaluation Statistics

The process of pattern matching can be viewed as a hypothesis testing, where the null hypothesis (H_0) is that the observations are the result of pure chance, and the alternative hypothesis (H_1) is that the observations show a real effect. Then the $p - value$ can be computed, which is the probability that a test statistic at least as significant as the one observed would be obtained assuming that the null hypothesis were true. The smaller the $p - value$ the stronger evidence against the null hypothesis. In a hypothesis test this p-value would be compared to an acceptable significance value α. If the $p - value$ is at least as small as α the null hypothesis is ruled out and the alternative hypothesis is valid.

If put in a pattern matching context H_0 is that the sequence segment is noise (generated by the background), and H_1 that it is a TFBS. The $p - value$ would be the probability that the sequence segment is generated by the background model. This $p - value$ is compared to a threshold which determines whether is a match is found.

Significance

The significance of motifs in this project is a negative logarithm (base 2) of the required p-value to constitute a hit. In other words a negative log likelihood over the background.

Symbol Explanation

These symbols are used for the calculation of the remaining statistical values.

- TP - True Positive
- FP - False Positive
- TN - True Negative
- FN - False Negative

Sensitivity

The probability that a statistical test will be positive for a true statistic, or that a

TFBS is predicted where one in present.

$$Sn = \frac{TP}{TP + FN} \tag{3.10}$$

Specificity

The probability that a statistical test will be negative for a negative statistic, or that no TFBS is predicted where none is present.

$$SP = \frac{TN}{TN + FP} \tag{3.11}$$

Possitive Predictive Value

The probability that a test gives a true result for a true statistic.

$$PPV = \frac{TP}{TP + FP} \tag{3.12}$$

Correlation Coefficient

Also known as the Pearson product-moment coefficient of correlation in the particular case of two binary variables.

$$CC = \frac{TP * TN - FN * FP}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}} \tag{3.13}$$

Performance Coefficient

Proposed used in this context by Pevzner et.al. [33], and used in the model this project uses for evaluation [28].

$$PC = \frac{TP}{TP + FN + FP} \tag{3.14}$$

Average Site Performance

Proposed by Burset et.al. [26], also used in [28].

$$ASP = \frac{Sn + PPV}{2} \tag{3.15}$$

Chapter 4

Design and Implementation

This chapter describes the main challenges and chosen solutions for the implementation of the tool. It presents the datastructures and algorithms in various detail depending on theire significance for the tool as a whole.

4.1 Motifs and Variants

The model of the motif in is largely inherited from GCMD[15, 43] in that the search is based on bitwise operations on a bitset. It is also similar in that a significance is needed to drive the search. The support or coverage value is essentially the cardinality of the bitset, and is therefore available. In this project however PWMs are to be used as input in addition to regular expressions, and distance restriction are imposed. The inclusion of PWMs poses some interesting questions on how to compute the significance and support values, discussed in section 4.3. The inclusion of PWMs does however result in three different kinds of motifs. These being the two input types and the composite. They are essentially different in what they contain, althoug they share the values needed for the search. The motifs in this project are modeled as in figure 4.1.

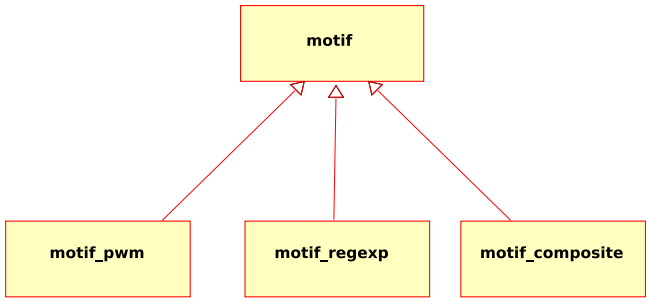


Figure 4.1: The diffenrent kinds of motifs and theire relationship

Another model considered was to have the composite motif, just be an index of other motif models. This would be very easy to understand for reuse and generally elegant. It would however have to be implemented with caching, to avoid the calculation of resulting bitsets, significance and matches to sequences every time the motif is to be expanded or tested for values. These issues makes that model approximate the

one used. In addition there are some considerations conserning mutability with bitsets. The *and* opperations work on the bitset, so a new bitset has to be created to represent the composite, else the original motif is changed. Also the python *copy* and *deepcopy* operations are unable to traverse into the java code. This could be remedied by implementing them, but it was decided that these considerations combined justifies the model with the composite as a subclass of motif.

A deciding factor on the general model of the motif is that it needs to incorporate distance restraints. This means keeping track of distinct hits, and a separate bitset and thereby support for each distance class. A distance class in this context refers to one of possibly several distance restraints that are run concurrently. Figure 4.1 shows the model. The choice of this solution over creating separate motifs for each distance lies in the amount of work to evaluate the motif distances, and is discussed int section 4.2.

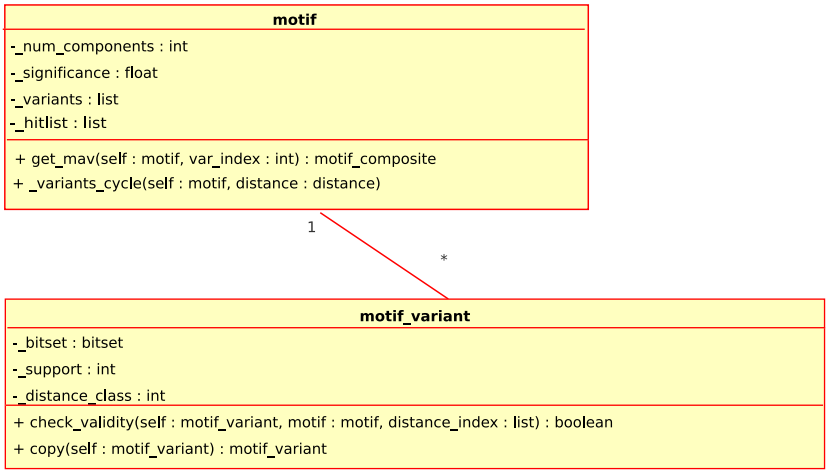


Figure 4.2: The Motif model with variants. The figure only display variables and methods that are discussed.

4.2 Distance

The distance restrictions should not limit what the tool is able to find, and thereby foregoing the advantage in the flexibility of the set model. Instead it hoped to achieve a distance "categorisation" so that one can output optimal composites for each distance class, for comparison.

Distance restrictions are kept track of by a dedicated class, see figure 4.2 The primary function here is the to evaluate the requred size of the distance window in a sequence for the motif. This is done by the function *calc_dist_class*. The function uses the hitlists[1] of the motif and generates the permutations of hits in each sequence. The permutations are then evaluated and the tightest window and its position is kept, see algorithm 4.2.1. By arranging the diffenrent distance classes as variants of motifs, this calculation only has to be excecuted once for each motif, not for each variant. Variants receive the minimal distace for each sequence and can thereby determine theire bitsets, and whether they are to continue theire existence. The calculation also removes the posibillity of overlaps between members of a composite.

[1] A list containing matches for each sequence

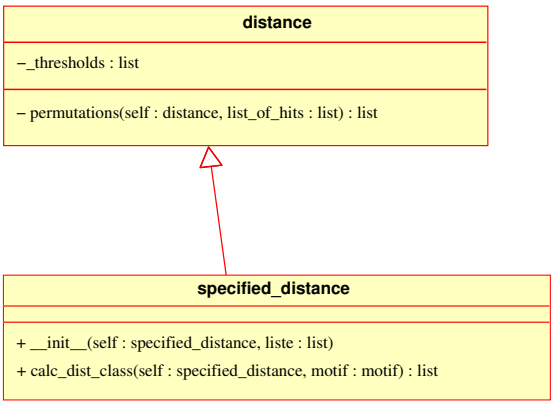


Figure 4.3: The distance class, only discussed members. This is a simplified picture, in reality specified distance inherits from k_distance, a similar class initiated with a k that creates the thresholds $10 * 2^x, x = 1, 2, .., k$.

Algorithm 4.2.1: DISTANCE EVALUATION($permutations$)

procedure CALC LENGTH($permutations$)
local $stub \leftarrow (0, 1)$
local $distance = MAX\_INT$
local $window$
for each $perm \in permutations$
do
- **local** $temp$
- **for each** $position \in perm$
  - **do**
    - **comment:** Add the start and end positions for the hit,
    - **comment:** along with markers to a list
    - $temp.append(position.start, 0)$
    - $temp.append(position.end, 1)$
- $temp.sort()$
- **comment:** Assert that there are no overlaps, using the markers
- **if** $temp.column(2) = stub * len(perm)$
  - **then**
    - **comment:** Check whether this is the tightest window
    - **if** $temp.last - temp.first < distance$
      - **then**
        - **do**
          - $distance \leftarrow temp.last - temp.first$
          - $window \leftarrow (temp.first, temp.last)$

return $(distance, window)$

4.3 Significance

In a pattern recognition context the term significance most commonly refers to the probability of observing the same pattern by chance, or in other words how likely the pattern is in a background model. The following is a discussion on how to evaluate and implement significance for the two motif representations discussed in section 3.1.

4.3.1 Regular Expressions

For regular expressions this project applies as significance a logarithm of the likelihood of hits in random sequences. The calculation estimates this by assuming independency of nucleotides in sequences, and using background likelyhoods for the nucleotides. To ease the calculation of significance log2 of likelihoods are used. This allows for addition instead of multiplication. Ambiguities have to be treated as a single char of the pattern with regard to the significance. And the "any symbol" '.' has a likelyhood of 1. This calculation results in the logartihm of the likelihood of matching a sequence in one position. s_{pos} ($p-value$). To expand this into the significance one must evaluate the length of the pattern compared to the average length of sequences. This is computed as can be seen in equation 4.1, except in two special cases.

- Pattern is longer than the average sequencelength
  In this case the positional significance is used ($log2(p_{pos})$).
- The likelihood is so small as to create computational problems.
  This happens at a likelyhood of about 10^{-13}.
  In this case the calculation in equation 4.2 is used.

$$s_{seq} = log2(1 - (1 - p_{pos})^{l_{avg}-l}) \tag{4.1}$$

$$s_{seq} = p_{pos} + log2(l_{avg} - l) \tag{4.2}$$

s_{seq}	Significanse
s_{pos}	Possitional significance
p_{pos}	Possitional likelihood in background
l_{avg}	Average length of sequences
l	Pattern length

4.3.2 PWM

Significance for pwms are based on the same principle. The likelyhood of matching random sequences, generated from the background. Still independence is assumed in the background. The background is essentially a Markov chain of order 0.

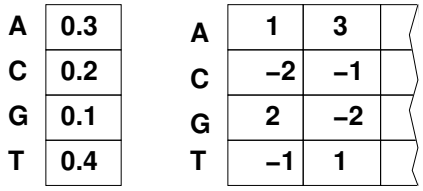
The main difference from the regular expression patterns, is that with regular expressions there is no need for a threshold value. Whether the reg.exp. hits a sequence is boolean. With PWMs however there is a score calculated for each position, and it therefore needs a threshold to define a hit. This threshold value is the score needed to achieve a low enough likelihood over the background, and is therefore defined by the significance or *visa versa*.

The most usually approach to setting these values is to specify the wanted significance (p-value). This achieves good control with the False Positives (FP) rate, since

all matches must have a small enough likelihood over the background to satisfy the p-value. In addition to this two other approaches has been implemented in this project. The second uses True Positives (TP) to set the threshold, and the third a balance between False Positives and False Negatives. All three are presently described.

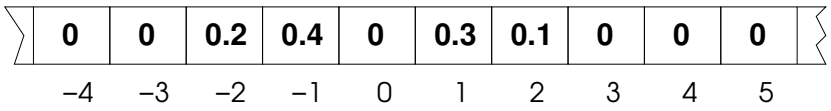
Significance set by False Positives

The implementation of this solution is made by first defining the wanted granularity of scores. Here the default is to include one fractional digit. Than a buffer is created with a size equal to the difference between the highest possible score while traversing the PWM and the lowest (with the specified granularity). This buffer is indexed by scores and contains probabilities on background. Populating the buffer is done by traversing the positions in the PWM (columns). The very first column is just used as a plain index by adding the background likelihood of each nucleotide to the index specified by the PWM (see figure 4.4.3). Thereafter the scores in the PWM can be viewed as offsets. Each index in the buffer containing a likelihood before this column is treated is moved according to the score and multiplied by the background probability for the current signal. The process is displayed in figure 4.4. The score buffer will always sum to 1, being probabilities. Conceptually this buffer can be thought of as the result of building a tree of the possible "paths" through the PWM, populating the nodes with the background probability. The score buffer would be the leaf nodes, indexed by score.

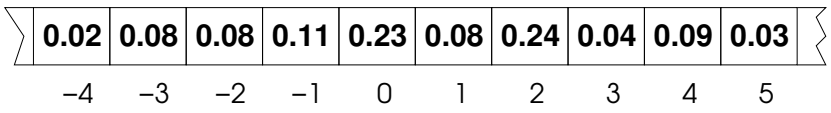


4.4.1: Background probabilities

4.4.2: PWM



4.4.3: Score buffer after first column



4.4.4: Score buffer after secound column

Figure 4.4: Visualisation of threshold calculation. Granularity keept at integers for simplicity

By adding the values of the buffer from right to left until one has achieved the target p-value, one can read the PWM threshold out of the index of the score buffer.

Significance by Balancing False Positives and False Negatives

In addition to the positional p-value, or the probability of a score equal or greater than t when the sequence segment is generated by the background model, this approach needs to calculate the probability of observing a t below given that the sequence segment was generated by the PWM. Call these values $\alpha(t)$ and $\beta(t)$ respectively. Then the threshold value t and thereby the significance is defined by the formulae in equation (4.3). The variable c in the equation allows for weighting the importance of the two error probabilities.

$$\alpha(t)c = \beta(t) \tag{4.3}$$

Implementation of this alternative is similar to what is displayed in figure 4.4. It needs another buffer for $\beta(t)$ which is populated with the likelihoods from the pwm instead of the background probabilities. These probabilities are calculated as in equation (3.1).

Significance by specifying True Positives

In this approach the support is specified, and the significance is set to allow this. The implementation is fairly simple. Sequences are scanned with each PWM and the highest score for each sequence recorded. Thereafter the scores are scores are sorted and the x'th highest score is used as threshold for this PWM. Significance is then calculated by creating the same probability buffer, as with the two other approaches and adding the probabilities above the threshold.

