MotifGP: DNA Motif Discovery Using Multiobjective Evolution

by

Manuel Belmadani

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master degree in
Computer Science Specialization in Bioinformatics

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The motif discovery problem is becoming increasingly important for molecular biologists as new sequencing technologies are producing large amounts of data, at rates which are unprecedented. The solution space for DNA motifs is too large to search with naive methods, meaning there is a need for fast and accurate motif detection tools. We propose MotifGP, a multiobjective motif discovery tool evolving regular expressions that characterize overrepresented motifs in a given input dataset. This thesis describes and evaluates a multiobjective strongly typed genetic programming algorithm for the discovery of network expressions in DNA sequences. Using 13 realistic data sets, we compare the results of our tool, MotifGP, to that of DREME, a state-of-art program. MotifGP outperforms DREME when the motifs to be sought are long, and the specificity is distributed over the length of the motif. For shorter motifs, the performance of MotifGP compares favourably with the state-of-the-art method. Finally, we discuss the advantages of multi-objective optimization in the context of this specific motif discovery problem.

Acknowledgments

I would first like to thank my thesis supervisor Dr. Marcel Turcotte for his guidance and insights throughout the course of this research. Your commitment to high standards of quality was inspirational, motivating, and affected my own work with improvements that would not have been possible without your counselling.

I would also like to thank the faculty members of the Ottawa-Carleton Institute of Computer Science for providing me the opportunities to better my academic abilities as a graduate student.

Table of Contents

List of Tables

List of Figures

Glossary

ChIP-seq Chromatin immunoprecipitation sequencing. 5, 15, 17, 27, 28, 39, 56

DNA Deoxyribonucleic acid. 36

DREME Discriminative Regular Expression Motif Elicitation. 14, 15, 18

EA Evolutionary Algorithm. 7, 8, 20, 22, 31, 45

EM Expectation-Maximization. 11, 13, 18

GA Genetic Algorithm. 7, 8, 18

GP Genetic Programming. 6, 7, 8, 35, 36, 37

IUPAC International Union of Pure and Applied Chemistry. 5, 6, 36

MDP Motif Discovery Problem. 7, 8

MOEA Multiobjective Evolutionary Algorithms. 20, 32

MOGP Multiobjective Genetic Programming. 28

NSGA-II Non-dominated Sorting Genetic Algorithm II. 21, 22, 23, 30

PBM Protein binding microarray. 42

PRNG Pseudorandom number generator. 45

PSPM Position-Specific Probability Matrix. 13, 14, 89, 90

PSSM Position-Specific Scoring Matrix. 11, 12, 89

PWM Position Weight Matrix. 18, 36, 42, 89, 90

RE Regular Expressions. 6, 15, 36, 37, 40

SPEA Strength Pareto Evolutionary Algorithm. 24

SPEA2 Strength Pareto Evolutionary Algorithm 2. 24, 25, 30

STGP Strongly Typed Genetic Programming. 6, 8, 31, 35

TF Transcription Factor. 4, 5, 8, 9, 18, 27, 40, 42, 71

TFBS Transcription Factor Binding Site. 4, 6, 8, 14, 17, 27, 28, 35, 36, 40, 41, 42, 45, 49, 54, 70, 80, 81, 84, 87

Chapter 1

Introduction

Bioinformatics and Computational Biology are disciplines which leverage computer science and mathematical concepts to enable research and advancements in life sciences. Genomics, metabolomics and proteomics are specific domains of biology that have seen great advancements and increased impact due to tools and methods that have been developed with a biological question in mind. Thanks to high-throughput next-generation sequencing technologies, experts in genomics are now presented a torrential quantity of data, at rates which are unprecedented.

While molecular biologists are well equipped in resources and knowledge to interpret the genetic information encoded in DNA sequences, parsing, analyzing and retrieving relevant information from these large volumes of data pose additional challenges that must be addressed prior to any interpretation. On one part, the sheer quantity of information cannot possibly be studied by hand, and thus require robust computational methods adapted for specific data processing tasks (e.g. developing tools, pipelines, databases) for biological data. Then, we need reproducible methods, as this type of research is typically applied towards discovering unknown features about organic samples. Any time new mechanisms for gene complexes are discovered, we hope to better our collective understanding of genomics, and data will have to be revisited, re-evaluated, and reclassified. In order to research relevant regulation of species-specific gene functions, biologists also require reproducible methods that identify consistent biological markers between different sequencing reads and experiments.

As pointed out by Buffalo in *Bioinformatics Data Skills: Reproducible and Robust Research With Open Source Tools* [15], the requirement for such methods has become a high priority in recent years since the rate which we gain access to new genetic information drastically increased. Approximately 15 years ago, the cost to sequence a million bases of DNA could cost over $USD 5000, and an entire genome would be nearly $USD 95,000,000 [35]. Today, a million bases of DNA can be sequenced for a mere $USD 0.05, and an entire genome under $USD 5000. Figure 1.1 shows the cost for genome sequencing between 2001 and 2010 [35].

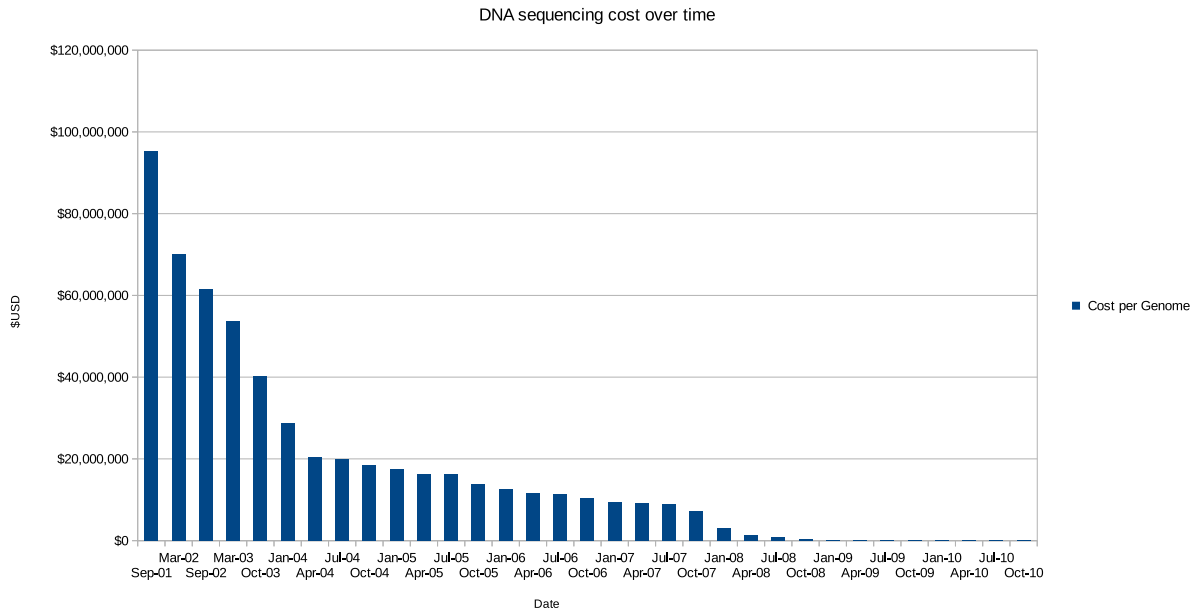


Figure 1.1: The decline of the cost of sequencing a genome throughout recent years. Data available at www.genome.gov/sequencingcosts.

While the cost of sequencing DNA (or RNA, mRNA, or other *omics* data) has dropped dramatically, the number of publicly accessible sequencing reads has sharply risen. The National Center for Biotechnology Information (NCBI) hosts a database for these reads, known as the Sequence Read Archive (SRA) [36]. Especially in the last few years, the rate at which new sequencing reads are submitted have rapidly grown. Indeed, as of August 18th 2015, the open access data from the SRA offers 1,488,379,819.3 Million bytes of data, which translates to 2,189,555,058.7 Million bases of DNA. Figure 1.2 shows rate at which new open access bytes are added to the Sequence Read Archive [36].