4.4 The Pareto Front

Since the Support and Significance of Motifs constitute a Multiobjective Optimization Problem a Pareto Front is used for storing and comparing motifs. For single motifs this would be a vector of significance indexed by support. For this vector to have only elements which are Pareto optimal the elements in the vector have to be uniformly lower with higher significance. The difference is displayed in figure 4.5. This is done by applying a "smoothing", resulting in each support inheriting the significance of any higher support with better significance. The storage for motifs in the search will henceforth be referred to as "pareto", although it is a data structure consisting of multiple pareto fronts.

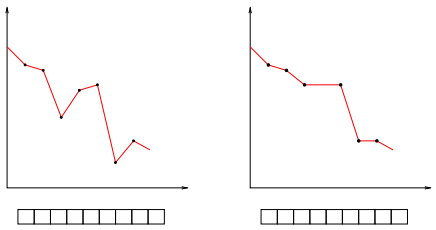


Figure 4.5: Visualisation of the effect of Pareto optimality. Here support plotted against significance.

Only the pareto for the target number of components is needed, but since the algorithm garantees optimality with lower number of components aswell ,a front for each

number of components is keept. This leads to one pareto front for each value of components. (see figure 4.6)

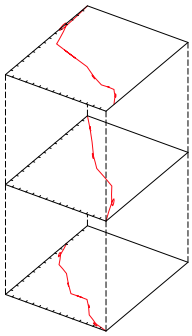


Figure 4.6: Visualisation of the Pareto Front for multiple number of components.

In addition - this project is also interested in distance. Therefore another dimension is introduced to the pareto. This is the distance classes. Each distance class is defined by a threshold on distance, depending on what kind of restraint is used. This results in an internal structure like displayed in figure 4.7.

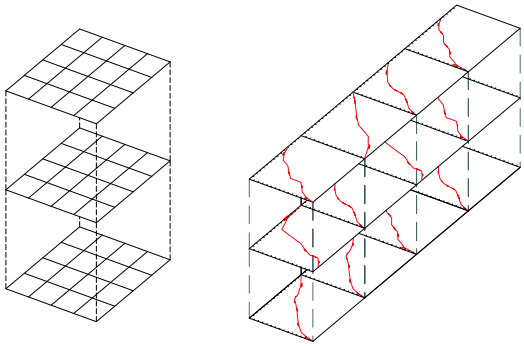


Figure 4.7: Visualisation of the Pareto Front for multiple distance classes and multiple sumber of components.

The pareto is resposible for evaluating the motifs added. The search itself is dumb with respect to whether the motif is acctually a part of the pareto front. The reasoning behind this apporoach lies with keeping all the pareto fronts below target number of components. When a motif is added, all the variants of the motif are checked against the contents of the pareto. If any of them qualify - a new motif is created from the variant and inserted. (*get_mav* in figure 4.1.) During the search there will be fewer of the variants, as the distance classes are pruned. (se sections 4.5 and 4.6).

The pareto can produce a string representation of itself and an evaluation string of the “best” motifs for each distance class. Functionality to write the pareto object to hard-drive for further post processing has been implemented but is not supported by the current stable version of jython.

4.5 Composite Search

Motifs are combined in an implicit depth first tree through recursion, see algorithm 4.5.1. (Visualisation in figure 4.6) Every time a motif is added to a module and it is not caught by pruning it is passed to the pareto (see section 4.5). It is up to the pareto to check whether it is more significant that what currently is located there, and take apropriate action.

Since the motifs are pre-sorted on significance, it is likely that the early combinations will be the best. This effect is somewhat diminished by the distance restraints. Since the composite motifs must confer to different distance classes, it is not enough that they keep a high support and are significant, but they must also avoid spanning out of their distance class. This makes the pruning less effective (see: 4.6), but as distance restriction is one of the main objectives this is accepted.

Algorithm 4.5.1: COMPOSITE MOTIF SEARCH($pareto, numcomp$)

```
procedure SEARCH(X)
  X.sort()
  c ← X.pop(0)
  BRANCH(c, X)

procedure BRANCH(c, X)
  if c.components < numcomp
    then { if PRUNE(c)
             then return (0)
           pareto.insert(c)
           for each motif ∈ X
             do { new_c ← c
                  new_c.add(X.pop(0))
                  if PRUNE(c)
                    then continue
                    else BRANCH(new_c, X) }
    else { if not PRUNE(c)
             then pareto.insert(c)
```

4.6 Pruning

There are two values that can be used for pruning, support and significance. Leaving variants out of the picture, the pruning is presently described. With support pruning is straight foreward. Since support cannot be gained by adding another motif, the present support is the maximum for the current combination. If this is lower then some user specified threshold the branch (of the search tree) can be safely pruned. With significance the picture is a bit more complex, since significance may indeed increase by adding motifs. For pruning to have any effect it must happen above the leafnodes, the higher up the more combinations can be left out of the search. To achieve this the potential significance of expanding the motif to the target number of component must

be evaluated, and compared to what is currently in the pareto. Given that the motifs are sorted in a decending fashion before the search starts a upper bound on the potential significance can be calculated as follows. Let $sign$ be the signifcance of the composite motif, $last$ be the significance of the last component added, $components$ be the target number of components, and cur_comp be the current number in the composite. If

$$sign + last * (components - cur\_comp) < pareto \tag{4.4}$$

where $pareto$ pareto is the current value in the pareto front, at the apropriate significance, the motif can be safely pruned. It is imposible for the motif to attain grater significance then what is currently in the pareto front, at this level of support. Given "pareto optimality" (see figure 4.5) we now that we know that significance for lower values of support have to be equal or higher. This along with the fact that the motif cannot gain support guarantees that this composite can have no expansion that will be a part of the pareto front.

Taking variants back into account, the acctuall pruning mechanisms are much the same. However the bitsets for the variants, and therefore the support will differ. This is because not all sequences will have the combination of motifs within the tightest distance window. Therefore the pruning has to be performed on a pr. variant basis. In this project this is achieved in the following manner every time *prune* is called in the search algorithm.

1. *prune* is called on the composite motif
2. The composite calls *calc_dist_class* on the distance object, retriving a list of minimal distances for each sequence. (algorithm 4.2.1)
3. *Check validity* is called on each of the composites variants, with the minimal distance list. This causes the variants to update themselves with new bitsets and significances. They also prune themselves against the pareto, using the approaches described above.

Thereby when the motif is passed to the pareto only valid variants that may achieve a place in the pareto front are present. When all the variants are pruned the entire branch can be quited. A visualisation of this can be seen in figure 4.6.

4.7 Evaluation of Modules

To evaluate the success of the algorithm in benchmarks one usually has to select only one module. To facilitate making this choice the composite motifs has a method for evaluating themselves. This is basically a binomial hypothesis test, where the null hypothesis is that the motif is generated by the background model. Equation 4.5 displays the actual formulae.[36] It results in a one-tailed significance, the p-value of observed support, that can be used to compare the modules.

$$\sum_{x=s}^{s_{max}} \binom{s_{max}}{x} p^x (1-p)^{s_{max}-s} \tag{4.5}$$

4.8 Choice of Language

The choice of languages are limited to the following.

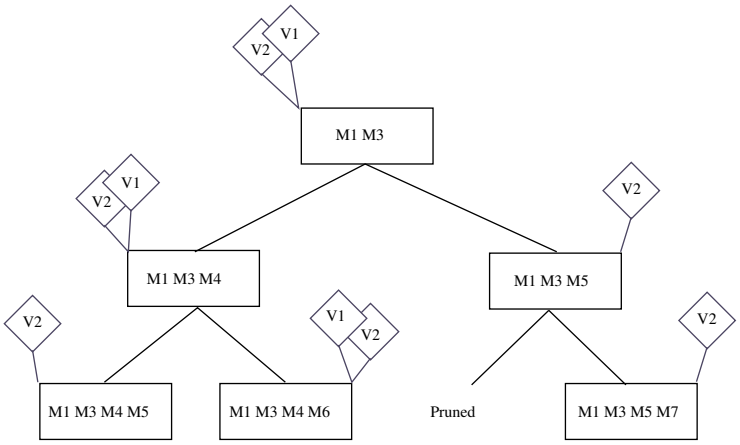


Figure 4.8: Visualisation of the search. Shows how Variants are pruned along the search

- Java
- C++
- Python
- Combination of the above

Reuseablility

One important aspect of the implementation of this tool is the reusabillity of the sourcecode. Reusabillity is made a principle point to facilitate for further development by others than the current developer.

On this point, Python is an obvious candidate. Python is known to be an "easy to use, and understand" language. This because of its easyly slooping learing curve, and the way it promotes an "easy to read" coding style.

Another point is that furture work on this tool is probably to be made at NTNU. NTNU has in many ways choosen Java as a primary language, wich should count in favour of Java.

Ease of development

To be able to produce as good a tool as possible in the allotted time, the ease of development is likely to have a major impact on the quality and diversity of the tool.

Also, unittests are preferred so that testing can be a part of the development process, and not something left to the end of the project that may be skipped. All three languages provide a unittesting framework, although python being an interpreted language lends itself easier to unittesting.

Efficiency

Efficiency will always be a major point in all tools such as this. A language unable to do the nessesary calculations efficiency must be dissqualified for choice of implementation language.

The Bitset

Most of the running time of this tool will be operation on bitsets. Therefore test have

been run on the efficiency of the different language candidates on excecution times on bitsets.

Language	Secounds
C++	0.17
Java	0.18
Python	1.31
Python with psyco	0.69

Table 4.1: The amount of time to do 1000 AND operations on 1000 bitsets 1000 times.

Appendix A.1 displays the results of these tests and describes how they where performed. (The code can be reviewed in A.2) Table 4.1 makes a summary of the results.

4.8.1 Conclusion

The implementation language decided upon is Jython. Thereby enabling the use of bitsets from Java while keeping the ease of development and potentially "easy to understand" source code from Python.

4.8.2 Language Limitations

The choice of Jython as implementation language has many benefits, but some drawbacks as well. This is mainly a result of Jython building on python 2.1, which is an edition of Python that can be said to be a bit "young". For this reason a *util* module has been created to implement frequently used functions that are not available in Python 2.1, or that are special for this project. This also contributes to making the code elsewhere more readable, since some complexity may be hidden here.

Some functionality however is lost with the language choice. It is currently impossible to serialize the pareto front (including motifs, bitsets and sequence information) bouth with Javas ObjectOutputStream and Pythons Pickle, due to the mix of Java and Python objects. Since this is believed to change with future Jython versions, code to achieve this is included.

Chapter 5

Usage of the diplom tool

This chapter cosist of an introduction to diplom in a practical sense.

5.1 Input

Diplom requires two inputfiles. The motifs and the sequences respectively. The sequences must conform to FASTA format. With every second line being a sequence and the others the sequence name preceded by a ">". An example of the format is displayed in listing 5.1.

Listing 5.1: Sequence File

```
>iYKL016C
CAAAGACATAGTCGAACAAAAATATAATTAGGTGATTCTGCAATTAAGTGCACGAAAAATTTCTTTT
AAATCCTAAGATTGTTCTTTTGCGTTACGTGTCAATATGAATATATAAACTTATATAGAAAAGTATT
GTACTAGTAGAAATATGTTTTACACCTACGCTAAGCAGCAAGTGGTAATTGGTGTATCTTTTACTTT
TTTGGGGGCATCAAGACAAATATCCAATCAAATCGAAGAGAAAATAATTCTTGTAACCCGCTCATTA
GTCACGTGGTCATCACGTGAGCATAATATATCACGTGATATACACCCATACCTCGTT
>iYBR248C
ATTAAAAAAACCTAAACTGGATACTGCTACTTCAATAGCTGCCTCTTTTCTTTTAAAACCTGATTGAG
TAGTCGTCGATATCAAAGGAATATCAACTTATGTATGTTTCGATGTCTGACTCTTTTCTCATGAATT
TTTCATTTTTTCATGATCACCTA
```

The motifs can be in one of two formats, regular expressions and PWMs. Motifs represented by regular expressions are much like the sequence file in that every second line is the name preceded by ">", and the other the regular expression. Lines starting with # are considered to be comments and ignored. The line with the pattern must also contain the match- and positional information see listing 5.2. The different columns in the pattern line are as follows:

1. The total number of hits
2. Number of hits in distinct sequences
3. The pattern, only the ambiguities in braces and the wildcard "." allowed.
4. The rest are pairs of sequence number and position in the sequence for each match.

Listing 5.2: Regular Expression Motifs

```
################################
#          FINAL RESULTS       #
################################
2        1        G[CG]A 1 2 1 12
3        2        GGC 0 0 0 14 1 11
2        2        C.C....TTT.G 0 3 1 1
2        2        C.....T[TA]T.G 0 5 1 1
2        1        G.C.....TTT.G 0 0 0 1
```

The PWM motifs are alignment matrices on the format in listing 5.3 Although it expects alignment matrices float counts are accepted. The columns represent positions in the pattern, and the rows are A,C,G and T respectively.

Listing 5.3: PWM Motifs

```
>AP1
0.0 0.0 46.0 8.0 10.0 2.0 29.0
0.0 0.0 0.0 20.0 5.0 29.0 13.0
0.0 42.0 0.0 19.0 7.0 0.0 0.0
51.0 9.0 5.0 4.0 29.0 20.0 9.0
>NFAT
4.0 3.0 6.0 15.0 14.0 12.0
0.0 0.0 0.0 0.0 0.0 1.0
6.0 12.0 9.0 0.0 1.0 0.0
3.0 0.0 0.0 0.0 0.0 1.0
```

5.2 Output

If the option "–outfile <file>" is specified two files will be written. They are names <file>.log and <file>.eval. The logfile contains a textual representation of the pareto, a list of the motifs used, and a list of motif dropped as a result of some threshold if any. The evalfile contains details for the composite motif chosen by the formulae in equation 4.5. One motif is chosen for each distance class.

5.3 Commandline and Configuration

The commandline options are described in the manpage in appendix B and will be omitted here. The entire configuration has not been made available as options however, and the remaining are described here. The following parameters reside in *conf.py* in the *system* package.

- min_support [1]
  This is a lower threshold on accepted coverage.
- revcomp [0]
  Boolean - Whether the reverse complement should be considered. This is equivalent to searching the other strand, orientation the same with respect three-prime / five-prime
- rev [0]
  Boolean - search for the motif with opposite orientation, on the same strand.
- comp [0]
  Boolean - search the opposite strand, orientation opposite with respect to tree-prime / five-prime
- wide [0]
  Boolean - preserve motifs with significance within 0.1 of the pareto front.

- max_seq_hits [30]
  A threshold on the maximum number of hits in one sequence. This is an optimisation to avoid too many permutations in the distance calculations.

Chapter 6

Results

The tool has been run on two different sets of datasets. For both sets only one composite motif is chosen from each distance class for evaluation. This is done as described in section 4.7. The datasets come with known sites and their positions, and python scripts for comparison to output.