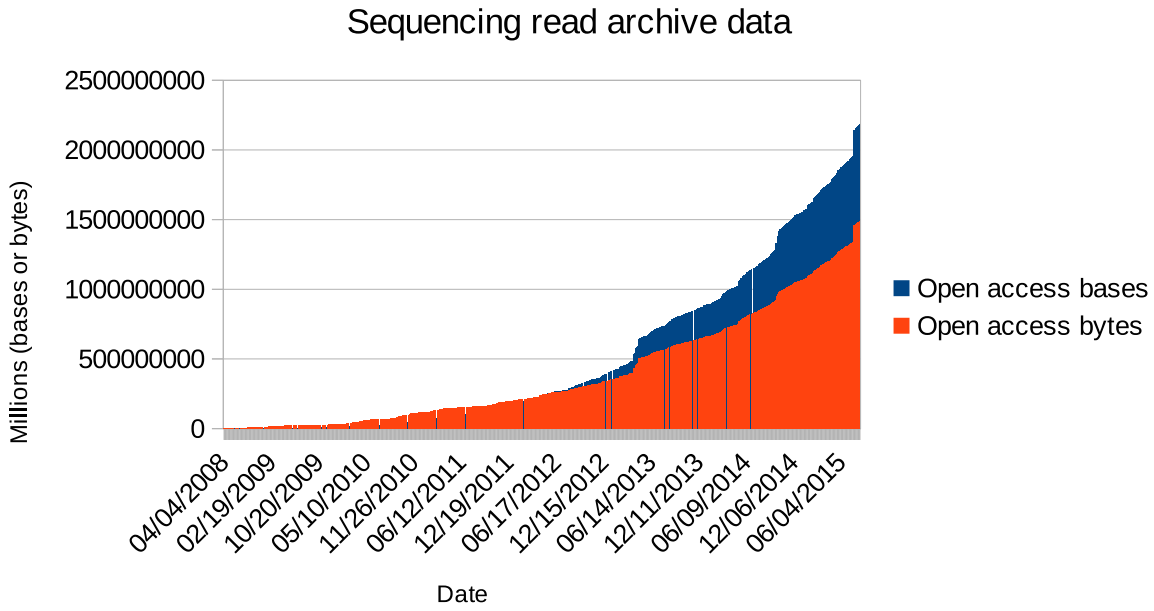


Figure 1.2: The rise in quantity of archived sequencing reads over time. Represented by bases (DNA) or as bytes (data). Data available at www.ncbi.nlm.nih.gov/Traces/sra.

Now that such large quantities of data are produced at increasingly affordable rates, the sequencing analysis also needs to ramp up in productivity. A frequent question molecular biologists try to understand is the relation (or differences) between a set of sequences resulting from a sequencing experiment. Sequencing protocols (such as Chromatin Immunoprecipitation sequencing, or ChIP-seq) are used to identify regions in a DNA sequence where a genetic interaction is identified. During a ChIP-seq experiment, DNA fragments are identified by proteins exposed to the chromatin, hence the chromatin immunoprecipitation. Flagged fragments of DNA are analyzed and mapped back the genome to obtain to location of the interaction [11].

Further analysis of this data is of interest to characterize and understand what fragment of DNA is responsible for the binding. A protein binding site, for example, is a short fragment of DNA where such an interaction occurs, and is often only 6–30 bases of DNA in length. Moreover, binding sites are polymorphic, meaning the same protein can conceivably have multiple valid binding sites with different representations, lengths and affinity. All of which contributes to the subtle regulatory mechanisms.

Motivation

Herein, we propose a novel method to process and identify protein (i.e. transcription factors) binding sites in large ChIP-seq datasets. The task of formulating multiple elements of a common set into a general pattern representation fits the description of *motif discovery*. Many existing motif discovery tools, such as DREME [7] from the MEME suite [8], use

deterministic algorithms that search for putative binding sites over the entire dataset by iteratively refining predictions until it cannot improve anymore. We surveyed various methods of motif discovery and found that while DREME is fairly time efficient, its approach assumes the binding sites are short core nucleotide motifs. This compromise may reduce the confidence behind the alignment of motifs against longer binding site representations.

Our proposed DNA motif discovery software uses Strongly Typed Genetic Programming, a form of evolutionary computing, optimized with a multiobjective fitness evaluation, an optimization method for multiple objective functions. This method obtains predictions by using metrics representative of a putative binding site as a measure of quality (fitness) to guide the search of solutions. Solutions are randomly generated at first, and then evolved amongst each other by favouring reproduction and mutation of candidates whom score best under the fitness metrics. This stochastic search process is beneficial to large ChIP-seq datasets as they allow us to rapidly obtain solutions without having to exhaustively search all the possible combinations of motifs.

The goal of this implementation is to provide a runtime efficient solution to DNA motif discovery that does not explicitly compromise prediction structure by making assumptions on its nucleotide (base) distribution and length. Such predictions will allow the ever-increasing amount of sequencing reads to be processed faster, more accurately and possibly uncover new patterns that other tools failed to capture. At this stage, we are primarily interested in transcription factor binding sites within ChIP-seq peaks, but if successful, these methods can conceivably be extended to deal with new challenges or types of *omics* data.

1.1 Biological question

Our development process requires some understanding of how biological sequencing is interpreted. Sequences are represented as text strings using the 4 basic nucleotides for unique bases (Table 1.1). While our approach is very similar to pattern mining in generic text, the biological nature of the sequences impose additional considerations over what should be mined. This section describes what are the predictions expected to be found by a motif discovery algorithm.

1.1.1 Transcription factor binding sites

A binding site in DNA typically indicates where certain proteins bind. They are found at specific locations in the genome. Molecules such as proteins will bind to these sites in order to regulate the transcription. These proteins are called Transcription Factor (TF) and their binding sites are respectively named Transcription Factor Binding Site (TFBS). Figure 1.3 shows how the binding sites are involved in the transcription process [77].

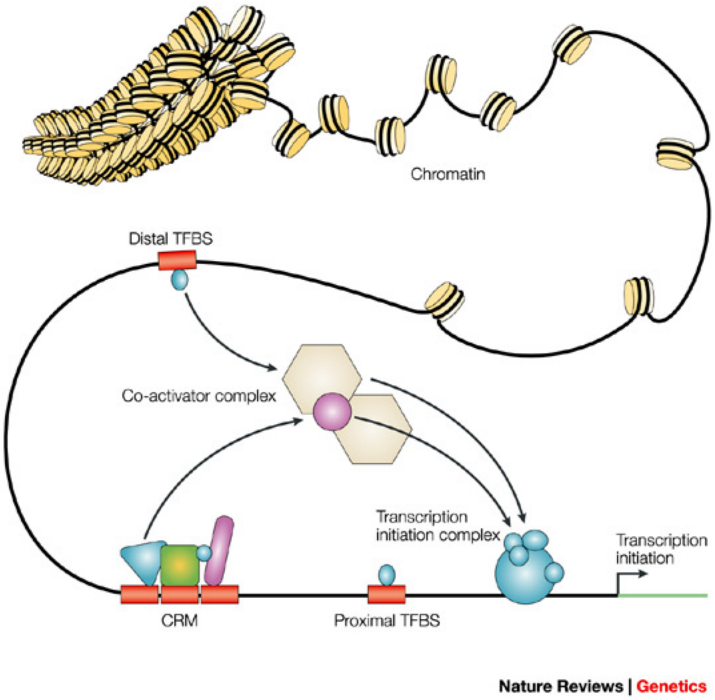


Figure 1.3: Transcription factor binding sites relative to the transcription start site. The black line represents the DNA. Different types of binding sites are represented by the orange boxes. Binding occurs upstream of transcription initiation. Multiple transcription factors can compose cis-regulatory modules [77].

These patterns are important to molecular biologists since their presence may indicate a protein interaction, or lack thereof in their absence. The need for good algorithms stems from the size of the datasets; DNA sequencing data is increasingly voluminous due to advancements in high-throughput sequencing methods such as ChIP-seq. This large input increases the solution space and therefore computing requirements.

On a given output of a ChIP-seq experiment, we are expecting multiple sequences to contain non-identical, instances of a binding site for a common transcription factor. Moreover, the same sequences may contain binding sites for different transcription factors. In that regard, it is expected that motif discovery algorithms identify multiple sites for different transcription factors, though there is usually a *primary factor* that is strongly enriched in the dataset, and additional *co-factors* that are also relevant to transcription regulation (see Figure 1.3). The primary TF was pulled through in the chromatin immunoprecipitation experiment, but binding sites for other factors may be located nearby. Patterns can be represented by means of a syntax, such as regular expressions, or in a more restrictive manner, with network expressions.

1.1.2 DNA Network Expressions

A widely used notation for strings of nucleic acids such as DNA bases is the International Union of Pure and Applied Chemistry (IUPAC) nucleotide alphabet. DNA is comprised of 4 bases: Adenine, Cytosine, Guanine, Thymine (or Uracil, after transcription to RNA). Theses 4 bases are represented by their first letter, i.e. A,C,G and T/U. It is possible to express a pattern that contains more than one character, similar to a regular expression using a character class (or character set). The full definition is listed in Table 1.1.

Table 1.1: The IUPAC nucleotide alphabet. Each letter represents one or more base in a DNA pattern.

IUPAC nucleotide code	**Base**
A	Adenine
C	Cytosine
G	Guanine
T/U	Thymine/Uracil
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
. or -	gap

The candidate solutions in our Genetic Programming (GP) need to be expressive enough to represent biologically significant patterns. We chose to express solutions formed as Regular Expressions (RE). REs are well suited to model TFBS since their syntax can express non-exact characters (such as wild cards or options). Moreover, the RE can be represented by a combination of characters, and modifiers using the same characters. These characters and modifiers can be modelled into terminals and rules, which is what GP uses to generate and evolve candidate solutions.

It is worth mentioning that we can conceivably create solutions from the entire regular expression syntax, but at this stage, we only use a subset of the grammar as our Strongly Typed Genetic Programming (STGP) template. We opted for a more efficient and compact syntax called network expressions, which is a subset of the RE grammar without any Kleene closure operators [50]. Kleene operators allow strings or sets of strings in an expression to be repeated. For example, given the string S, and the Kleene star operator denoted by $*$, the expression $S*$ defines a repetition of 0 or more consecutive instances of S. This type

of operator is not part of the network expression grammar. We also restrict the alphabet of words to only A,C,G or T, and they can be combined in character classes to form the multiple base nucleotides from the IUPAC alphabet. This grammar is designed to be fully backwards compatible with regular expressions, and completely equivalent to the IUPAC nucleotide alphabet notation in terms of expressiveness.

1.2 Evolutionary Motif Discovery

Our approach to the Motif Discovery Problem (MDP) is to search the network expression solution space to infer patterns characterizing common transcription factors within multiple sequences. We search for solutions by means of an evolutionary approach as the solution space is broad and we wish to explore is efficiently. While Strongly Typed Genetic Programming is our preferred approach to the motif discovery problem, we are principally interested in achieving solutions via an evolutionary process optimized over multiple aspect of the predictions. This process is called mutliobjective optimization, and when combined with GP, this becomes multiobjective evolutionary programming.

1.2.1 Inception of MotifGP

We initially got interested in the prospect of employing an evolutionary paradigm for the motif discovery problem after studying PerlGP [44]. PerlGP is a framework to develop general purpose search algorithms using Genetic Programming. Indeed, it has been used in biological motif discovery for protein nuclear localization [34]. We were inspired by the latter as it employs an evolutionary approach to search the regular expression space by the means of a template grammar derived from the nucleic acids bases expressed under regular expressions. We chose to borrow this approach and extend it by restricting the grammar furthermore, and by employing multiobjective optimization.

The DEAP (Distributed Evolutionary Algorithms in Python) framework [27] is a collection of Python modules built for evolutionary programming. We deemed this a good environment under which to build motif discovery software, as DEAP provides a toolbox environment that easily integrates with more specific and customized software. We chose to build MotifGP on top of DEAP as it provides flexibility in our development and already has support for genetic programming. Using a Python solution is also favourable for future development, as there are many bioinformatics libraries and resources available for the language, a major one being Biopython [19].

1.2.2 Evolutionary algorithms

The goal is to develop a tool for the MDP in DNA sequences using an Evolutionary Algorithm (EA) as its search heuristic. EAs are stochastic search processes able to provide good approximations on optimal solutions to complex problems when we do not know of a deterministic polynomial time algorithm. Generally speaking, EAs proceed by evolving

candidate solutions according to a pre-defined measure of fitness. A Genetic Algorithm (GA) is a basic type of EA composed of a population of solutions that are compared in a tournament and combined to create successively improved generations. The solutions are often bit strings, and they are reproduced by simply altering or combining partial solutions over generations.

With GP, the idea is extended to represent solutions as structured programs. A template grammar is used to produce candidate solutions that compile to a given program. Candidates are typically represented as trees where leaf nodes are *terminals* that serve as program parameters and other nodes are functions to interpret them, called *primitives*. This is in contrast to a GA with binary string solutions, where randomly chosen crossovers may not preserve strong attributes from parents.

A specific version of GP is STGP. This adds a data type to the terminals as well as typing constraints for the primitive's inputs and outputs. The GP's template grammar is then strongly typed, allowing more control over candidate solution creation. The template grammar also guarantees type-checked programs through the generations of solutions. This adds more control over what kind of crossovers can happen between solutions since its components are now assigned types which are checked for compatibility during reproduction.

1.2.3 Fitness and multiobjective optimization

An EA must select solutions for reproduction, usually on the basis of its fitness. A fitness is a score, defined by the implementation of the algorithm, which tells how good a solution is. Generally, this applies to problems where we do not know of a way to directly infer solutions with high fitness, but if we are given a solution, we can test its quality; fitness.

This approach works well when a single function is sufficient to compare different solutions. For example, the Traveling Salesman Problem asks what is the shortest open walk in a complete graph. Between two solutions (which are paths), we can immediately tell which is best by picking the shortest distance.

For the MDP, we believe there is not a single metric of fitness that encapsulates the property of a precise and relatively enriched TFBS. For that reason, our approach will consist of different fitness functions that each has a desirable trait of a putative TFBS. The different implementations of our STGP algorithm preserve solutions that are non-dominated amongst each other in what is called a Pareto optimal front. Optimizing solutions on the basis of not one, but multiple fitness objectives are called multiobjective optimization. Over the course of our work, we will demonstrate different attempts at finding an ideal multiobjective fitness function for the MDP.

1.3 Contribution

The following is a list of the main contributions:

- A motif discovery engine using GP for accurate TF prediction.
- A method capable of identifying TF that aligns to known motif databases with a stronger statistical significance than alternative tools can provide.
- Predictions for motifs other methods cannot find.
- Usage of multiobjective optimization algorithms to provide Pareto-optimal set of candidate solutions to the motif discovery problem.
- An open-source implementation capable of handling out-of-the-box motif prediction on variable input as well as search refinement and customization options for specific datasets.

Chapter 2

Literature review

Motif discovery is the broad discipline of finding overrepresented patterns in a given data set [47]. This chapter presents the relevant literature on motif discovery, but also evolutionary algorithms, and multiobjective optimization. We limit this review of existing motif discovery tools to those that are applicable to large ChIP-seq experiments.

2.1 Sequence motif discovery

The discovery of sequence motifs is a well-studied problem. Accordingly, dozens of reviews have been published, including the following ones [14, 22, 20, 6, 52, 49, 38, 1, 74, 73]. Brazma *et al.* [14] proposes a formal framework to classify sequence motifs discovery methods according the solution space, the ranking of the motifs, and the search algorithm. Their methods consider two sets of sequences used for sequence motifs learning: Positive sequences, which are the dataset from which motifs are found, and negative sequences used as a control for prediction classification. The solution space is defined by the overlap between motifs found in the positive and negative sequence set. The data is considered "clean" if the positive sequences contain only true positives, and none are in the negative sequences, whereas "noisy" data is the case where both datasets may have some overlap in motif distribution. To evaluate datasets which are clean, a simple 0/1 method can be used, where motifs are given a score of 1 if they match all the positive sequences and none of the negative sequences, and 0 otherwise. This is potentially the simplest and most straightforward evaluation method. However, in practice, we cannot assume to have absolutely correct data. Real DNA sequences used in a discriminative search can be noisy. With this consideration, Brazma *et al.* propose a more appropriate method of evaluating candidates using *sensitivity* and *specificity*. Let f be a motif. Sequences are counted as TP (True positive) if f matches it and the sequence is positive, a TN (True negative) if f does not match it and the sequence is negative, and a FP (False positive) if f matches it and the sequence is negative. The *sensitivity* is computed by:

$$Sn(f) = \frac{TP}{TP+FP}$$

And *specificity* is computed by:

$$Sp(f) = \frac{TN}{TN+FP}$$

While there is no absolute standard in terms of identifying a correct motif, both of these metrics offer insight on what the prediction means. These types of functions can be used to compare predictions scores from different methods, either individually or as part of a composite method. Multiple reviews of the field [14, 72, 20, 49] emphasize that comparing the performance of various motif discovery methods is a difficult task, as each method is designed to predict different kinds of patterns. The focus of these previous scoring functions are well suited for deterministic patterns expressed in the regular language space, where in general, the goal is to generate patterns matching a subset of provided positive sequences. Given a different method, dataset or motif representation (such as probabilistic models), this method may not be appropriate.

Motifs can be represented through deterministic or probabilistic methods. A word ($k-$mer or k-gram) is an example of a deterministic representation. They are often incorporated by methods looking for a word that is conserved or over-represented in the input sequences. Such enumerative word-based methods include WordUP, Oligo-Analysis, Dyad-Analysis, WEEDER and YMF [58, 75, 76, 66, 54]. Amongst those tools, WEEDER and YMF reportedly fare well in planted binding site experiments, though this performance comes with a high-stringency that may miss subtle patterns [20].

Methods that employ an enumerative approach, where the solution space is exhaustively searched, avoid the risk of getting stuck in local optimum, but also miss out on subtle patterns that can be hidden in noisy data. The Expectation-Maximization (EM) method is a probabilistic search process which can be used in identification of putative binding sites. EM methods are featured in various tools, such as MEME, LOGOS, PhyME, GIMF and ALSE [9, 80, 65, 60]. These algorithms usually choose a starting point (solution) to search for other candidates based on a characteristic to improve. For example, MEME improves an initial motif model over iterations in order to maximize its log likelihood across the sequences distribution through the EM.