Tests run using the "FPFN-approach" to setting significance and threshold gave no viable results, and none are therefore presented. Instead the more traditional approaches are used.

6.1 TransCompel Datasets

The first datasets are part of a benchmark created by members of the bioinfomatics group at the Norwegian Unitversity of Science and Technology (NTNU), and stems from the TransCompel[32] database. TransCompel is a specialised database of composite regulatory elements. That is a combination of two binding sites of two different transcription factors, which through this combination form a module with new regulatory properties.

The datasets have been created by retriving from TransCompel all modules with matches in five or more promotor regions have been selected, along with the sequences in witch they match. From this data three categories have been created. These are created by introducing noise in the form of additional matrices assumed not to be significant in the sequences. The categories are the percentage of matrices that should match the sequences. The percentages are set at 25, 50 and 100. A description of how the tests where run and the results themselves are presented in section 6.3.

6.2 Yeast Datasets

As with the TranCompel datasets, the yeast sets are provided by members of the bioinformatics group at NTNU. The sets consist of the match matrices and assorted decoys, their sequences and lists of known members for evaluation. The datasets stem from ChIP-CHIP experiments, where protein binding is recorded. This set has no positional information and is therefore evaluated on what motif presence alone. The results are presented in section 6.4.

6.3 TransCompel Results

A series of tests where run on the set with only match matrices, to get some idèa of how well the matrices could be discerned from the background. The reason for this is to evaluate what the sequence likelihood has to be to predict matches in a reasonable amount of sequences in a positive set. The test series consisted of running the toll with input p-value (sequence likelihood) set to 0.05, 0.1, 0.2, 0.3, 0.4 and 0.5. The results are presented in figure 6.3, with an extra set labelled *fasit* that represent the number of sequences actually containing matches, which is also the total number of sequences.

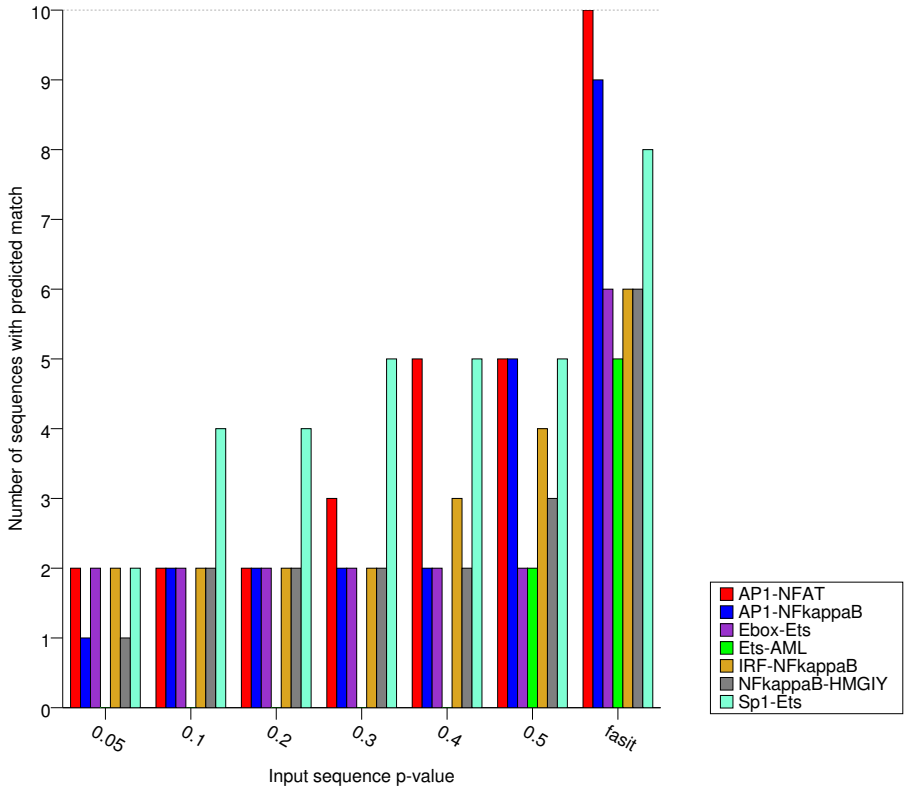


Figure 6.1: Achived support on a positive set, with different sequence likelihoods as input, and no restriction on distance.

As can be seen from the figure (6.3) the motifs differ only slightly from the background model. Specially the dataset *Ets-AML* in which there is no prediction until the sequence p-value is set to 0.5, which denotes a 50% chance of matching a pattern generated by the background.

With the transcompel datasets the evaluation is performed on a pr. nucleotide position basis. Figure 6.3 suggests the concept. The values nTP, nFP, nFN and nFP are added over the datasets and the statistics presented in chapter 3 are calculated and presented. Below each table of results over the combined dataset a barchart with the correlation coefficients of each single dataset is show for comparison. Please note that these charts differ in resolutions. A comlete set of results is found in appendix C.

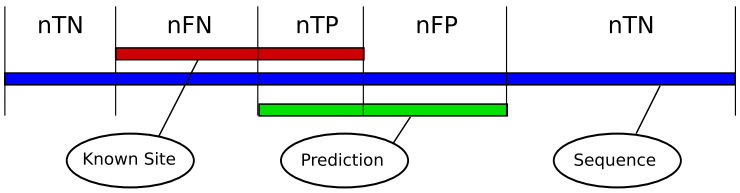


Figure 6.2: The position of the prediction and the known site displayed in relation to the sequence.

The test where also run with the different orientation options described in section 5.3, without any significant difference to the result, so those results have been omitted. The same is the case where the sequence files for all datasets where concatenated and used for background, as this had little or no effect on the probabilities in the background model.

6.3.1 100% sets

TransCompel results for TP significance approach

D. Class	nSn	nPPV	nSP	nPC	nCC	nASP
None	0.151	0.026	0.954	0.023	0.044	0.089
75	0.065	0.137	0.997	0.046	0.09	0.101
150	0.116	0.105	0.992	0.058	0.103	0.111
225	0.109	0.075	0.989	0.046	0.081	0.092

Table 6.1: Results from likelihoods set to 0.8, 0.9 and 1.0

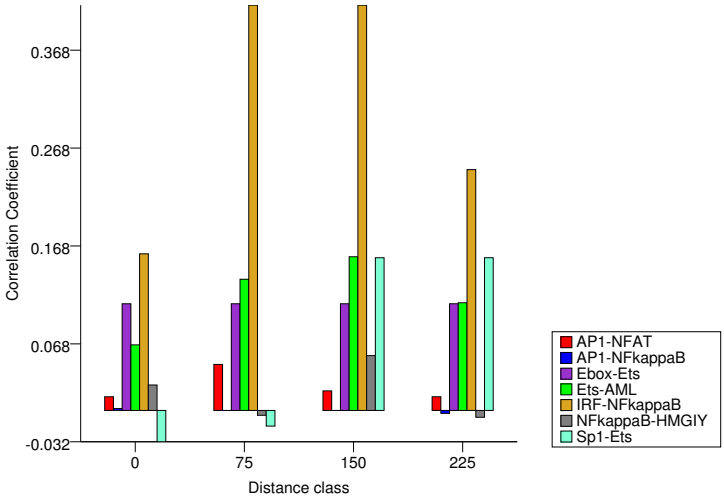


Figure 6.3: Correlation Coefficient for the Individual Datasets.

TransCompel results for FP significance approach

Liklihood	D. Class	nSn	nPPV	nSP	nPC	nCC	nASP
0.2, 0.3, 0.4	None	0.284	0.014	0.82	0.014	0.025	0.149
	75	0.051	0.253	0.999	0.044	0.11	0.152
	150	0.199	0.428	0.998	0.157	0.288	0.314
	225	0.244	0.314	0.995	0.159	0.271	0.279
0.4, 0.6, 0.8	None	0.299	0.063	0.962	0.055	0.122	0.181
	75	0.162	0.308	0.997	0.119	0.219	0.235
	150	0.168	0.205	0.994	0.102	0.179	0.186
	225	0.168	0.156	0.992	0.088	0.155	0.162

Table 6.2: Results on 100% set, FP significance

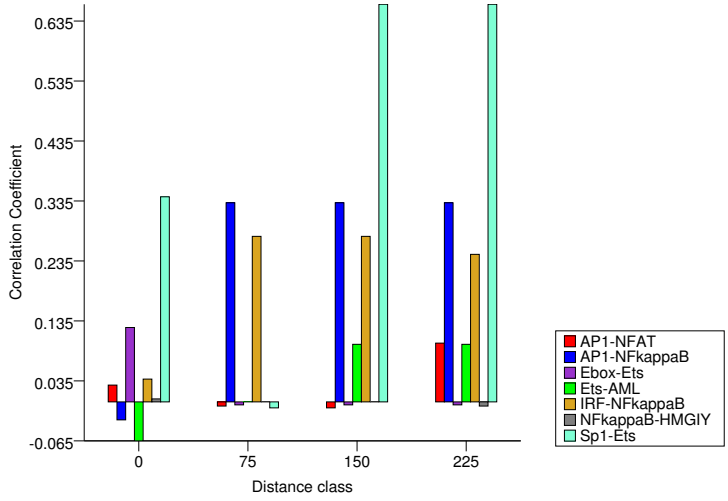


Figure 6.4: Correlation Coefficients for likelihoods at 0.2, 0.3 and 0.4

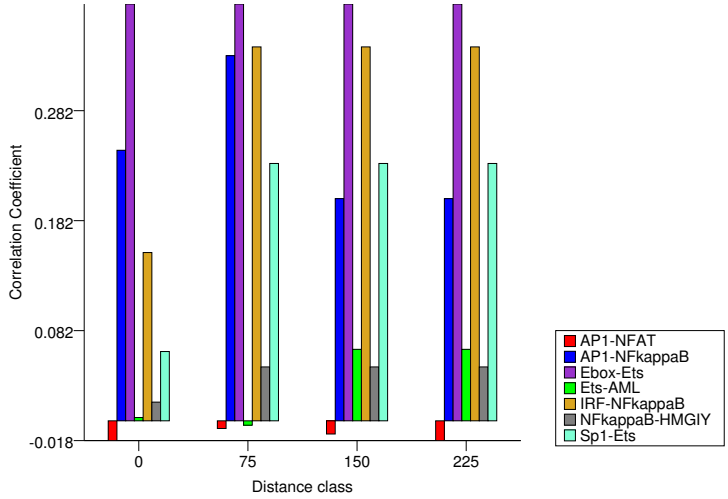


Figure 6.5: Correlation Coefficient for likelihoods at 0.4, 0.6 and 0.8

6.3.2 50% sets

TransCompel results for TP significance approach

D. Class	nSn	nPPV	nSP	nPC	nCC	nASP
None	0.216	0.033	0.948	0.029	0.065	0.124
75	0.12	0.185	0.996	0.079	0.144	0.153
150	0.105	0.069	0.989	0.044	0.076	0.087
225	0.099	0.051	0.985	0.035	0.06	0.075

Table 6.3: Results from likelihoods set to 0.8, 0.9 and 1.0

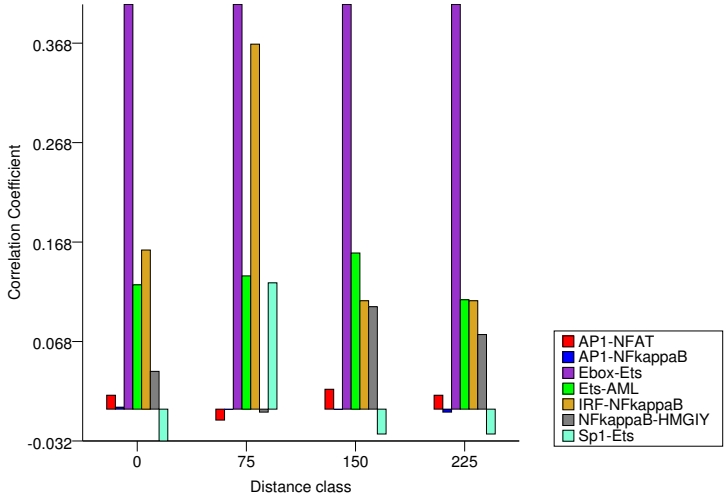


Figure 6.6: Correlation Coefficient for the Individual Datasets.

TransCompel results for FP significance approach

Likelihoods	D. Class	nSn	nPPV	nSP	nPC	nCC	nASP
0.2, 0.3, 0.4	None	0.3	0.038	0.931	0.035	0.085	0.169
	75	0.054	0.15	0.997	0.041	0.086	0.102
	150	0.214	0.294	0.995	0.141	0.245	0.254
	225	0.23	0.265	0.994	0.14	0.24	0.247
0.4, 0.6, 0.8	None	0.321	0.045	0.941	0.041	0.1	0.183
	75	0.127	0.246	0.997	0.091	0.172	0.186
	150	0.122	0.133	0.993	0.068	0.12	0.128
	225	0.121	0.102	0.991	0.059	0.103	0.112

Table 6.4: Results on 50% set, FP significance

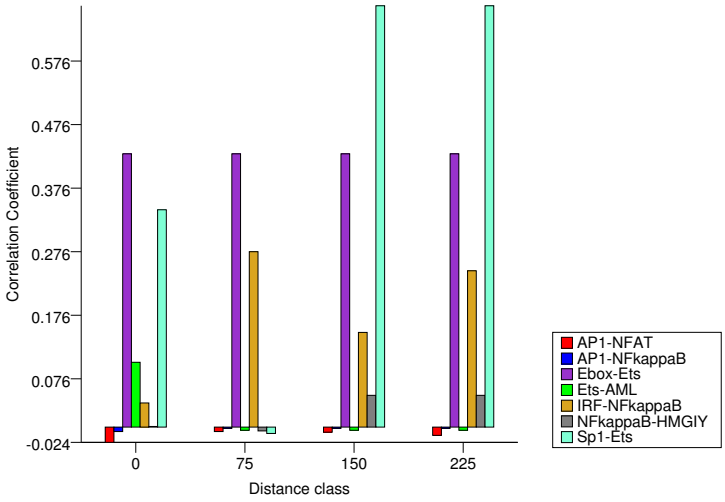


Figure 6.7: Correlation Coefficients for likelihoods at 0.2, 0.3 and 0.4

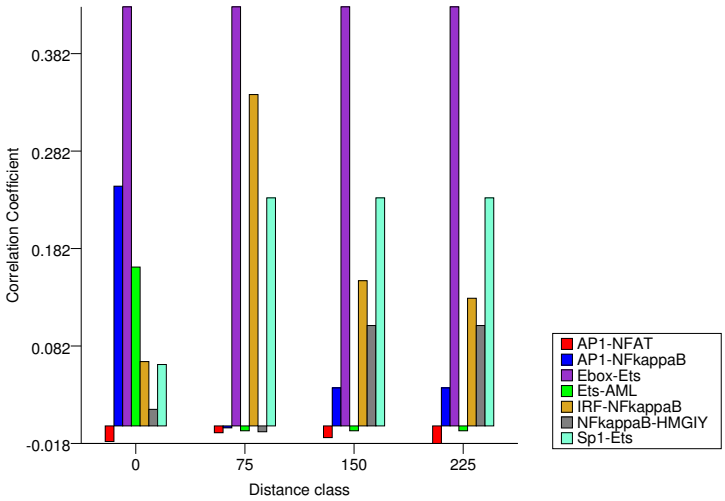


Figure 6.8: Correlation Coefficient for likelihoods at 0.4, 0.6 and 0.8

6.3.3 25% set

TransCompel results for TP significance approach

D. Class	nSn	nPPV	nSP	nPC	nCC	nASP
None	0.424	0.038	0.91	0.036	0.105	0.231
75	0.131	0.178	0.995	0.082	0.147	0.155
150	0.116	0.069	0.987	0.045	0.08	0.092
225	0.107	0.044	0.982	0.032	0.058	0.076

Table 6.5: Results from likelihoods set to 0.8, 0.9 and 1.0

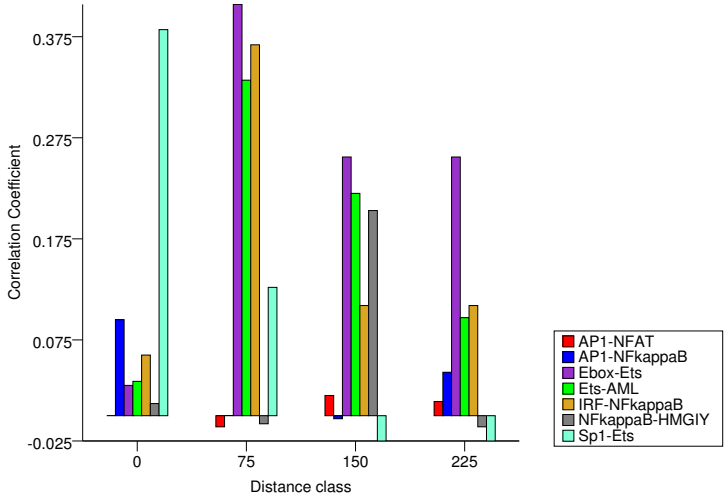


Figure 6.9: Correlation Coefficient for the Individual Datasets.

TransCompel results for FP significance approach

Likelihoods	D. Class	nSn	nPPV	nSP	nPC	nCC	nASP
0.2, 0.3, 0.4	None	0.332	0.032	0.91	0.03	0.078	0.182
	75	0.037	0.121	0.998	0.029	0.062	0.079
	150	0.223	0.255	0.994	0.135	0.232	0.239
	225	0.196	0.162	0.991	0.097	0.17	0.179
0.4, 0.6, 0.8	None	0.363	0.03	0.898	0.028	0.079	0.196
	75	0.138	0.25	0.996	0.098	0.181	0.194
	150	0.144	0.155	0.993	0.081	0.143	0.149
	225	0.133	0.084	0.988	0.054	0.096	0.108

Table 6.6: Results on 25% set, FP significance

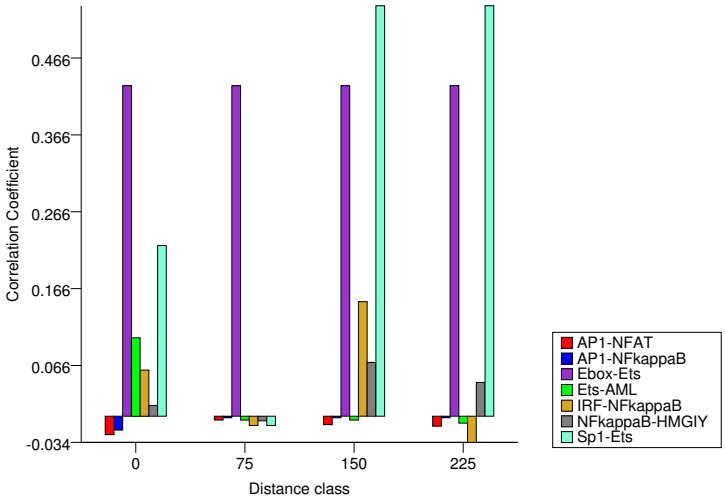


Figure 6.10: Correlation Coefficients for likelihoods at 0.2, 0.3 and 0.4

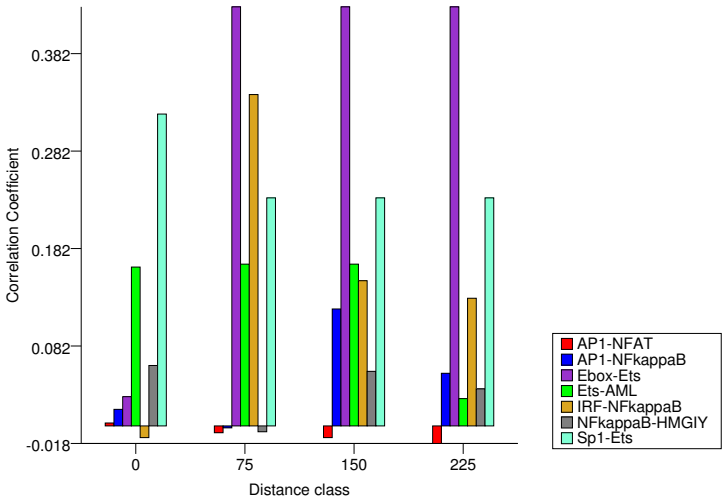


Figure 6.11: Correlation Coefficient for likelihoods at 0.4, 0.6 and 0.8

6.4 Yeast Results

In the results displayed the values sTP, sFP, sFN and sTN have been combined from all the datasets, and the evaluation values calculated from there as if it was one big dataset. This arrangement was suggested by Tompa et. al. [28]. In the results with yeast the values are not connected to nucleotide positions but rather just the presence of the motif in the sequence.

Table Explenation

The first two columns in the result tables are the method used to set significance and the distance class or the size of the "distance restriction window". The next six columns are the evaluation values presented in section 3.6. Some of the tables has an additional column labelled "N". This column is the number of datasets where no composite motif could be discerned within the restrictions.

Yeast results for likelihoods 0.05, 0.1, 0.2

Sign	D. Class	sSn	sPPV	sSP	sPC	sCC	sASP	Sign	N
FP	None	0.364	0.45	0.954	0.252	0.35	0.407	8.297	0
	50	0.32	0.36	0.945	0.204	0.28	0.34	8.97	3
	100	0.316	0.392	0.951	0.212	0.295	0.354	8.993	0
	200	0.392	0.482	0.957	0.276	0.384	0.437	8.471	0
TP	None	0.325	0.353	0.941	0.204	0.276	0.339	17.426	0
	50	0.209	0.235	0.933	0.124	0.150	0.222	12.697	1
	100	0.283	0.307	0.938	0.173	0.229	0.295	17.604	0
	200	0.284	0.297	0.934	0.170	0.223	0.291	18.392	0

Table 6.7: Results for the Yeast dataset (see section 6.4). The results displayed where achieved with likelihood set to 0.05,0.1 and 0.2, and a maximum number of hits for one motif in one sequence of 10.

Yeast results for likelihoods 0.2, 0.4, 0.6

Sign	D. Class	sSn	sPPV	sSP	sPC	sCC	sASP	Sign
FP	None	0.319	0.362	0.944	0.204	0.279	0.341	4.071
	50	0.199	0.214	0.928	0.115	0.131	0.207	4.917
	100	0.241	0.269	0.935	0.145	0.185	0.255	4.435
	200	0.29	0.335	0.942	0.184	0.248	0.312	4.314
TP	None	0.348	0.373	0.942	0.22	0.299	0.361	10.746
	50	0.357	0.379	0.944	0.225	0.309	0.368	10.752
	100	0.309	0.341	0.942	0.194	0.263	0.325	10.61
	200	0.321	0.353	0.942	0.202	0.275	0.337	10.581

Table 6.8: Results for the Yeast dataset (see section 6.4). The results displayed where achieved with likelihood set to 0.2, 0.4 and 0.6, and a maximum number of hits for one motif in one sequence of 10.

Yeast results for likelihoods 0.4, 0.6, 0.8

Sign	D. Class	sSn	sPPV	sSP	sPC	sCC	sASP	Sign
FP	None	0.344	0.373	0.942	0.218	0.297	0.359	2.567
	50	0.14	0.15	0.926	0.078	0.069	0.145	2.73
	100	0.224	0.243	0.932	0.132	0.162	0.233	2.609
	200	0.22	0.237	0.93	0.129	0.155	0.229	2.282
TP	None	0.293	0.311	0.935	0.178	0.235	0.302	7.222
	50	0.336	0.342	0.939	0.204	0.278	0.339	7.736
	100	0.324	0.354	0.942	0.204	0.277	0.339	7.334
	200	0.31	0.326	0.936	0.189	0.252	0.318	6.939

Table 6.9: likelihood 0.4, 0.6 and 0.8, maximum hits pr. sequence 10

Chapter 7

Discussion

7.1 TransCompel

As can be seen from figure 6.3 the motifs appear, to the algorithm, very similar to what can be generated by the background model. That is, the algorithm has difficulties finding the matches on single motifs. This can be caused by several factors. The PWMs used in the datasets are combinations of experimentally validated factors, and may be a bit to general to find good matches. Additionally the datasets allow overlaps, while the algorithm does not. This may cause combinations of two motifs to loose a significant part of the single motifs' support. The literature argues that a higher order markov chain as background yields a positive effect, but the factor is not believed to be large.

Reviewing the actual results a fairly small but positive correlation coefficient is present throughout the combined datasets. Much of the reason for the correlation been low is believed to lie with the problem of attaining matches for the single motifs. Further evidence for this view can be gained from graphs in figures 6.3-6.11. The results are based on combinations of datasets into what can be thought of as one big set. The graphs display the correlation for the individual datasets. It can be seen for the graphs that some datasets lack correlation entirely, as no modules where found for the set.

In the results with all approaches the highest correlation is found with restriction on distance. This is as expected in this dataset, since it consists of TFBS that are close together. The sensitivity, specificity and the positive predictive value suggest that this is due to an increase in true negatives and a decrease in false positives rather then more true positives. This does not however diminish the value of the restriction. A related effect can be seen from the graphs (figures 6.3-6.11). In many cases the correlation is equal for several distance classes. The reason for this is that algorithm 4.2.1 chooses the least possible distance without regard for what the distance classes are. This is discussed further in section 7.3.

7.2 Yeast

The yeast datasets are based on biological ChIP-CHIP experiments only and the expectations to the results therefore must be moderate. This also excludes positional information, for this reason evaluation is based on presence alone. As can be seen in section 6.4 correlations are all positive, and generally above 0.15.

In this dataset the correlation is not consistently greater with distance restrictions. Studying the results separately however does reveal a trend. By looking at the results for "FP-significance" approach, it seems that when the input p-values are low (most reliable results) the correlation, specificity and sensitivity are highest when a distance restriction of 200 nucleotides is imposed. For the "TP-significance" approach the likelihood is the factor of sequences in which we want matches. For this approach the smallest "likelihoods" are too small as reflected in the significance, if compared to the significances of results in "FP-approach". Keeping these considerations in mind it is plausible to assume that the data adhere to some constraint on distance.

7.3 Algorithm

As mentioned in section 7.1 the search effectively chooses the tightest possible window for each composite motif. This is a result of algotirhm 4.2.1 wich allways returns the tightest possible parameters. The choice might be a cause for consern, since causes the chosen pattern to be the same for all distance classes in wich it will fit. A plausible remedy might be to choose the pattern closest to the window restriction. This would in the worst case resolve the problem with no discrimination between the distance classes, and possibly enable use of the tool to evaluate distance restrictions based on the result. This could be done much in the same way as it currently is, by returning a list of tuples instead. The tupples would then consist of the widest window within the various distance classes. This would discriminate more between distance classes while still keeping the advantage of doing it once for all variants of the motif.

Bibliography

[1] Wikipedia.
http://en.wikipedia.org/wiki/DNA.

[2] Abbreviations and symbols. *European Journal of Biochemistry*, 74:1–3, Mar. 1977.

[3] I.S Mian K. Sjolander D. Haussler A. Krogh, M. Brown. Hidden markov models in computational biology. *Journal of Molecular Biology*, 236, 1994.

[4] E. Gößling I. Reuter E. Cheremushkin O.V. Kel-Margoulis E. Wingender A.E. Kel. Match: a tool fro searching transctiption factor binding sites in dna sequences. *Nucleic Acids Res.*, 31, 2003.

[5] P. Green AFA. Smit, R. Hubley. Repeatmasker open-3.0.
http://www.repeatmasker.org.

[6] Pär Engström Wyeth Wasserman Boris Lenhard Albin Sandelin, Wynand Alkema. Jaspar: an open access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.*, 32, 2004.

[7] T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

[8] K. Blekas, D. I. Fotiadis, and A. Likas. Protein sequence classification using probabilistic motifs and neural networks. In *Procidings of the 13th International Conference on Artificial Neural Networks*, 2003. Istanbul.

[9] K. Blekas, D. I. Fotiadis, and A. Likas. Motif-based protein sequence classification using neural networks. *Journal of Computational Biology*, 12:64–82, 2005.

[10] B.D. Pfeiffer P. Tomancak S. Celniker-M. Levine G.M. Rubin M.B. Eisen B.P. Berman, Y. Nibu. Exploiting transcription factor binding site clustering to indentify cis-regulatory modules involved in pattern formation in the drosophila genome. *PNAS*, 99:757–762, 2001.

[11] Andrew A. Reilly Charles E. Lawrence. An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function, and Genetics*, 7:41–51, 2004.

[12] Carlos A Coello Coello. An updated survey of ga-based multiobjective optimization techniques. *ACM Computing Surveys*, 32, 2000.

[13] G.D. Stormo D.G. Thakurta. Identifying target sites for cooperatively binding factors. *Bioinformatics*, 17:608–621, 2001.

[14] D. Eveillard, A. Larhlimi, D. Ropers, S. Billaut, and S. Peyrefitte. Koalab: A new method for regulatory motif search. illustration on alternative splicing regulation in hiv-1. `http://www.loria.fr/equipes/modbio/KOALAB.html`.