Another category of probabilistic motif discovery tools are those produced through Gibbs sampling methods, such as the Gibbs Recursive Sampler, MotifSampler and GLAM, [71, 70, 29]. The idea of the Gibbs sampling algorithm is to iteratively build a solution composed of two data structures; one containing a sequence of variables to describe the pattern probabilities for occurrence and background frequencies, and another structure with position for the alignment of the pattern within the sequences. Initially, alignment positions are randomly selected. The rest of the algorithm alternates between a *predictive* step followed by a *sampling step*. The *predictive* step calculates the probability of the pattern according to the alignment positions in the sequences except one randomly picked sequence z. The *sampling* step tries to match pattern to each possible subsequence of z to assign them a weight. The weight influences the selection of the new alignment position for this sequence. Essentially, these two steps iteratively test pattern alignment positions and shift the position of alignments, one sequence at a time, according to the pattern match at the position that describes best the other sequences [20].

Alternatively, the methods can be looking for a word of length k with an approximate match in each input sequence, for example with no more than ϵ errors. In order to cope with errors and uncertainty, a probabilistic representation is sometimes desirable. This is more akin to the profile or Position-Specific Scoring Matrix (PSSM) representation. Typically, the PSSM consists of a fixed number of (sequence) positions. Profiles and the PSSM motif representations can be combined with an EM [9, 80, 65, 60], or Gibbs sampling [71, 70, 29], and can even provide probabilistic transcription factor binding sites representations. The PSSM is a matrix where each column represents a position in a pattern, and the values are probabilities for each four possible nucleotides. Position specific weight matrices are standard in many motif discovery tools, including MEME, SeSiMCMC and GLAM [9, 24, 29]. A PSSM is built as follows [49]: 1) Equal length sub-sequences representing predicted binding site locations are given as input and then aligned. 2) The position frequency matrix is built by counting the nucleotide occurrences on each subsequence per position. For a motif of length N, this creates a $4 * N$ matrix given that DNA has an alphabet of 4 nucleotides. 3) A position probability matrix is created from the frequency matrix. Before converting the position counts to a position probability, the zeroes are eliminated by adding a pseudo-count to each value. For example, using a pseudo-count of 1/4 for every nucleotide, the frequency matrix N becomes the *adjusted* frequency matrix $N\prime$ as follows: $N\prime_{a,i} = N_{a,i} + 1/4$. The probability matrix is created by dividing the adjusted count of a given nucleotide by the total number of adjusted nucleotide counts for the same position. Formally, the probability of the nucleotide a at position i is given by:

$$p_{a,i} = \frac{N\prime_{a,i}}{\sum_{A,C,G,T} N\prime_{a,i}}$$

Different pseudo counts can be used instead of 1/4, though there is no universal guideline on what the pseudo-count should be [49]. The question of pseudo-counts in PSSM has been studied more extensively in dedicated literature [3]. 4) The position probability matrix is transformed into a PSSM. The probabilities are converted into log-odds ratio scores ($s_{a,i} = ln\frac{p_{a,i}}{q_a}$). A nucleotide sequence and a motif of length n, are scored as follows:

$$S = \sum_{i=1}^{n} S_{a_i,i}$$

The main weakness of matrices when it comes to describing patterns is that they do not consider dependence between nucleotide positions within the binding site [52, 49]. For example, binding sites exposed through biological assays revealed cases where pairs of positions were dependent on each other [46, 23]. These patterns subtleties are not well represented by a PSSM.

It may be the case that the binding site representation is sufficiently well represented using a profile or PSSM, though it is possible to use a different model to capture dependence between binding site positions. A regular expression is an expressive representation for which efficient matching engines exist. Deterministic representations have a number of drawbacks, but their main limitation is the binary nature of a match (a motif is found or not). Still, DREME [7] is an example of a method using regular expressions as motif representations, and reports statistically significant short core motifs in faster runtimes than MEME or WEEDER.

A more involved representation would be, for example, a Hidden Markov Model (HMM). A motif represented by a HMM can be a sequence of nucleotides ordered by Bayesian inference. This has the benefit of capturing nucleotide interdependence. However, it has been shown that in some cases, the HMM underperforms against a PSSM model, as it may over-fit the problem [12]. HMM are also less intuitive and more difficult to compute than other motif representations, like the PSSM [52]. This suggests that the HMM may be of interest for certain motif discovery tasks, though PSSM offer an adequate representation with the added benefit of being easier to develop and match against sequences. Indeed, HMMs are reportedly more often used in protein domain discovery over DNA nucleotide motifs [49]

To address the sequence motif discovery problem, the authors of various reviews recommend taking a few considerations:

- Compare predictions between different motif discovery algorithms [6, 72, 14], as some motifs are missed by certain methods, while a combined view of predictions from the differents methods provides a better consensus of what is in the data.
- Use known motifs databases [6, 72], such as JASPAR [48] or TRANSFAC [78] to validate end results predictions.
- Motif discovery tools are inherently designed to favour certain types of motif discovery. A set of tools may perform better on data sets given properties such as their source organisms, or because of their sequences lengths [22].
- Interpretation of the predicted binding site requires a decent understanding of the input data sequences in order to make appropriate conclusions on predicted motifs. Kauffman *et al.* [38] recommend using existing data sets. It follows that since each motif discovery techniques have their limitations [49], it would be unfair to compare an existing tool to a new dataset chosen arbitrarily.

2.2 The MEME Suite

An extensive collection of software purposed for motif discovery tasks are offered through the MEME Suite [10]. This collection of software proposes 13 tools covering various use cases associated motif discovery. Interestingly, 4 different tools are implementations for motif discovery methods, including MEME and GLAM. Another interesting tool is DREME, which is reported to handle larger datasets faster than MEME. The suit also proposes a tool for motif comparison, Tomtom. Our primary interest is tools helpful in large scale high-throughput sequencing data analysis. The web version of the tools are publicly available at `http://meme-suite.org/`.

2.2.1 MEME

The original motif discovery tools from the MEME Suite, MEME [9], uses an EM algorithm to search in the Position-Specific Probability Matrix (PSPM) space for candidates representing frequently occurring motif is DNA sequences. A PSPM is a matrix with one dimension representing each position in a motif and the other with probabilities for each possible nucleotide position.

MEME is an optimization algorithm. Its objective function is a statistical significance test based on log-likehood ratio of a PSPM occurring on input sequences versus a background model. The background is an n-order Markov model based on the nucleotide frequency of the input sequences. The calculation of the log-likelihood ratio is given by Equation 2.1. The value $P(site|motif)$ is the probability of occurrence for the PSPM motif to be on a given site. The $P(site|background)$ value is the probability of occurrence using the background model instead of the motif PSPM.

$$llr = \log(\frac{P(site|motif)}{P(sites|background))}) \tag{2.1}$$

An E-value is computed from the log-likelihood ratio, the width of the motif, the number of occurrences for the motif, the background model (distribution of nucleotide probabilities), the length of the input sequences and the model (a parameter to determine the distribution of motif sites, e.g. whether a sequence may contain more than one motif).

According to the MEME documentation[1], the runtime is cubic with respect to the number of input sequences, and quadratic with respect to the number of characters. Reportedly, datasets with over 1000 input sequences can be intractable in terms of runtime due to the expensive p-value computation. A proposed parallel implementation addresses runtime considerations. ParaMEME [32], reportedly scales up to 64 nodes with 72% efficiency, though this inevitably requires the supporting hardware.

The MEME tool implementation has been under development and has seen improvements for more than two decades. We enumerate a few features that are worthy of mention for motif discovery interest:

- Support for DNA and protein sequence motif discovery
- Ability to search for different motif distribution profiles: OOPS (One Occurence per Sequence), ZOOPS (Zero or One Occurence Per Sequence), and ANR (Any Number of Repetitions).
- Biased search for motifs with when supplied position specific prior distribution of motifs sites locations.

[1] http://meme-suite.org/doc/meme.html (Accessed on 23-11-2015)

2.2.2 DREME

Discriminative Regular Expression Motif Elicitation (DREME) [7] is a discriminative motif discovery tool designed to find short TFBS. It is another tool available through the MEME suite. A major advantage over MEME is that it can correctly predict motifs on ChIP-seq experiment sequences in a shorter runtime than MEME. It enumerates short k-mers and applies Fisher's exact test using the counts of sequences matched to calculate the enrichment of that mer relative to two input datasets: A training set and a background for discrimination. If no background is provided, a dinucleotide shuffle [18] is applied on the training dataset to create a permutation of the original input.