[15] Finn Drabløs Geir Kjetil Sandve. Generalized composite motif discovery. 2005.

[16] W. N. Grundy, T. L. Bailey, C. P. Elkan, and M. E. Baker. Meta-meme: motif-based hidden markov models of protein families. *Computer Applications in the Biosciences*, 13:397–406, 1997.

[17] K. Gulukota and C. DeLisi. Neural network method for predicting peptides that bind major histocompatibility complex molecules. *Methods Molecular Biology*, pages 201–209, 2001.

[18] G. Z. Hertz and G. D. Stormo. Identifying dna and protein patterns with statistically significant alignments of multiple sequences. 15:563–577, 1999.

[19] J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church. Computational identification of cisregulatory elements associated with groups of functionally related genes in saccharomyces cerevisiae. *Journal of Molecular Biology*, 296:1205–1219, 2000.

[20] R. Hughey and A. Krogh. Hidden markov models for sequence analysis: extension and analysis of the basic method. *Computer Applications in the Biosciences*, 12:95–107, 1996.

[21] J. Collado-Vides J. van Helden, A. F. Rios. Discovering regulatory element in non coding sequences by analysis of spaced dyads. *Nucleic Acids Res.*, 28:1808–1818, 2000.

[22] I. Jonassen. Efficient discovery of conserved patterns using a pattern graph, 1996. Technical Report 118, Department of Informatics, University of Bergen, Norway.

[23] Gabriel Kreiman. Identification of sparesely distributed clusters of cis-regulatory elements in sets of co-expressed genes. *Nucleic Acids Res.*, 32, 2004.

[24] E. Rubin L. Pennacchio. Genomic strategies to identify mamalian regulatory sequences. *Nature Reviews Genetics*, 17:100–109, 2001.

[25] X. Liu, D. L. Brutlag, and J. S. Liu. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes,. *Pac Symp Biocomput*, pages 127–138, 2001.

[26] R. Guigò M. Burset. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.

[27] M. Endrizzi B. Birren-E. Lander M. Kellis, N. Patterson. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423:241–254, 2003.

[28] Timothy L Bailey George M Church-Bart De Moor Eleazar Eskin Alexander V Favorov Martin C Frith Yutao Fu W James Kent Vsevolod J Makeev Andrei A Mironov William Stafford Noble Giulio Pavesi Graziano Pesole Mireille Rêgnier Nicolas Simonis Saurabh Sinha Gert Thijs Jacques van Helden Mathias Vandenbogaert Shiping Weng Christopher Workman Chun Ye Zhou Zhu Martin Tompa, Nan Li. Assessing computational tools for the discovery of transcrion factor finding sites. *Nature Biotechnology*, 23, 2005.

[29] G.M. Church A.M. Michelson M.S. Halfon, Y. Grad. Computation-based discovery of related transcriptional regulatory modules and motifs using an experimentally validated combinatorial model. *Genome Research*, 12:1019–1028, 2002.

[30] V. Bulliard L. Cerutti E. De Castro P.S. Langendijk-Genevaux M. Pagni C.J.A. Sigrist N. Hulo, A. Bairoch. The prosite database. *Nucleic Acids Res.*, 34:D227–D230, 2006.

[31] A. Osyczka. Multicriteria optimization for engineering design. *Design Optimization*, pages 193–227, 1985.

[32] I. Reuter E.V. Deineko E. Wingender O.V. Kel-Margoulis, A.E. Kel. Transcompel: a database on composite regulatory elements in eukaryotic genes. *Nucleic Acids Res.*, 30:332–334, 2002.

[33] S.H. Sze P. Pevzner. Combinatorial approaches to finding subtle signals in dna sequences. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 269–278, 2000.

[34] V. Pareto. Cours deconomie politique. 1896.

[35] A. Krogh R. Houghey. Hidden markov models for sequence analysis: extension and analysis of the basic method. *Computer Applications in the Biosciences*, 12:95–107, 1996.

[36] S. L. Myers R. Walpole, R. H. Myers. *Probability and Statistics for Engeneers and Scientists*. Sixth edition, 1998.

[37] Roded Sharan Ron Shamir Yosef Shiloh Ran Elkon, Chaim Linhart. Genome-wide in silico identification of transcriptional regulators controlling the cell cycle in human cells. *Genome Research*, 13:773–780, 2003.

[38] I. Rigoutsos and A. Floratos. Motif discovery without alignment or enumeration. In *Proceedings of the secound annual international conference on computational molecular biology*, pages 221–227, 1998.

[39] S. Sinha. Composite motifs in promoter regions of genes: models and algorithms. General Report, University of Washington.

[40] H. O. Smith, T. M. Annau, and S. Chandrasegaran. Finding sequence motifs in groups of functionally related proteins. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 826–830, 1990.

[41] Martin Vingon Sven Rahmann, Tobias Müller. On the power of profiles for transcription factor binding site detection. *Statistical Applications in Genetics and Molecular Biology*, 2, 2003.

[42] Chen X. Hehl R. Karas H. Liebich I. Matys V. Meinhardt-T. Prüß M. Reuter I. Schacherer F Wingender, E. Transfac: an integrated system for gene expression regulation. *Nucleic Acids Res.*, 28:316–319, 2000.

[43] Øystein Lekang. Gige is gcmd expanded.

Appendix A

Efficiency Test on Bitsets

A.1 Results and Description

Tests where run on a Intel(R) Pentium(R) 4 CPU 3.00GHz running Debian GNU/Linux 2.4.27 for 686-smp. C++ compiler was run with the "-O3" optimizer, and one of the python code uses the "psyco" library as can be seen in the source code.

For the timing of the C++ code integers are used in so the values are not inserted by the precompiler.

Language	Compiler	Version	Time in Secounds			
			1st	2cound	3rd	Average
C++	GCC/G++	3.3.6	0.17	0.16	0.17	0.17
Java	Blackdown	1.4.2-02	0.175	0.183	0.181	0.18
Python (psyco)	Python	2.3.5	0.69	0.69	0.68	0.69
Python	Python	2.3.5	1.31	1.32	1.30	1.33

Table A.1: Efficiencytest of bitsets

A.2 Source Code

Listing A.1: Java Bitset Test

```
1
3  import java.util.BitSet;
4  import java.util.Random;
8  class test{
9      static int ANT_BITSET = 1000;
10     static int STR_BITSET = 1000;
11     static int ANT_GANGER_AND = 1000;
12     static BitSet[] data;
```

```
14      public static void main(String[] args){
15
16          data = generate();
17          tid(data);
18
19      }
20
21      static BitSet[] generate(){
22
23          Random ran = new Random();
24          BitSet[] retur = new BitSet[ANT_BITSET];
25
26          for(int j = 0; j < ANT_BITSET; j++){
27              BitSet x = new BitSet(STR_BITSET);
28              for(int i = 0; i < STR_BITSET; i++){
29                  if (ran.nextBoolean()) {
30                      x.set(i);
31                  }
32              }
33              retur[j] = x;
34          }
35
36
37          return retur;
38
39      }
40
41
42      static void tid(BitSet[] data){
43
44          long before, after;
45
46          before = System.currentTimeMillis();
47          for (int j = 0; j < ANT_GANGER_AND; j++){
48              for (int i = 0; i < ANT_BITSET; i++){
49                  data[i].and(data[ANT_BITSET -i -1]);
50
51              }
52          }
53          after = System.currentTimeMillis();
54          System.out.print("start: ");
55          System.out.print(before);
56          System.out.print(" slutt: ");
57          System.out.print(after);
58          System.out.print(" totalt: ");
59          System.out.print(after-before);
60          System.out.print(" sek: (diff(millisek)/1000) ");
61          System.out.print((float)(after-before)/1000.0);
62
63          System.out.println();
64
65
66
67      }
```

```
68 }
```

Listing A.2: C++ Bitset Test

```
1  #ifndef BITSETTEST
2  #define BITSETTEST
3  #include <vector>
4  #include <iostream>
5  #include <ctime>
6  #include <cstdlib>
7  #include <boost/dynamic_bitset.hpp>

9  using boost::dynamic_bitset;
10 using namespace std;

12 vector<dynamic_bitset<> > generate();
13 void tid(vector<dynamic_bitset<> > data);
14 int main(int argc, char* argv[]);

16 #endif

18 int ANT_BITSET = 1000;
19 int STR_BITSET = 1000;
20 int ANT_GANGER_AND = 1000;

22 vector<dynamic_bitset<> > generate() {
23     vector<dynamic_bitset<> > retur;

25     for(int j=0; j < ANT_BITSET; j++){
26         dynamic_bitset<> x(STR_BITSET);

28         for(int i=0; i < STR_BITSET; i++){
29             x[i] =  abs(ceil(2*rand() / (RAND_MAX + 1.0)));
30         }
31         retur.push_back(x);
32     }

34     return retur;
35 }

37 void tid(vector<dynamic_bitset<> > data) {

39     clock_t fortid, ettertid;
40     fortid = clock();
41     for(int j = 0 ; j < ANT_GANGER_AND; j++){
42         for(int i = 0; i < ANT_BITSET; i++) {

44             data[i] = data[i] & data[ANT_BITSET - i -1] ;

46         }
47     }
48     ettertid = clock();
49     cout << "start: " << fortid << " stopp: " << ettertid
50         << " totalt: " << ettertid-fortid
```

```
51          << " sek: (diff/CLOCKS_PR_SEC) "
52          << (float) (ettertid-fortid) / (3 * CLOCKS_PER_SEC) << '\n';

54  }

56  int main(int argc, char* argv[]) {

58      vector<dynamic_bitset<> > data = generate();
59      tid(data);

61  }
```

Note: With POSIX compliant compilers CLOCKS_PER_SEC is defined as 1000000.

Listing A.3: Python Bitset Test

```
1   import Numeric
2   import random
3   import time
4   try:
5       import psyco
6       psyco.profile()
7   except:
8       print 'Psyco not found, ignoring it'

10  ANT_BITSET = 1000
11  ANT_GANGER_AND = 1000
12  STR_BITSET = 1000

14  class bitset:

16      def __init__(self, liste):
17          self._bitset = Numeric.numarray(liste, Numeric.numarray.Bool)

19      def __getitem__(self, y):
20          return self._bitset[y]

22      def __setitem__(self, i, f):
23          self._bitset[i] = f


26      def AND(self, bitset):
27          if len(self._bitset) != len(bitset._bitset):
28              return -1

30          for i,f in enumerate(self._bitset):
31              pass

33          dir(self._bitset[0])

35  def main():
36      data = generate()
37      tid(data)

39  def generate():
```

```
40      retur = []
41      for f in range(ANT_BITSET):
42          a = [random.randint(0,1) for f in range(STR_BITSET)]
43          retur.append(a)

45      return retur


48  def tid(data):
49      i = 0
50      test = data[:]
51      before = time.clock()
52      while i < ANT_GANGER_AND:
53          for i, f in enumerate(data):
54              data[i] = [j.__and__(data[STR_BITSET -1 -i][k]) \
55                  for k,j in enumerate(f)]
56          i += 1

58      after = time.clock()
59      print "start: "+ str(before) + " stopp: " + str(after) \
60              + " totalt: " + str(after - before)

62  if __name__ == "__main__":
63      main()
```

Appendix B

Manual page

NAME

diplom – program to discover sets of motif combinations with distance restrictions.

SYNOPSIS

diplom [*options*] sequencefile patternfile

DESCRIPTION

This manual page documents briefly the **gige** program.

diplom is based on "General Composite Motif Discovery", and is a tool to find sets of motifs that occur together in sequences of a positive data set. Hereby bettering the significance and sensitivity of the motif. It also finds the best within every given distance window, if specified.

OPTIONS

This program follow the usual GNU command line syntax, with long options starting with two dashes (‘-’). All options must be stated before the arguments. A summary of options is included below.

–h, ––help
Show summary of options.

–b, ––background <file>
Use <file> to generate background model. If not specified the background model is created from the sequence file.

–d, ––distances d1,d2,d3,...
Comma separated lits of the distance restrictions you want diplom to run. The default is one distance class, half the size of the average sequence length.

–s, ––signtype FP | TP | FP,FN
The mode of significance calculation wanted.
FP - Default, likelihoods treated as target p-values.
TP - likelihoods treated as coverage. values below 1 are treated as factors of sequence count.
FP,FN - likelihoods are treated as a factor C in the equation "ErrorI * C = ErrorII".

–l, ––likelihoods l0,l1,l2,..
Comma separated list of likelihoods. The evaluation of the values depend on the signtype. Default is that they are interpreted as target p-values.

–o, ––outfile <basename>
This spesifies the basename of outputfiles. There will be two. One log file and one eval file. The log file contains an ascii representation of the pareto, and the eval file contains the information on the best composite for each distance class.

–c, ––components <number>
The target number of components in a composite motif.

–v Verbose output to screen. Yealds more information on the progress of the algorithm.

AUTHOR

diplom was written by Øystein Lekang, for his masters degree.

This manual page was written by Øystein Lekang <lekang@idi.ntnu.no>.

Appendix C

Complete TransCompel Results

-s TP -l 0.8,0.9,1.0

Listing C.1: 100% Combined

```
distance: 0
nFN: 1208 nFP: 8067 nTN: 167137 nTP: 215
nASP: 0.089 nCC: 0.044 nPC: 0.023
nPPV: 0.026 nSP: 0.954 nSn: 0.151

distance: 75
nFN: 1278 nFP: 559 nTN: 166474 nTP: 89
nASP: 0.101 nCC: 0.09 nPC: 0.046
nPPV: 0.137 nSP: 0.997 nSn: 0.065

distance: 150
nFN: 1199 nFP: 1333 nTN: 165711 nTP: 157
nASP: 0.111 nCC: 0.103 nPC: 0.058
nPPV: 0.105 nSP: 0.992 nSn: 0.116

distance: 225
nFN: 1290 nFP: 1953 nTN: 173226 nTP: 158
nASP: 0.092 nCC: 0.081 nPC: 0.046
nPPV: 0.075 nSP: 0.989 nSn: 0.109
```

Listing C.2: 100% Distance 75

```
AP1—NFAT
nFN: 144 nFP: 161 nTN: 18060 nTP: 9
nASP: 0.056 nCC: 0.047 nPC: 0.029
nPPV: 0.053 nSP: 0.991 nSn: 0.059

AP1—NFkappaB
nFN: 388 nFP: 0 nTN: 82997 nTP: 0
nASP: * nCC: * nPC: 0.0
nPPV: * nSP: 1.0 nSn: 0.0

Ebox—Ets
nFN: 134 nFP: 20 nTN: 13006 nTP: 7
nASP: 0.155 nCC: 0.109 nPC: 0.043
nPPV: 0.259 nSP: 0.998 nSn: 0.05

Ets—AML
nFN: 80 nFP: 105 nTN: 13590 nTP: 15
nASP: 0.142 nCC: 0.134 nPC: 0.075
nPPV: 0.125 nSP: 0.992 nSn: 0.158

IRF—NFkappaB
nFN: 199 nFP: 16 nTN: 9766 nTP: 58
nASP: 0.505 nCC: 0.414 nPC: 0.212
nPPV: 0.784 nSP: 0.998 nSn: 0.226

NFkappaB—HMGIY
nFN: 73 nFP: 71 nTN: 15103 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.995 nSn: 0.0

Sp1—Ets
nFN: 260 nFP: 186 nTN: 13952 nTP: 0
nASP: 0.0 nCC: −0.016 nPC: 0.0
nPPV: 0.0 nSP: 0.987 nSn: 0.0
```

Listing C.3: 100% Distance 150

```
AP1—NFAT
nFN: 108 nFP: 596 nTN: 17661 nTP: 9
nASP: 0.046 nCC: 0.02 nPC: 0.013
nPPV: 0.015 nSP: 0.967 nSn: 0.077

AP1—NFkappaB
nFN: 388 nFP: 0 nTN: 82997 nTP: 0
nASP: * nCC: * nPC: 0.0
nPPV: * nSP: 1.0 nSn: 0.0

Ebox—Ets
nFN: 134 nFP: 20 nTN: 13006 nTP: 7
nASP: 0.155 nCC: 0.109 nPC: 0.043
nPPV: 0.259 nSP: 0.998 nSn: 0.05

Ets—AML
nFN: 71 nFP: 197 nTN: 13498 nTP: 24
nASP: 0.181 nCC: 0.157 nPC: 0.082
nPPV: 0.109 nSP: 0.986 nSn: 0.253

IRF—NFkappaB
nFN: 199 nFP: 16 nTN: 9766 nTP: 58
nASP: 0.505 nCC: 0.414 nPC: 0.212
nPPV: 0.784 nSP: 0.998 nSn: 0.226

NFkappaB—HMGIY
nFN: 64 nFP: 260 nTN: 14914 nTP: 9
nASP: 0.078 nCC: 0.056 nPC: 0.027
nPPV: 0.033 nSP: 0.983 nSn: 0.123

Sp1—Ets
nFN: 235 nFP: 244 nTN: 13869 nTP: 50
nASP: 0.173 nCC: 0.156 nPC: 0.095
nPPV: 0.17 nSP: 0.983 nSn: 0.175
```

Listing C.4: 100% Distance 225

```
AP1—NFAT
nFN: 108 nFP: 765 nTN: 17492 nTP: 9
nASP: 0.045 nCC: 0.014 nPC: 0.01
nPPV: 0.012 nSP: 0.958 nSn: 0.077

AP1—NFkappaB
nFN: 480 nFP: 199 nTN: 90933 nTP: 0
nASP: 0.0 nCC: −0.003 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

Ebox—Ets
nFN: 134 nFP: 20 nTN: 13006 nTP: 7
nASP: 0.155 nCC: 0.109 nPC: 0.043
nPPV: 0.259 nSP: 0.998 nSn: 0.05

Ets—AML
nFN: 71 nFP: 379 nTN: 13316 nTP: 24
nASP: 0.157 nCC: 0.11 nPC: 0.051
nPPV: 0.06 nSP: 0.972 nSn: 0.253

IRF—NFkappaB
nFN: 189 nFP: 187 nTN: 9595 nTP: 68
nASP: 0.266 nCC: 0.246 nPC: 0.153
nPPV: 0.267 nSP: 0.981 nSn: 0.265

NFkappaB—HMGIY
nFN: 73 nFP: 159 nTN: 15015 nTP: 0
nASP: 0.0 nCC: −0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.99 nSn: 0.0

Sp1—Ets
nFN: 235 nFP: 244 nTN: 13869 nTP: 50
nASP: 0.173 nCC: 0.156 nPC: 0.095
nPPV: 0.17 nSP: 0.983 nSn: 0.175
```

-s FP -l 0.2,0.3,0.4

Listing C.5: 100% Combined

```
distance: 0
nFN: 1137 nFP: 31562 nTN: 143478 nTP: 450
nASP: 0.149 nCC: 0.025 nPC: 0.014
nPPV: 0.014 nSP: 0.82 nSn: 0.284

distance: 75
nFN: 1385 nFP: 219 nTN: 174949 nTP: 74
nASP: 0.152 nCC: 0.11 nPC: 0.044
nPPV: 0.253 nSP: 0.999 nSn: 0.051

distance: 150
nFN: 1261 nFP: 418 nTN: 174635 nTP: 313
nASP: 0.314 nCC: 0.288 nPC: 0.157
nPPV: 0.428 nSP: 0.998 nSn: 0.199

distance: 225
nFN: 1200 nFP: 845 nTN: 174195 nTP: 387
nASP: 0.279 nCC: 0.271 nPC: 0.159
nPPV: 0.314 nSP: 0.995 nSn: 0.244
```

Listing C.6: 100% Distance 75

```
AP1—NFAT
nFN: 153 nFP: 106 nTN: 18115 nTP: 0
nASP: 0.0 nCC: −0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0

AP1—NFkappaB
nFN: 426 nFP: 1 nTN: 91131 nTP: 54
nASP: 0.548 nCC: 0.332 nPC: 0.112
nPPV: 0.982 nSP: 1.0 nSn: 0.113

Ebox—Ets
nFN: 141 nFP: 30 nTN: 12996 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

Ets—AML
nFN: 95 nFP: 0 nTN: 13695 nTP: 0
nASP: ∗ nCC: ∗ nPC: 0.0
nPPV: ∗ nSP: 1.0 nSn: 0.0

IRF—NFkappaB
nFN: 237 nFP: 0 nTN: 9782 nTP: 20
nASP: 0.539 nCC: 0.276 nPC: 0.078
nPPV: 1.0 nSP: 1.0 nSn: 0.078

NFkappaB—HMGIY
nFN: 73 nFP: 0 nTN: 15174 nTP: 0
nASP: ∗ nCC: ∗ nPC: 0.0
nPPV: ∗ nSP: 1.0 nSn: 0.0

Sp1—Ets
nFN: 260 nFP: 82 nTN: 14056 nTP: 0
nASP: 0.0 nCC: −0.01 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0
```

Listing C.7: 100% Distance 150

```
AP1—NFAT
nFN: 153 nFP: 231 nTN: 17990 nTP: 0
nASP: 0.0 nCC: −0.01 nPC: 0.0
nPPV: 0.0 nSP: 0.987 nSn: 0.0

AP1—NFkappaB
nFN: 426 nFP: 1 nTN: 91131 nTP: 54
nASP: 0.548 nCC: 0.332 nPC: 0.112
nPPV: 0.982 nSP: 1.0 nSn: 0.113

Ebox—Ets
nFN: 141 nFP: 30 nTN: 12996 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

Ets—AML
nFN: 86 nFP: 73 nTN: 13622 nTP: 9
nASP: 0.103 nCC: 0.096 nPC: 0.054
nPPV: 0.11 nSP: 0.995 nSn: 0.095

IRF—NFkappaB
nFN: 237 nFP: 0 nTN: 9782 nTP: 20
nASP: 0.539 nCC: 0.276 nPC: 0.078
nPPV: 1.0 nSP: 1.0 nSn: 0.078

NFkappaB—HMGIY
nFN: 73 nFP: 0 nTN: 15174 nTP: 0
nASP: ∗ nCC: ∗ nPC: 0.0
nPPV: ∗ nSP: 1.0 nSn: 0.0

Sp1—Ets
nFN: 145 nFP: 83 nTN: 13940 nTP: 230
nASP: 0.674 nCC: 0.663 nPC: 0.502
nPPV: 0.735 nSP: 0.994 nSn: 0.613
```

Listing C.8: 100% Distance 225

```
AP1—NFAT
nFN: 140 nFP: 312 nTN: 17896 nTP: 26
nASP: 0.117 nCC: 0.098 nPC: 0.054
nPPV: 0.077 nSP: 0.983 nSn: 0.157

AP1—NFkappaB
nFN: 426 nFP: 1 nTN: 91131 nTP: 54
nASP: 0.548 nCC: 0.332 nPC: 0.112
nPPV: 0.982 nSP: 1.0 nSn: 0.113

Ebox—Ets
nFN: 141 nFP: 30 nTN: 12996 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

Ets—AML
nFN: 86 nFP: 73 nTN: 13622 nTP: 9
nASP: 0.103 nCC: 0.096 nPC: 0.054
nPPV: 0.11 nSP: 0.995 nSn: 0.095

IRF—NFkappaB
nFN: 189 nFP: 187 nTN: 9595 nTP: 68
nASP: 0.266 nCC: 0.246 nPC: 0.153
nPPV: 0.267 nSP: 0.981 nSn: 0.265

NFkappaB—HMGIY
nFN: 73 nFP: 159 nTN: 15015 nTP: 0
nASP: 0.0 nCC: −0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.99 nSn: 0.0

Sp1—Ets
nFN: 145 nFP: 83 nTN: 13940 nTP: 230
nASP: 0.674 nCC: 0.663 nPC: 0.502
nPPV: 0.735 nSP: 0.994 nSn: 0.613
```

-s FP -l 0.4,0.6,0.8

Listing C.9: 100% Combined

```
distance: 0
nFN: 1037 nFP: 6617 nTN: 168530 nTP: 443
nASP: 0.181 nCC: 0.122 nPC: 0.055
nPPV: 0.063 nSP: 0.962 nSn: 0.299

distance: 75
nFN: 1242 nFP: 540 nTN: 174605 nTP: 240
nASP: 0.235 nCC: 0.219 nPC: 0.119
nPPV: 0.308 nSP: 0.997 nSn: 0.162

distance: 150
nFN: 1233 nFP: 966 nTN: 174179 nTP: 249
nASP: 0.187 nCC: 0.179 nPC: 0.102
nPPV: 0.205 nSP: 0.994 nSn: 0.168

distance: 225
nFN: 1233 nFP: 1345 nTN: 173800 nTP: 249
nASP: 0.162 nCC: 0.155 nPC: 0.088
nPPV: 0.156 nSP: 0.992 nSn: 0.168
```

Listing C.10: 100% Distance 75

```
AP1—NFAT
nFN: 153 nFP: 106 nTN: 18115 nTP: 0
nASP: 0.0 nCC: −0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0

AP1—NFkappaB
nFN: 426 nFP: 1 nTN: 91131 nTP: 54
nASP: 0.548 nCC: 0.332 nPC: 0.112
nPPV: 0.982 nSP: 1.0 nSn: 0.113

Ebox—Ets
nFN: 87 nFP: 79 nTN: 12949 nTP: 52
nASP: 0.386 nCC: 0.379 nPC: 0.239
nPPV: 0.397 nSP: 0.994 nSn: 0.374

Ets—AML
nFN: 95 nFP: 33 nTN: 13662 nTP: 0
nASP: 0.0 nCC: −0.004 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

IRF—NFkappaB
nFN: 178 nFP: 114 nTN: 9668 nTP: 79
```

```
nASP: 0.358 nCC: 0.34 nPC: 0.213
nPPV: 0.409 nSP: 0.988 nSn: 0.307

NFkappaB–HMGIY
nFN: 68 nFP: 110 nTN: 15064 nTP: 5
nASP: 0.056 nCC: 0.049 nPC: 0.027
nPPV: 0.043 nSP: 0.993 nSn: 0.068

Sp1–Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

Listing C.11: 100% Distance 150

```
AP1–NFAT
nFN: 153 nFP: 323 nTN: 17898 nTP: 0
nASP: 0.0 nCC: −0.012 nPC: 0.0
nPPV: 0.0 nSP: 0.982 nSn: 0.0

AP1–NFkappaB
nFN: 426 nFP: 92 nTN: 91040 nTP: 54
nASP: 0.242 nCC: 0.202 nPC: 0.094
nPPV: 0.37 nSP: 0.999 nSn: 0.113

Ebox–Ets
nFN: 87 nFP: 79 nTN: 12949 nTP: 52
nASP: 0.386 nCC: 0.379 nPC: 0.239
nPPV: 0.397 nSP: 0.994 nSn: 0.374

Ets–AML
nFN: 86 nFP: 151 nTN: 13544 nTP: 9
nASP: 0.076 nCC: 0.065 nPC: 0.037
nPPV: 0.056 nSP: 0.989 nSn: 0.095

IRF–NFkappaB
nFN: 178 nFP: 114 nTN: 9668 nTP: 79
nASP: 0.358 nCC: 0.34 nPC: 0.213
nPPV: 0.409 nSP: 0.988 nSn: 0.307

NFkappaB–HMGIY
nFN: 68 nFP: 110 nTN: 15064 nTP: 5
nASP: 0.056 nCC: 0.049 nPC: 0.027
nPPV: 0.043 nSP: 0.993 nSn: 0.068

Sp1–Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

Listing C.12: 100% Distance 225

```
AP1–NFAT
nFN: 153 nFP: 702 nTN: 17519 nTP: 0
nASP: 0.0 nCC: −0.018 nPC: 0.0
nPPV: 0.0 nSP: 0.961 nSn: 0.0

AP1–NFkappaB
nFN: 426 nFP: 92 nTN: 91040 nTP: 54
nASP: 0.242 nCC: 0.202 nPC: 0.094
nPPV: 0.37 nSP: 0.999 nSn: 0.113

Ebox–Ets
nFN: 87 nFP: 79 nTN: 12949 nTP: 52
nASP: 0.386 nCC: 0.379 nPC: 0.239
nPPV: 0.397 nSP: 0.994 nSn: 0.374

Ets–AML
nFN: 86 nFP: 151 nTN: 13544 nTP: 9
nASP: 0.076 nCC: 0.065 nPC: 0.037
nPPV: 0.056 nSP: 0.989 nSn: 0.095

IRF–NFkappaB
nFN: 178 nFP: 114 nTN: 9668 nTP: 79
nASP: 0.358 nCC: 0.34 nPC: 0.213
nPPV: 0.409 nSP: 0.988 nSn: 0.307

NFkappaB–HMGIY
nFN: 68 nFP: 110 nTN: 15064 nTP: 5
nASP: 0.056 nCC: 0.049 nPC: 0.027
nPPV: 0.043 nSP: 0.993 nSn: 0.068

Sp1–Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

-s TP -l 0.8,0.9,1.0

Listing C.13: 50% Combined