Initially, a set of k-mers is chosen as seeds. The inner loop of DREME evaluates an extension over some pattern *seeds*, and estimates their expected probability (E-value) on the input dataset. The most significant patterns are kept to be used as seeds for the next generation. The outer loop marks what is captured on the input dataset and reports the top motifs until no improvements are found.

DREME's approach spends a fair amount of runtime searching the RE space in an enumerative fashion. This becomes more computationally intensive as the predictions grow, and requires additional repeated scans of the dataset for each extended motif it evaluates. However, experiments on 13 ChIP-seq datasets from mouse embryonic stem cells [7] show that DREME finds more known motifs (primary and co-factor) than MEME on the same 10 datasets. This is also with a runtime of 1153 seconds over 4869 seconds for MEME. While the motifs are shorter than MEME, they are long enough to match correctly to known motifs using a comparison tool, in a shorter amount of time than MEME.

2.3 Comparing motif discovery methods

A wide variety of motif discovery algorithms have been published and refined over the years. The task of comparing different methodologies is further complicated by the various tradeoffs which comes with each method. Earlier literature [72] comparing motif discovery solutions reports that it is a difficult task due to the variability of prediction accuracy. No method performs best on every dataset. The review authors compared 13 different tools and created test datasets using real planted binding sites. Predicted sites are evaluated using knowledge of the span and location of the binding sites. Metrics were developed to compute prediction accuracy. Predictions overlapping the planted sites and their nucleotides are counted and used for scoring. The scores for nucleotide metrics are:

- nTP: the number of predicted nucleotides in a motif which are part of a known site (True positive)
- nFN: the number of nucleotides in known sites that were not predicted in a motif (False negative)
- nFP: the number of nucleotides outside of known sites that were predicted in a motif (False positive)

- nTN: the number of nucleotides outside of known sites that were not predicted in a motif (True negative)

Using these metrics, functions of prediction accuracy and correlation are computed for each motif discovery tool. Two important examples include the nucleotide level performance coefficient [59] (nPC) and a nucleotide level correlation coefficient [16] (nCC). The two methods seem to score the different tools somewhat proportionally. A tabulation of each correlation coefficient (nCC) for each tool reveals that the highest-scoring tool in this regard is Weeder [55], YMF [66], ANN-Spec [79], and MEME [9].

Another set of 4 metrics for sites follow a similar approach. These counts are added when

- sTP: the number of known sites overlapped by predicted sites (True positive)
- sFN: the number of known sites not overlapped by predicted sites (False negative)
- sFP: the number of predicted sites not overlapping any known sites (False positive)
- The "true negative" does not exist at site level.

A measure of prediction accuracy at the site level is the site positive predictive value (sPPV), given by Equation 2.2.

$$sPPV = sTP/(sTP + sFP) \tag{2.2}$$

When looking at the sPPV, Weeder seem top be the highest-scoring tool, with other high-scoring methods including MEME. However, a table listing datasets for which tools predicted no motifs show that Weeder predicted no motif on 17 out of 56 datasets. The authors indeed mentioned that some tools will have high sPPV when they omit predictions on hard datasets. Tools which predicted motifs on every dataset include Improbizer [4] and SeSiMCMC [24] (which has 0 datasets with no predicted motif, but lower sPPV than MEME or Weeder). MEME fares well, with only 6 out of 56 datasets where no motif was discovered. In summary, MEME seems to offer a good compromise between nucleotide and site level prediction accuracy. It also produces solutions for most datasets while still scoring a good site positive predictive value. Weeder does not always make predictions, but predicts very accurate results when it does. It is also worth mentioning that the study also considered a variation of MEME called MEME3, which uses a more stringent objective function to obtain more accurate predictions. Evaluation of MEME3 shows similar nCC and sPPV scores to MEME, but reports no motif on more datasets than MEME. Another tool part of the same software suite [10] as MEME called GLAM was also studied. GLAM discovers no motif on only 3 out of 56 datasets, but its sPPV and nCC are much lower than MEME. GLAM is presented as a motif discovery tool able to perform gaped motif search, which serves a different purpose than generic motif discovery solutions, which is not explicitly investigated in this assessment.

In terms of generating synthetic datasets for testing, the authors admit that this method lacks the 'correct' answer. In a sense, there is no way to know if the planted site correctly represents what would be found naturally in biological sequences. Other methods of creating synthetic data, such as Markov chains, are also susceptible to introduce additional biases since this will not necessarily create DNA sequences with the same stochastic process used in nature.

The assessment datasets are available online[2]. The different types of assessment sequences provided are labelled as *real* (binding sites in real promoter sequences), *generic* (binding sites planted in random promoter sequences from organisms), and *markov* (Markov chain generated data with planted binding sites). Unfortunately, the datasets do not scale very high in terms of size. For example, the generic set contains a total 56 datasets of binding sites planted in a genomic promoter sequences. The largest one has only 70 sequences, and some of the smaller ones have 2 sequences. This is relatively small compared to a real sequencing experiment such as ChIP-seq [17], where we can expect tens of thousands of sequences.

Overall, this assessment illustrates some difficulties in evaluating competing motif discovery tools and offers a few methods to measure binding site prediction accuracy. There are many tools, but each method comes with a compromise. The datasets are a legitimate way to evaluate prediction accuracy, however, their size may not represent the same challenges posed by larger scale datasets, as they are harder to analyze and also require more computational work. Datasets with thousands of sequences with planted binding sites would be useful to assess methods against larger scale experiments, such as data generated from ChIP-seq experiments.

2.3.1 Tomtom

Tomtom [33] is used as a motif comparison and alignment tool. For a given input motif, Tomtom will compare it to every motif in a chosen database and reports the most statistically significant match. This tool is particularly useful in *de novo* motif discovery, where we do not know if the predicted sites from a motif discovery algorithm represent real TFBS.

For a given pair of motifs (one predicted and another from the database), Tomtom will first compute a p-value. The first step is computing the offset between the prediction and the site. For each possible offsets, a column similarity function is applied. The different functions provided by Tomtom all reliably normalize for varying lengths of motifs, though the default method is Pearson's correlation coefficient for motif comparison [62]. The score for each column (position) overlapping the motif is summed and converted into a p-value. An E-value is also calculated by multiplying the p-value against twice the number of database targets matched.

After a prediction analysis, Tomtom outputs the alignment offset, p-value, E-value, and a graphical logo representation of the alignment. The predictions with lowest E-values are considered the most statistically significant. This can be used for TFBS identification on

[2] http://bio.cs.washington.edu/assessment/

whether a reported prediction matches a database target expected to be in the dataset, if their computed E-value < 0.01. Alternatively, predictions aligning to the same target can be compared on the basis of their E-value to determine which one represents the binding site more accurately.

2.3.2 MEME-ChIP

A web service called MEME-ChIP [45] offers motif analysis. The software presents a user interface for processing tasks of motif discovery, motif enrichment analysis, motif visualization, binding site affinity analysis and motif identification. Most steps in this pipeline are handled by other tools of the MEME Suite, and the motif discovery is performed by both MEME and DREME. MEME is employed to find accurate binding profiles for TF. Additional predictions are provided by DREME. MEME can miss shorter motifs, which could be useful in further MEME-ChIP processing (e.g. to visualize a cis-regulatory module from multiple binding sites). While this tool is not a standalone motif discovery algorithm, it benefits from combining complimentary predictions from differing methods of motif discovery. MEME-ChIP will also support validation of the predictions against databases of known motifs using Tomtom.

2.4 Evolutionary Computing in motif discovery

An approach that was not covered in the previous reviews or the MEME suite is evolutionary computing. Here, we briefly mention an algorithm for motif discovery using a genetic algorithm, and a review of evolutionary metaheuristics.

2.4.1 GADEM

GADEM [40] is another motif discovery method using an EM algorithm, like MEME. However, it is embedded in a GA, evolving a population of gapped motifs called spaced dyads. For example, $ACNNNGTG$ is a spaced dyad where $a_1 = AC$, $x = 3$, $a_2 = GTG$.

The spaced dyads are formed as pairs of words joint by a fixed length gap. The dyads are converted into Position Weight Matrix (PWM)s. At each generation of the GA, multiple steps of EM are applied on the population individuals to derive new PWMs. A log likelihood ratio score [67] is used to compute the alignment of the PWMs (E-value) as a fitness score. Individuals with an E-value below a certain cutoff are kept in the population, and the remaining space is filled with offspring of randomly chosen individuals via crossovers or mutations.

The authors of GADEM report that the EM process over multiple sequences is computationally costly. They reported a runtime of 96 hours for a CTCF dataset of 13,721 CCCTC-binding factor), and 6-10 hours on smaller datasets (603 loci with OCT4 binding sites and 542 loci of P53 binding sites). While the datasets are not identical, this is far

from what DREME reported for similar size of datasets (at most a few hours for ten of thousands of sequences). The authors report that on a ∼419 kbp dataset, GADEM identifies 14 motifs in ∼5 hours, while MEME found 11 in ∼144 hours. The reduced runtime yielded by GADEM shows a good justification for the usage of a GA as part of the motif search, rather than elicitation as in MEME.