```
distance: 0
nFN: 1114 nFP: 9138 nTN: 166068 nTP: 307
nASP: 0.125 nCC: 0.065 nPC: 0.029
nPPV: 0.033 nSP: 0.948 nSn: 0.216

distance: 75
nFN: 1198 nFP: 723 nTN: 166315 nTP: 164
nASP: 0.153 nCC: 0.144 nPC: 0.079
nPPV: 0.185 nSP: 0.996 nSn: 0.12

distance: 150
nFN: 1189 nFP: 1887 nTN: 165184 nTP: 140
nASP: 0.087 nCC: 0.076 nPC: 0.044
nPPV: 0.069 nSP: 0.989 nSn: 0.105

distance: 225
nFN: 1281 nFP: 2630 nTN: 172576 nTP: 140
nASP: 0.075 nCC: 0.06 nPC: 0.035
nPPV: 0.051 nSP: 0.985 nSn: 0.099
```

Listing C.14: 50% Distance 75

```
AP1–NFAT
nFN: 132 nFP: 294 nTN: 17948 nTP: 0
nASP: 0.0 nCC: −0.011 nPC: 0.0
nPPV: 0.0 nSP: 0.984 nSn: 0.0

AP1–NFkappaB
nFN: 388 nFP: 0 nTN: 82997 nTP: 0
nASP: ∗ nCC: ∗ nPC: 0.0
nPPV: ∗ nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 87 nFP: 62 nTN: 12966 nTP: 52
nASP: 0.415 nCC: 0.407 nPC: 0.259
nPPV: 0.456 nSP: 0.995 nSn: 0.374

Ets–AML
nFN: 80 nFP: 105 nTN: 13590 nTP: 15
nASP: 0.142 nCC: 0.134 nPC: 0.075
nPPV: 0.125 nSP: 0.992 nSn: 0.158

IRF–NFkappaB
nFN: 196 nFP: 41 nTN: 9741 nTP: 61
nASP: 0.418 nCC: 0.367 nPC: 0.205
nPPV: 0.598 nSP: 0.996 nSn: 0.237

NFkappaB–HMGIY
nFN: 73 nFP: 28 nTN: 15146 nTP: 0
nASP: 0.0 nCC: −0.003 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

Sp1–Ets
nFN: 242 nFP: 193 nTN: 13927 nTP: 36
nASP: 0.143 nCC: 0.127 nPC: 0.076
nPPV: 0.157 nSP: 0.986 nSn: 0.129
```

Listing C.15: 50% Distance 150

```
AP1–NFAT
nFN: 108 nFP: 596 nTN: 17661 nTP: 9
nASP: 0.046 nCC: 0.02 nPC: 0.013
nPPV: 0.015 nSP: 0.967 nSn: 0.077

AP1–NFkappaB
nFN: 388 nFP: 0 nTN: 82997 nTP: 0
nASP: ∗ nCC: ∗ nPC: 0.0
nPPV: ∗ nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 87 nFP: 62 nTN: 12966 nTP: 52
nASP: 0.415 nCC: 0.407 nPC: 0.259
nPPV: 0.456 nSP: 0.995 nSn: 0.374

Ets–AML
nFN: 71 nFP: 197 nTN: 13498 nTP: 24
nASP: 0.181 nCC: 0.157 nPC: 0.082
nPPV: 0.109 nSP: 0.986 nSn: 0.253

IRF–NFkappaB
nFN: 218 nFP: 288 nTN: 9494 nTP: 39
nASP: 0.136 nCC: 0.109 nPC: 0.072
nPPV: 0.119 nSP: 0.971 nSn: 0.152

NFkappaB–HMGIY
nFN: 57 nFP: 269 nTN: 14905 nTP: 16
nASP: 0.138 nCC: 0.103 nPC: 0.047
nPPV: 0.056 nSP: 0.982 nSn: 0.219

Sp1–Ets
nFN: 260 nFP: 475 nTN: 13663 nTP: 0
nASP: 0.0 nCC: −0.025 nPC: 0.0
nPPV: 0.0 nSP: 0.966 nSn: 0.0
```

Listing C.16: 50% Distance 225

```
AP1–NFAT
nFN: 108 nFP: 765 nTN: 17492 nTP: 9
nASP: 0.045 nCC: 0.014 nPC: 0.01
nPPV: 0.012 nSP: 0.958 nSn: 0.077

AP1–NFkappaB
nFN: 480 nFP: 199 nTN: 90933 nTP: 0
nASP: 0.0 nCC: -0.003 nPC: 0.0
nPPV: 0.0 nSP: 0.998 nSn: 0.0

Ebox–Ets
nFN: 87 nFP: 62 nTN: 12966 nTP: 52
nASP: 0.415 nCC: 0.407 nPC: 0.259
nPPV: 0.456 nSP: 0.995 nSn: 0.374

Ets–AML
nFN: 71 nFP: 379 nTN: 13316 nTP: 24
nASP: 0.157 nCC: 0.11 nPC: 0.051
nPPV: 0.06 nSP: 0.972 nSn: 0.253

IRF–NFkappaB
nFN: 218 nFP: 288 nTN: 9494 nTP: 39
nASP: 0.136 nCC: 0.109 nPC: 0.072
nPPV: 0.119 nSP: 0.971 nSn: 0.152

NFkappaB–HMGIY
nFN: 57 nFP: 462 nTN: 14712 nTP: 16
nASP: 0.126 nCC: 0.075 nPC: 0.03
nPPV: 0.033 nSP: 0.97 nSn: 0.219

Sp1–Ets
nFN: 260 nFP: 475 nTN: 13663 nTP: 0
nASP: 0.0 nCC: -0.025 nPC: 0.0
nPPV: 0.0 nSP: 0.966 nSn: 0.0
```

-s FP -l 0.2,0.3,0.4

Listing C.17: 50% Combined

```
distance: 0
nFN: 1113 nFP: 12114 nTN: 162924 nTP: 476
nASP: 0.169 nCC: 0.085 nPC: 0.035
nPPV: 0.038 nSP: 0.931 nSn: 0.3

distance: 75
nFN: 1380 nFP: 447 nTN: 174721 nTP: 79
nASP: 0.102 nCC: 0.086 nPC: 0.041
nPPV: 0.15 nSP: 0.997 nSn: 0.054

distance: 150
nFN: 1249 nFP: 817 nTN: 174221 nTP: 340
nASP: 0.254 nCC: 0.245 nPC: 0.141
nPPV: 0.294 nSP: 0.995 nSn: 0.214

distance: 225
nFN: 1224 nFP: 1011 nTN: 174027 nTP: 365
nASP: 0.248 nCC: 0.24 nPC: 0.14
nPPV: 0.265 nSP: 0.994 nSn: 0.23
```

Listing C.18: 50% Distance 75

```
AP1–NFAT
nFN: 153 nFP: 106 nTN: 18115 nTP: 0
nASP: 0.0 nCC: -0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0

AP1–NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: -0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets–AML
nFN: 95 nFP: 49 nTN: 13646 nTP: 0
nASP: 0.0 nCC: -0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.996 nSn: 0.0

IRF–NFkappaB
nFN: 237 nFP: 0 nTN: 9782 nTP: 20
nASP: 0.539 nCC: 0.276 nPC: 0.078
nPPV: 1.0 nSP: 1.0 nSn: 0.078

NFkappaB–HMGIY
nFN: 73 nFP: 99 nTN: 15075 nTP: 0
nASP: 0.0 nCC: -0.006 nPC: 0.0
nPPV: 0.0 nSP: 0.993 nSn: 0.0

Sp1–Ets
nFN: 260 nFP: 82 nTN: 14056 nTP: 0
nASP: 0.0 nCC: -0.01 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0
```

Listing C.19: 50% Distance 150

```
AP1–NFAT
nFN: 168 nFP: 118 nTN: 18088 nTP: 0
nASP: 0.0 nCC: -0.008 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0

AP1–NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: -0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets–AML
nFN: 95 nFP: 49 nTN: 13646 nTP: 0
nASP: 0.0 nCC: -0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.996 nSn: 0.0

IRF–NFkappaB
nFN: 214 nFP: 205 nTN: 9577 nTP: 43
nASP: 0.17 nCC: 0.149 nPC: 0.093
nPPV: 0.173 nSP: 0.979 nSn: 0.167

NFkappaB–HMGIY
nFN: 65 nFP: 251 nTN: 14923 nTP: 8
nASP: 0.071 nCC: 0.05 nPC: 0.025
nPPV: 0.031 nSP: 0.983 nSn: 0.11

Sp1–Ets
nFN: 145 nFP: 83 nTN: 13940 nTP: 230
nASP: 0.674 nCC: 0.663 nPC: 0.502
nPPV: 0.735 nSP: 0.994 nSn: 0.613
```

Listing C.20: 50% Distance 225

```
AP1–NFAT
nFN: 168 nFP: 330 nTN: 17876 nTP: 0
nASP: 0.0 nCC: -0.013 nPC: 0.0
nPPV: 0.0 nSP: 0.982 nSn: 0.0

AP1–NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: -0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets–AML
nFN: 95 nFP: 49 nTN: 13646 nTP: 0
nASP: 0.0 nCC: -0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.996 nSn: 0.0

IRF–NFkappaB
nFN: 189 nFP: 187 nTN: 9595 nTP: 68
nASP: 0.266 nCC: 0.246 nPC: 0.153
nPPV: 0.267 nSP: 0.981 nSn: 0.265

NFkappaB–HMGIY
nFN: 65 nFP: 251 nTN: 14923 nTP: 8
nASP: 0.071 nCC: 0.05 nPC: 0.025
nPPV: 0.031 nSP: 0.983 nSn: 0.11

Sp1–Ets
nFN: 145 nFP: 83 nTN: 13940 nTP: 230
nASP: 0.674 nCC: 0.663 nPC: 0.502
nPPV: 0.735 nSP: 0.994 nSn: 0.613
```

-s FP -l 0.4,0.6,0.8

Listing C.21: 50% Combined

```
distance: 0
nFN: 1017 nFP: 10305 nTN: 164825 nTP: 480
nASP: 0.183 nCC: 0.1 nPC: 0.041
nPPV: 0.045 nSP: 0.941 nSn: 0.321

distance: 75
nFN: 1296 nFP: 576 nTN: 174567 nTP: 188
nASP: 0.187 nCC: 0.172 nPC: 0.091
```

```
nPPV: 0.246 nSP: 0.997 nSn: 0.127

distance: 150
nFN: 1285 nFP: 1157 nTN: 174007 nTP: 178
nASP: 0.128 nCC: 0.12 nPC: 0.068
nPPV: 0.133 nSP: 0.993 nSn: 0.122

distance: 225
nFN: 1305 nFP: 1568 nTN: 173575 nTP: 179
nASP: 0.111 nCC: 0.103 nPC: 0.059
nPPV: 0.102 nSP: 0.991 nSn: 0.121
```

Listing C.22: 50% Distance 75

```
AP1—NFAT
nFN: 153 nFP: 106 nTN: 18115 nTP: 0
nASP: 0.0 nCC: −0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0

AP1—NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: −0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox—Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 95 nFP: 49 nTN: 13646 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.996 nSn: 0.0

IRF—NFkappaB
nFN: 178 nFP: 114 nTN: 9668 nTP: 79
nASP: 0.358 nCC: 0.34 nPC: 0.213
nPPV: 0.409 nSP: 0.988 nSn: 0.307

NFkappaB—HMGIY
nFN: 73 nFP: 99 nTN: 15075 nTP: 0
nASP: 0.0 nCC: −0.006 nPC: 0.0
nPPV: 0.0 nSP: 0.993 nSn: 0.0

Sp1—Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

Listing C.23: 50% Distance 150

```
AP1—NFAT
nFN: 132 nFP: 353 nTN: 17889 nTP: 0
nASP: 0.0 nCC: −0.012 nPC: 0.0
nPPV: 0.0 nSP: 0.981 nSn: 0.0

AP1—NFkappaB
nFN: 470 nFP: 113 nTN: 91019 nTP: 10
nASP: 0.051 nCC: 0.039 nPC: 0.017
nPPV: 0.081 nSP: 0.999 nSn: 0.021

Ebox—Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 95 nFP: 49 nTN: 13646 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.996 nSn: 0.0

IRF—NFkappaB
nFN: 214 nFP: 205 nTN: 9577 nTP: 43
nASP: 0.17 nCC: 0.149 nPC: 0.093
nPPV: 0.173 nSP: 0.979 nSn: 0.167

NFkappaB—HMGIY
nFN: 57 nFP: 269 nTN: 14905 nTP: 16
nASP: 0.138 nCC: 0.103 nPC: 0.047
nPPV: 0.056 nSP: 0.982 nSn: 0.219

Sp1—Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

Listing C.24: 50% Distance 225

```
AP1—NFAT
nFN: 153 nFP: 702 nTN: 17519 nTP: 0
nASP: 0.0 nCC: −0.018 nPC: 0.0
nPPV: 0.0 nSP: 0.961 nSn: 0.0

AP1—NFkappaB
nFN: 470 nFP: 113 nTN: 91019 nTP: 10
nASP: 0.051 nCC: 0.039 nPC: 0.017
nPPV: 0.081 nSP: 0.999 nSn: 0.021

Ebox—Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 95 nFP: 49 nTN: 13646 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.996 nSn: 0.0

IRF—NFkappaB
nFN: 213 nFP: 267 nTN: 9515 nTP: 44
nASP: 0.156 nCC: 0.131 nPC: 0.084
nPPV: 0.141 nSP: 0.973 nSn: 0.171

NFkappaB—HMGIY
nFN: 57 nFP: 269 nTN: 14905 nTP: 16
nASP: 0.138 nCC: 0.103 nPC: 0.047
nPPV: 0.056 nSP: 0.982 nSn: 0.219

Sp1—Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

-s TP -l 0.8,0.9,1.0

Listing C.25: 25% Combined

```
distance: 0
nFN: 850 nFP: 15770 nTN: 159382 nTP: 625
nASP: 0.231 nCC: 0.105 nPC: 0.036
nPPV: 0.038 nSP: 0.91 nSn: 0.424

distance: 75
nFN: 1183 nFP: 824 nTN: 166214 nTP: 179
nASP: 0.155 nCC: 0.147 nPC: 0.082
nPPV: 0.178 nSP: 0.995 nSn: 0.131

distance: 150
nFN: 1258 nFP: 2241 nTN: 172963 nTP: 165
nASP: 0.093 nCC: 0.08 nPC: 0.045
nPPV: 0.069 nSP: 0.987 nSn: 0.116

distance: 225
nFN: 1217 nFP: 3160 nTN: 172104 nTP: 146
nASP: 0.076 nCC: 0.058 nPC: 0.032
nPPV: 0.044 nSP: 0.982 nSn: 0.107
```

Listing C.26: 25% Distance 75

```
AP1—NFAT
nFN: 132 nFP: 294 nTN: 17948 nTP: 0
nASP: 0.0 nCC: −0.011 nPC: 0.0
nPPV: 0.0 nSP: 0.984 nSn: 0.0

AP1—NFkappaB
nFN: 388 nFP: 0 nTN: 82997 nTP: 0
nASP: ∗ nCC: ∗ nPC: 0.0
nPPV: ∗ nSP: 1.0 nSn: 0.0

Ebox—Ets
nFN: 87 nFP: 62 nTN: 12966 nTP: 52
nASP: 0.415 nCC: 0.407 nPC: 0.259
nPPV: 0.456 nSP: 0.995 nSn: 0.374

Ets—AML
nFN: 65 nFP: 54 nTN: 13641 nTP: 30
nASP: 0.337 nCC: 0.332 nPC: 0.201
nPPV: 0.357 nSP: 0.996 nSn: 0.316

IRF—NFkappaB
nFN: 196 nFP: 41 nTN: 9741 nTP: 61
nASP: 0.418 nCC: 0.367 nPC: 0.205
nPPV: 0.598 nSP: 0.996 nSn: 0.237

NFkappaB—HMGIY
nFN: 73 nFP: 180 nTN: 14994 nTP: 0
nASP: 0.0 nCC: −0.008 nPC: 0.0
nPPV: 0.0 nSP: 0.988 nSn: 0.0

Sp1—Ets
nFN: 242 nFP: 193 nTN: 13927 nTP: 36
nASP: 0.143 nCC: 0.127 nPC: 0.076
nPPV: 0.157 nSP: 0.986 nSn: 0.129
```

Listing C.27: 25% Distance 150

```
AP1–NFAT
nFN: 108 nFP: 596 nTN: 17661 nTP: 9
nASP: 0.046 nCC: 0.02 nPC: 0.013
nPPV: 0.015 nSP: 0.967 nSn: 0.077

AP1–NFkappaB
nFN: 480 nFP: 117 nTN: 91015 nTP: 0
nASP: 0.0 nCC: −0.003 nPC: 0.0
nPPV: 0.0 nSP: 0.999 nSn: 0.0

Ebox–Ets
nFN: 93 nFP: 183 nTN: 12843 nTP: 48
nASP: 0.274 nCC: 0.256 nPC: 0.148
nPPV: 0.208 nSP: 0.986 nSn: 0.34

Ets–AML
nFN: 65 nFP: 154 nTN: 13541 nTP: 30
nASP: 0.24 nCC: 0.22 nPC: 0.12
nPPV: 0.163 nSP: 0.989 nSn: 0.316

IRF–NFkappaB
nFN: 218 nFP: 288 nTN: 9494 nTP: 39
nASP: 0.136 nCC: 0.109 nPC: 0.072
nPPV: 0.119 nSP: 0.971 nSn: 0.152

NFkappaB–HMGIY
nFN: 34 nFP: 428 nTN: 14746 nTP: 39
nASP: 0.309 nCC: 0.203 nPC: 0.078
nPPV: 0.084 nSP: 0.972 nSn: 0.534

Sp1–Ets
nFN: 260 nFP: 475 nTN: 13663 nTP: 0
nASP: 0.0 nCC: −0.025 nPC: 0.0
nPPV: 0.0 nSP: 0.966 nSn: 0.0
```

Listing C.28: 25% Distance 225

```
AP1–NFAT
nFN: 108 nFP: 765 nTN: 17492 nTP: 9
nASP: 0.045 nCC: 0.014 nPC: 0.01
nPPV: 0.012 nSP: 0.958 nSn: 0.077

AP1–NFkappaB
nFN: 398 nFP: 479 nTN: 90713 nTP: 22
nASP: 0.048 nCC: 0.043 nPC: 0.024
nPPV: 0.044 nSP: 0.995 nSn: 0.052

Ebox–Ets
nFN: 93 nFP: 183 nTN: 12843 nTP: 48
nASP: 0.274 nCC: 0.256 nPC: 0.148
nPPV: 0.208 nSP: 0.986 nSn: 0.34

Ets–AML
nFN: 67 nFP: 621 nTN: 13074 nTP: 28
nASP: 0.169 nCC: 0.097 nPC: 0.039
nPPV: 0.043 nSP: 0.955 nSn: 0.295

IRF–NFkappaB
nFN: 218 nFP: 288 nTN: 9494 nTP: 39
nASP: 0.136 nCC: 0.109 nPC: 0.072
nPPV: 0.119 nSP: 0.971 nSn: 0.152

NFkappaB–HMGIY
nFN: 73 nFP: 349 nTN: 14825 nTP: 0
nASP: 0.0 nCC: −0.011 nPC: 0.0
nPPV: 0.0 nSP: 0.977 nSn: 0.0

Sp1–Ets
nFN: 260 nFP: 475 nTN: 13663 nTP: 0
nASP: 0.0 nCC: −0.025 nPC: 0.0
nPPV: 0.0 nSP: 0.966 nSn: 0.0
```

-s FP -l 0.2,0.3,0.4

Listing C.29: 25% Combined

```
distance: 0
nFN: 1046 nFP: 15818 nTN: 159242 nTP: 521
nASP: 0.182 nCC: 0.078 nPC: 0.03
nPPV: 0.032 nSP: 0.91 nSn: 0.332

distance: 75
nFN: 1532 nFP: 429 nTN: 174607 nTP: 59
nASP: 0.079 nCC: 0.062 nPC: 0.029
nPPV: 0.121 nSP: 0.998 nSn: 0.037

distance: 150
nFN: 1234 nFP: 1034 nTN: 174005 nTP: 354
nASP: 0.239 nCC: 0.232 nPC: 0.135
nPPV: 0.255 nSP: 0.994 nSn: 0.223

distance: 225
nFN: 1277 nFP: 1609 nTN: 173430 nTP: 311
nASP: 0.179 nCC: 0.17 nPC: 0.097
nPPV: 0.162 nSP: 0.991 nSn: 0.196
```

Listing C.30: 25% Distance 75

```
AP1–NFAT
nFN: 168 nFP: 48 nTN: 18158 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.997 nSn: 0.0

AP1–NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: −0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets–AML
nFN: 95 nFP: 47 nTN: 13648 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.997 nSn: 0.0

IRF–NFkappaB
nFN: 257 nFP: 52 nTN: 9730 nTP: 0
nASP: 0.0 nCC: −0.012 nPC: 0.0
nPPV: 0.0 nSP: 0.995 nSn: 0.0

NFkappaB–HMGIY
nFN: 73 nFP: 99 nTN: 15075 nTP: 0
nASP: 0.0 nCC: −0.006 nPC: 0.0
nPPV: 0.0 nSP: 0.993 nSn: 0.0

Sp1–Ets
nFN: 377 nFP: 72 nTN: 13949 nTP: 0
nASP: 0.0 nCC: −0.012 nPC: 0.0
nPPV: 0.0 nSP: 0.995 nSn: 0.0
```

Listing C.31: 25% Distance 150

```
AP1–NFAT
nFN: 168 nFP: 242 nTN: 17964 nTP: 0
nASP: 0.0 nCC: −0.011 nPC: 0.0
nPPV: 0.0 nSP: 0.987 nSn: 0.0

AP1–NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: −0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox–Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets–AML
nFN: 95 nFP: 47 nTN: 13648 nTP: 0
nASP: 0.0 nCC: −0.005 nPC: 0.0
nPPV: 0.0 nSP: 0.997 nSn: 0.0

IRF–NFkappaB
nFN: 214 nFP: 205 nTN: 9577 nTP: 43
nASP: 0.17 nCC: 0.149 nPC: 0.093
nPPV: 0.173 nSP: 0.979 nSn: 0.167

NFkappaB–HMGIY
nFN: 65 nFP: 143 nTN: 15031 nTP: 8
nASP: 0.082 nCC: 0.07 nPC: 0.037
nPPV: 0.053 nSP: 0.991 nSn: 0.11

Sp1–Ets
nFN: 130 nFP: 286 nTN: 13738 nTP: 244
nASP: 0.556 nCC: 0.534 nPC: 0.37
nPPV: 0.46 nSP: 0.98 nSn: 0.652
```

Listing C.32: 25% Distance 225

```
AP1–NFAT
nFN: 168 nFP: 330 nTN: 17876 nTP: 0
nASP: 0.0 nCC: −0.013 nPC: 0.0
nPPV: 0.0 nSP: 0.982 nSn: 0.0

AP1–NFkappaB
nFN: 480 nFP: 40 nTN: 91092 nTP: 0
nASP: 0.0 nCC: −0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox–Ets
```

```
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 95 nFP: 163 nTN: 13532 nTP: 0
nASP: 0.0 nCC: −0.009 nPC: 0.0
nPPV: 0.0 nSP: 0.988 nSn: 0.0

IRF—NFkappaB
nFN: 257 nFP: 422 nTN: 9360 nTP: 0
nASP: 0.0 nCC: −0.034 nPC: 0.0
nPPV: 0.0 nSP: 0.957 nSn: 0.0

NFkappaB—HMGIY
nFN: 65 nFP: 297 nTN: 14877 nTP: 8
nASP: 0.068 nCC: 0.044 nPC: 0.022
nPPV: 0.026 nSP: 0.98 nSn: 0.11

Sp1—Ets
nFN: 130 nFP: 286 nTN: 13738 nTP: 244
nASP: 0.556 nCC: 0.534 nPC: 0.37
nPPV: 0.46 nSP: 0.98 nSn: 0.652
```

-s FP -l 0.4,0.6,0.8

Listing C.33: 25% Combined

```
distance: 0
nFN: 965 nFP: 17889 nTN: 157224 nTP: 549
nASP: 0.197 nCC: 0.079 nPC: 0.028
nPPV: 0.03 nSP: 0.898 nSn: 0.363

distance: 75
nFN: 1279 nFP: 614 nTN: 174529 nTP: 205
nASP: 0.194 nCC: 0.181 nPC: 0.098
nPPV: 0.25 nSP: 0.996 nSn: 0.138

distance: 150
nFN: 1253 nFP: 1141 nTN: 174023 nTP: 210
nASP: 0.15 nCC: 0.143 nPC: 0.081
nPPV: 0.155 nSP: 0.993 nSn: 0.144

distance: 225
nFN: 1235 nFP: 2070 nTN: 173133 nTP: 189
nASP: 0.109 nCC: 0.096 nPC: 0.054
nPPV: 0.084 nSP: 0.988 nSn: 0.133
```

Listing C.34: 25% Distance 75

```
AP1—NFAT
nFN: 153 nFP: 106 nTN: 18115 nTP: 0
nASP: 0.0 nCC: −0.007 nPC: 0.0
nPPV: 0.0 nSP: 0.994 nSn: 0.0

AP1—NFkappaB
nFN: 480 nFP: 41 nTN: 91091 nTP: 0
nASP: 0.0 nCC: −0.002 nPC: 0.0
nPPV: 0.0 nSP: 1.0 nSn: 0.0

Ebox—Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 78 nFP: 86 nTN: 13609 nTP: 17
nASP: 0.172 nCC: 0.166 nPC: 0.094
nPPV: 0.165 nSP: 0.994 nSn: 0.179

IRF—NFkappaB
nFN: 178 nFP: 114 nTN: 9668 nTP: 79
nASP: 0.358 nCC: 0.34 nPC: 0.213
nPPV: 0.409 nSP: 0.988 nSn: 0.307

NFkappaB—HMGIY
nFN: 73 nFP: 99 nTN: 15075 nTP: 0
nASP: 0.0 nCC: −0.006 nPC: 0.0
nPPV: 0.0 nSP: 0.993 nSn: 0.0

Sp1—Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

Listing C.35: 25% Distance 150

```
AP1—NFAT
nFN: 132 nFP: 353 nTN: 17889 nTP: 0
nASP: 0.0 nCC: −0.012 nPC: 0.0
nPPV: 0.0 nSP: 0.981 nSn: 0.0

AP1—NFkappaB
nFN: 447 nFP: 119 nTN: 91013 nTP: 33
nASP: 0.143 nCC: 0.12 nPC: 0.055
nPPV: 0.217 nSP: 0.999 nSn: 0.069

Ebox—Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 78 nFP: 86 nTN: 13609 nTP: 17
nASP: 0.172 nCC: 0.166 nPC: 0.094
nPPV: 0.165 nSP: 0.994 nSn: 0.179

IRF—NFkappaB
nFN: 214 nFP: 205 nTN: 9577 nTP: 43
nASP: 0.17 nCC: 0.149 nPC: 0.093
nPPV: 0.173 nSP: 0.979 nSn: 0.167

NFkappaB—HMGIY
nFN: 65 nFP: 210 nTN: 14964 nTP: 8
nASP: 0.074 nCC: 0.056 nPC: 0.028
nPPV: 0.037 nSP: 0.986 nSn: 0.11

Sp1—Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```

Listing C.36: 25% Distance 225

```
AP1—NFAT
nFN: 153 nFP: 702 nTN: 17519 nTP: 0
nASP: 0.0 nCC: −0.018 nPC: 0.0
nPPV: 0.0 nSP: 0.961 nSn: 0.0

AP1—NFkappaB
nFN: 398 nFP: 327 nTN: 90865 nTP: 22
nASP: 0.057 nCC: 0.054 nPC: 0.029
nPPV: 0.063 nSP: 0.996 nSn: 0.052

Ebox—Ets
nFN: 82 nFP: 71 nTN: 12955 nTP: 59
nASP: 0.436 nCC: 0.43 nPC: 0.278
nPPV: 0.454 nSP: 0.995 nSn: 0.418

Ets—AML
nFN: 89 nFP: 242 nTN: 13453 nTP: 6
nASP: 0.044 nCC: 0.028 nPC: 0.018
nPPV: 0.024 nSP: 0.982 nSn: 0.063

IRF—NFkappaB
nFN: 213 nFP: 267 nTN: 9515 nTP: 44
nASP: 0.156 nCC: 0.131 nPC: 0.084
nPPV: 0.141 nSP: 0.973 nSn: 0.171

NFkappaB—HMGIY
nFN: 65 nFP: 364 nTN: 14810 nTP: 8
nASP: 0.066 nCC: 0.038 nPC: 0.018
nPPV: 0.022 nSP: 0.976 nSn: 0.11

Sp1—Ets
nFN: 235 nFP: 97 nTN: 14016 nTP: 50
nASP: 0.258 nCC: 0.234 nPC: 0.131
nPPV: 0.34 nSP: 0.993 nSn: 0.175
```