2.4.2 Multiobjective metaheuristics

A comparison between multiobjective optimization algorithms in the context of DNA motif discovery was recently published [30]. A total of 7 metaheuristics are compared using datasets and metrics from the previously mentioned assessment [72] of motif discovery tools. The tested metaheuristics include two standard multiobjective algorithm (SPEA2 [81] and NSGA-II [21]) as well as a few more sophisticated evolutionary heuristics such as Multiobjective Artificial Bee Colony (MOABC).

Every metaheuristic takes a different path to evolve candidates, though they follow some common rules in their selection. Motifs of nucleotides from single sequences are combined into a consensus motif representing multiple sequences. Each motif is optimized for support, similarity, and complexity. The first step is to align the candidate motifs on their sequences. The nucleotides of each candidate are combined to form an initial consensus motif. Individual candidate motifs which contain at least 50% of the nucleotides content from the consensus are kept and formed into the final motif. Additional constraints are also imposed on the predictions, such as a restricted length (motif should be between 7 and 64 characters), and a minimum sequence support count (e.g., motifs on datasets with 4 sequences or less are required to match 2 sequences or more, or else they are discarded.

A set of measures for multiobjective function evaluation were applied on the outputs of each metaheuristics. Algorithms are compared by a function called a hypervolume [21], which computes the fitness of a set of output solutions based on their multiobjective fitness. Across the different assessments, MOABC generally performs best compared to other metaheuristics. However, on 2 instances of *Drosophila melanogaster* datasets (out of 7) showed that NSGA-II (slightly) outperforms MOABC. On the *Homo sapiens* datasets, NSGA-II scores the highest hypervolumes in 3 datasets, while MOABC gets 10. Another metaheuristic, the Multiobjective Firefly Algorithm (MO-FA) obtains the highest hypervolume on 8 datasets, and the Differential Evolution (DEPT) scores the highest on 5 datasets. The *Mus musculus* instances also reveal a similar trend, with MOABC scoring the highest hypervolume on the most datasets (6), and NSGA-II, MO-FA and DEPT all scoring the best on 2 datasets each. In summary, MOABC scores consistently high hypervolumes, indicating that the optimization works reliably using this metaheuristics. NSGA-II also frequently outperforms other metaheuristics, despite being a standard multiobjective algorithm.

Other non-evolutionary motif discovery methods were also evaluated against the best performing metaheuristic (MOABC) and compared between their biological accuracy metrics. At the nucleotide prediction level, MOABC is compared to 13 other tools according to nucleotide sensitivity, predictive value, performance coefficient and correlation coefficient [72]. MOABC outperforms other algorithms in most cases, though a few datasets

(see Hs8, Hs10, Hs17 for example) shows that MOABC underperforms compared to other methods. At the site prediction level, MOABC still performs favourably, though it obtains mixed results in terms of site sensitivity ($sSn = sTP/(sTP + sFN)$). Overall, this experiment shows the merit of evolutionary metaheuristics in its ability to search for a motif in DNA sequences. While NSGA-II and SPEA2 are not the best performing algorithms in this assessment, they are reported as standard and serve as a good baseline for evaluation. The performance of multiobjective evolutionary computing solutions are susceptible to many variables, such as dataset sizes, choice of objective functions and distribution of binding sites in the sequences, which makes it favourable to have a benchmark such as these algorithms. There also is not a metaheursitics which universally outperform every other method.

2.5 Multiobjective Optimization

Multiobjective optimization provides a way to evolve solutions in evolutionary computing according to more than one fitness objective. Optimizing for more than one objective is not achieved by simply combining the scores of different fitness functions, but rather through a multiobjective evolutionary algorithm. In this section, we will expand on two multiobjective optimization algorithms which serve as a good baseline for our exploration of multiobjective evolutionary motif discovery. Further in this section, we explain how the fitness scores are conserved independently, and how solutions are ranked according to their relative non-domination.

2.5.1 Multiobjective Evolutionary Algorithms

While EA are powerful search heuristics, they can potentially be very expensive if they are not configured to search for solutions adequately. By using multiobjective optimization, it is possible to build Multiobjective Evolutionary Algorithms (MOEA). This type of EA is extended to have a refined selection mechanism that preserves diverse solutions through generations, based on multiple objectives and fitness scores.

In this approach, we use a fitness value to promote candidate selection as in any other EA, but instead of having a single function to compute a unary score, we use multiple functions to compute a tuple of scores. The values of tuples are not combined, but rather kept individually. When ranking different tuples of multiple objective scores, we are sorting them in fronts, such that each individual in a front is *non-dominated*.

2.5.2 Non-dominated solutions

A solution is said to be non-dominated within a set of solutions if there are no other instances in the set that scores higher in all of the multiple objectives. In this context, the status of domination and non-domination is always attributed in respect of a set of candidates. Consider a set of solutions where each solution is evaluated on two given

criteria, or objectives, of fitness that we are trying to maximize. Let *(A,B)* be a 2-tuple for the fitness, where A and B are real numbers from 0.0 to 1.0. A solution is non-dominated if the given objective A scores higher than any other solutions which scores a better value in objective B, and any other solution which has a higher score in A must have a small B value than itself. A non-dominated solution cannot be weaker in both A and B than any another solution in the set, or else it would be *dominated.*

In a population of solutions, we can pick up the first set of non-dominated solutions to compose the first Pareto-optimal front. We can repeat the process with remaining solutions and insert them in a subsequent set, or front of solutions, until all solutions are ranked. This means that by sorting non-dominated solutions, we have a different level of fronts, all which contains solutions that are not strictly better than another in regards to these multiple objectives. Note also that if a solution with a fitness of *(1.0, 1.0)* would dominate anything in the front, meaning we would have only one non-dominated solution. In a problem where candidate solutions have more than two objectives, we obtain non-dominated solution fronts in the same way by comparing each objective recursively.

2.5.3 NSGA-II

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [21] multiobjective optimization works on the basis of reproducing and altering solutions by favouring non-dominated solutions. NSGA-II is an extension over the standard NSGA algorithm with improvements over the population diversity due to a crowding distance operator for finer individual selection. The steps taken at each generation of the NSGA-II algorithm are listed in Figure 2.1.

Pseudo-algorithm

Solutions have two properties: 1) A rank in terms of non-domination, and 2) a crowding distance within that rank. The algorithm first divides the population by ranking the non-dominated solutions into fronts:

1. Initialize *Rank* to *0*
2. For each individual p in a population P, compare to all other candidates q of P

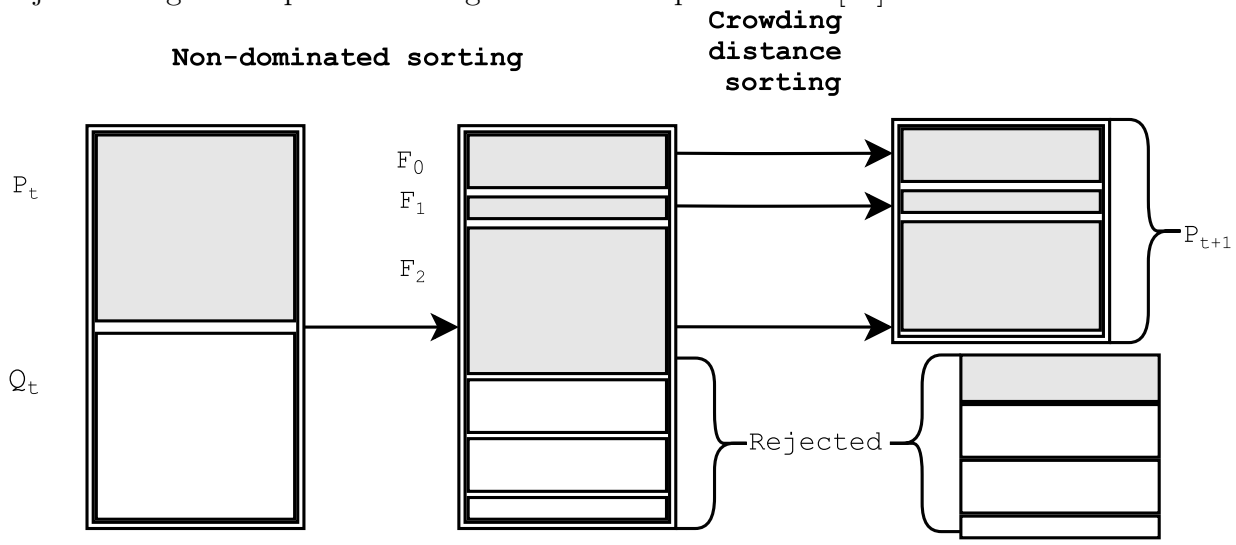
   If p dominates q, add q to the set of solutions dominated by p

   Else: Increment the domination count for p
3. Increment *Rank*
4. For each individual p in a population P,

   If p has a domination count of *0*, it belongs in $\mathrm{F}_{\text{rank-1}}$

   Else: p is temporarily kept in the front F_{rank}

Figure 2.1: Population selection and reproduction in a generation of NSGA-II. Initial population P_t and offspring Q_t are combined and sorted into k non-dominated fronts F_i, F_{i+1} .. F_k. The top fronts are preserved until the total number of retained individuals exceed the original size of P_t, and the remaining solutions are rejected. In this example, the solutions in F_2 would be sorted according to crowding distance such that only the best solutions are preserved. All subsequent fronts and remaining solutions from F_2 are rejected. Figure adapted from original NSGA-II publication [21].



5. If F_{rank} is not empty

   Go to 2. using the remaining solutions in F_{rank} as P

6. Else: Terminate non-dominated sorting

After this phase, we have a set of fronts that contains different slices of non-dominated solutions. Within these fronts, the NSGA-II algorithm then sorts the front by density. This helps EA convergence in general because it promotes population diversity, which is a desirable trait of the algorithm [21].

Given that the solutions are grouped in ranked sets, the solutions are sorted first by the order of the non-dominated front rank, and then by the crowding distance of that solution within the front. The crowding distance is a measure used to estimate the density as a means to order solutions within sets. Algorithm 1 shows how to compute the distance for all individuals in a given front F of multiobjective solutions.

The crowding distance is computed as follows: Each solution is given an initial distance of 0. For each objective $m \in M$, sort F on the m value of each individual. The first and last objective sorted with respect to m are given a distance of *infinity*. All other solutions have their *distance* value calculated according to line 9 of Algorithm 1. This sums the normalized distance of the individual to its nearest neighbours for each objective. After all objectives m have been used, we have a crowding distance for each individual in F.

Algorithm 1 Crowding distance computation

$N = |F|$
for $i = 1 \dots N$ **do**
  $F[i]_{distance} = 0$
end for
for $m = 1 \dots M$ **do**
  $Sort(F, m)$
  $F[1]_{distance} = F[N]_{distance} = \infty$
  for $i = 2 \dots N - 1$ **do**
    $F[i]_{distance} = F[i]_{distance} + \frac{F[i+1]_m - F[i-1]_m}{MAX(F_m) - MIN(F_m)}$
  end for
end for

For two solutions i, j with a *rank* and *distance* attribute, the partial order $\prec_c$ is defined by:

$$i \prec_c j := i_{rank} < j_{rank} \vee (i_{rank} = j_{rank} \wedge i_{distance} > j_{distance})$$

In this definition, i precedes j if it belongs to a lower non-dominated front rank, or else they belong to the same front and i has a higher crowding distance than j. This means that offspring from this algorithm will first come from non-dominated fronts, until the next front exceeds the maximum size of the population. Individuals from this front are added to the offspring until maximum population size is reached. The selection is done the basis of highest crowding distances, or in other terms, the least crowded solutions.

NSGA-II$_R$

The idea of crowding distance computation within NSGA-II has been recently revisited with some improvements on the crowding distance computation and the selection operators [28]. NSGA-II$_R$ brings modifications to NSGA-II to better handle cases where individual share the same fitness. Rather than comparing individuals for their crowding distance, the comparison is made on the set of their fitnesses, such that the score is independent of the fitness distribution in the population. The mating selection is also reviewed to use the unique fitness comparison, but also gives preferences to solutions with different fitnesses before considering two individuals with the same fitness.

With this change, a new procedure is needed to truncate the last front, since multiple individuals with the same fitness may overrepresent the front. NSGA-II$_R$ still adds fronts based on their non-domination rank first and foremost, but addresses the last front selection using the unique fitness crowding distance. The unique fitnesses are first sorted by crowding distances. A total of k candidates are selected by going over the lists of candidates with the associated crowding distances and selecting one randomly for each fitness. This allows individuals with different fitnesses to be inserted first, even if they have a higher crowding distance than individuals from previously selected fitnesses.

As with NSGA-II, the selection for mating individuals is also done through a binary tournament operator, but with modifications to promote selection of diverse candidates and which also prevents individuals with identical fitness from being compared. The procedure initializes an empty list S to insert N selected mating candidates from a population I. Fitnesse values from I compose the set F. While $|S| \neq N$, a selection of $k = min(2(|I| - |S|), |F|)$ unique fitnesses are selected in a random sample. The selected fitnesses are paired in a binary tournament comparing them using the crowded-comparison operator ($\prec_c$) using the crowding distance associated per fitness value. For each winning fitness, a candidate of I with the associated fitness is randomly selected and added to S. This step either returns $|I| - S$ individuals from unique fitnesses, at which point S is filled and the offspring are created. If not, then $|F|$ is smaller than the number of candidates required to make a tournament of unique fitness, therefore another round of selection is required until $|S| = N$.

2.5.4 SPEA2

Another multiobjective algorithm is the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [81]. It is an extension over SPEA. First, a brief description of SPEA. The algorithm keeps an initially empty *archive* and a random population. At the beginning of each generation, non-dominated solutions are added to the archive. If this insertion causes existing individuals in the archive to become dominated by the new insertion, the dominated solution must be removed. Once the archive is updated, a mating and variation phase is done in order to chose candidates for crossover and mutation between solutions from the archive and the population. The offspring of this mating phase replace the population for the next generation, and the process is repeated.

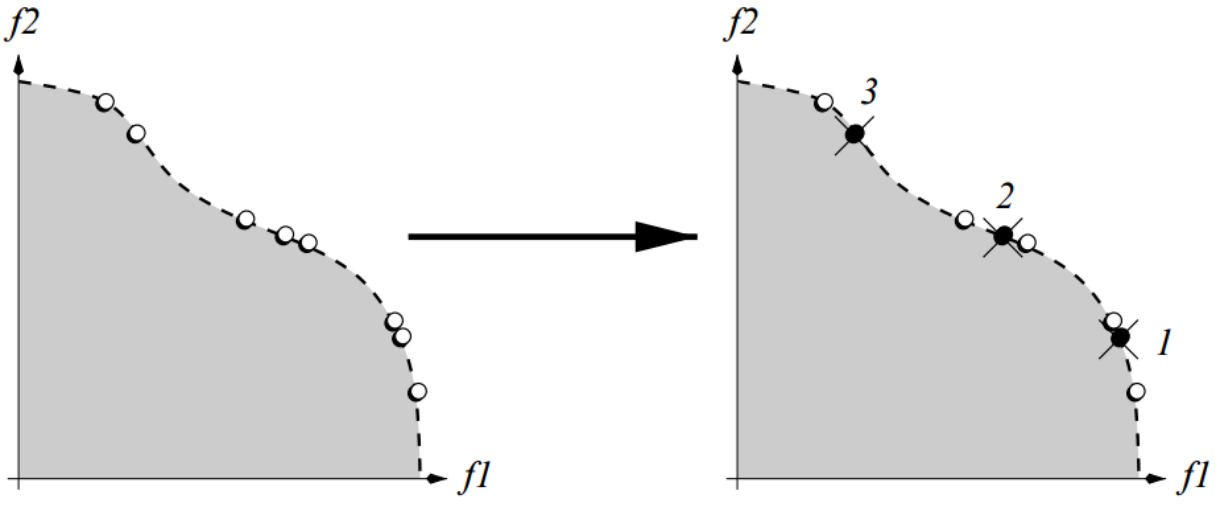


Figure 2.2: SPEA2 archive truncation. Axis f_1 and f_2 are objective functions and data points represent solutions with according fitness values. Assuming an archive size $\bar{N} = 5$, the second figure illustrates a truncation of the individuals with the highest density due to the proximity in the objective space relative to their neighbours.

SPEA2 improves on its predecessor (SPEA) in a few ways. First the archive is of a constant size, and is filled with dominated solutions if non-dominated solutions are not sufficient to fill the archive. Next, archived solutions are ranked on their *raw fitness*, which is a calculation of dominated and non-dominated individuals over the population and the archive, instead of simply the archive. This is necessary since now we have solutions in the archive that are dominated. Finally, the last major difference is that mating is only done in the archive. This means that there will always be a mating partner than is amongst the archived solution, while the population can still evolve to provide diversity to the archive.

Pseudo-algorithm

Listed below are the steps for the SPEA2 pseudo algorithm and Figure 2.2 represents the archive truncation operation [81].

- Input:

  N (population size)

  $\bar{N}$ (archive size)

  T (maximum number of generations)

- Output:

  A (non-dominated solutions)

- Algorithm:

  Step 1: **Initialization:** Generation an initial population P_0 and create an empty archive $\bar{P}_0 = \emptyset$. Set $t = 0$.

  Step 2: **Fitness Assignment:** Evaluated fitness for solutions of P_t and $\bar{P}_t$.

  Step 3: **Environmental Selection:** Copy all non-dominated solutions in P_t and $\bar{P}_t$ to P_{t+1}. If the size of P_{t+1} is smaller than $\bar{N}$, then add more individuals from P_t and $\bar{P}_t$ to $\bar{P}_{t+1}$. If there are more than $\bar{N}$ individuals, truncate the crowded solutions until $\bar{P}_{t+1}$ has only $\bar{N}$ solutions (see Figure 2.2).

  Step 4: **Termination:** If $t \geq T$, or an alternative termination criteria is reached, assigned the set of solutions in P_{t+1} to A. Terminate.

  Step 5: **Mating:** Perform binary tournament selection to add candidates from $\bar{P}_{t+1}$ to the mating pool.

  Step 6: **Variation** Apply crossovers and mutations on the mating pool P_{t+1} . Increment t and go to Step 2.

SPEA2 guide its selection process through a fitness assignment computed from the candidates multiobjective score. The SPEA2 fitness described here is a measure of non-domination within a Pareto front, and a density also computed using the multiobjective score of the individuals. The SPEA2 fitness function F is defined as:

$$F(i) = R(i) + D(i)$$

Where R is a measure of *raw fitness* and D is *density*. $R(i)$ is a sum of the domination count from objective dominated or equal by i, called the *strength* (S). Formally, the domination count of i, is:

$$S(i) = |\{j | j \in P_t + \bar{P}_t \wedge i \prec j\}|$$

Where $i \prec j$ denotes i is dominated by j according to Pareto dominance. S is computed for every solution in $\bar{P}_t$ and P_t. $S(i)$ represents the total number of individuals dominated by i in a population. The *raw fitness* is computed as follows:

$$R(i) = \sum_{j \in P_t + \bar{P}_t, j \prec i} S(j)$$

In other words, the *raw fitness* of i is the sum of individuals dominated by solution dominating i. If $R(i) = 0$, then i is a non-dominated individual.

The *raw fitness* provides a degree of elitism by considering the total number of dominated individuals in the population and archive, meaning solutions which identical domination count (S) can have different *raw fitness* (R) depending on which individual they each dominate. However, this measure is not sufficient when solutions do not dominate each other. This case is addressed by a *density* measure which discriminates between individuals with identical *raw fitness*. The individual *density* is the inverse of a distance function between i and its a k-th neighbour. The distance is obtained through the k-th nearest neighbour method [64]. The authors suggest using $k = \sqrt{N + \bar{N}}$ as a common parameter, which represents the size of $P_t + \bar{P}_t$ (i.e. the sample size). For a given individual i, the k-th nearest neighbour distance is denoted by σ_i^k. The *density* ($D(i)$) is computed by:

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

A value of 2 is added to the distance to ensure $0 < D < 1$, such that the *density* remains secondary to *raw fitness*. This becomes important during the environmental selection (Step 3). Individuals are first inserted into archive $\bar{P}_{t+1}$ only if they a fitness $F < 1$. This gives priority to non-dominated individuals since their raw fitness is 0. If the archive is larger than N, then it is truncated based on the σ_i^k distances. Individuals with the smallest distance to its first k-th neighbour is removed, and ties are resolved by next neighbours $(k+1)$. This is repeated until $|\bar{P}_{t+1}| = \bar{N}$. Alternatively, if $|\bar{P}_{t+1}| < \bar{N}$, the best $\bar{N} - |\bar{P}_{t+1}|$ dominated solution from $P_t + \bar{P}_t$ are added to $\bar{P}_{t+1}$.

Chapter 3

System

MotifGP is not only intended to be a standalone motif discovery tool, but also an extensible pattern mining engine for evolutionary algorithms. The different components in the system use a modular design to facilitate modifications and customization, such as changing the evolutionary algorithm, or the pattern matching mechanism. At this stage, the currently supported evolutionary algorithms and fitness metrics are those that are well suited for our intended scope: DNA *de novo* motif discovery for transcription factor binding sites in large ChIP-seq datasets.

With ChIP-seq datasets in consideration, we seek to have predictions that capture both strong and weak affinity TFBS. Affinity is the amount of interaction measured between a specific sequence of DNA and a protein (such as TFs) [68]. To capture motifs with various levels of affinity, we opted for a method that would preserve and evolve the solutions organized as Pareto fronts [21, 81]. Multiobjective approaches in evolutionary computing can rank solutions as a Pareto optimal set of solutions [21]. Our working hypothesis is as follows. Since solutions in a Pareto optimal set are strictly non-dominated amongst each other in terms of fitness, the various candidates in the front will cover different TFBS of varying affinity. Using objective functions to detect high affinity sites (such as frequent occurences in the input dataset), combined with discriminative measures against a control dataset, we can explore patterns which are truly overrepresented in the input dataset. By extension, marginal candidates in a Pareto front (those who are ranked at extremities) are those who scored very highly in a given objective with respect to the rest of the population. The Pareto front will also implicitly preserve diverse solutions, since we do not expect a TFBS to have a perfect score in all possible fitness objectives.

Genetic Programming is the chosen evolutionary computing method. The paradigm uses a template grammar to define a structure for solutions. This simplifies the solution search during evolution [44], as we will not evolve random, grammatically invalid solutions, that would not be contributing to solving our problem. The grammar ensures that randomly produced candidate solutions are at least pertinent to our problem. Enforcing solution sensibility using a grammar avoids having nonsense expressions that could not be evaluated. To extend the concept further, a template grammar can also consider types. Typing adds an additional constraint to the grammatical evolution of candidates, especially

on how the mutations or crossovers occur. In a Strongly Typed Genetic Programming setting, a candidate is an expression (in the mathematical sense) where each parameter of the function has type-checked inputs, and the inputs can be other functions from the grammar so long that its return value matches the expected type. This can also be represented as a tree of functions, values, and where all the leaf nodes are values.

This section will describe how we designed a DNA motif discovery tool using strongly typed genetic programming and multiobjective optimization, and how our implementation fits in a *de novo* motif discovery pipeline. Finally, we describe how we output our predictions and explain our proposed mechanisms added to optimize runtime.

3.1 De novo motif discovery pipeline

The proposed motif discovery tool uses Multiobjective Genetic Programming (MOGP) as its search heuristic. At the top level, the tool is analogous to DREME, in the sense that it provides motif predictions over sequences from ChIP-seq datasets without any instructions other than the input DNA sequences. The result of a motif discovery tool can be compared to TFBS databases using a motif comparison tool [33] and collections of species-specific TFBS. The comparison tool outputs a motif alignment over the target TFBS, as well as a significance score. The score is calculated by aligning the query pattern using matrices of weighted scores each representing a TFBS. The tool returns the query pattern's best found alignments amongst the given database matches.

The overall process is summarized by Figure 3.1.

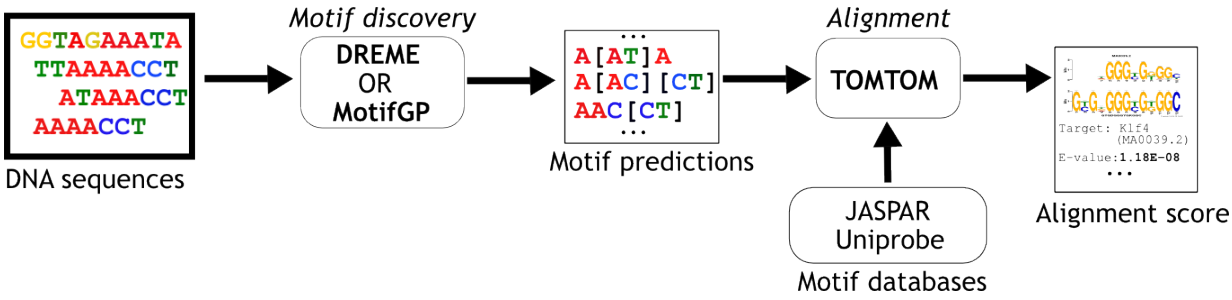


Figure 3.1: A *de novo* motif discovery analysis pipeline.

We used the same databases that were used in the DREME publication [7], though we opted for the 2014 version of the JASPAR database (see Section 4.2.1) which was not available at the time.

3.2 Evolutionary components

As in population-based evolution, MotifGP generates a population of candidates that will grow into solutions over generations. At each iteration, the population is evolved by means

of selection and reproduction. At this current design stage, we only use a conservative set of genetic operators and parameters, but more could be added for experimentation. These sections focus on the evolutionary mechanisms of the engine up until and excluding multiobjective optimization (covered by Section 3.3).

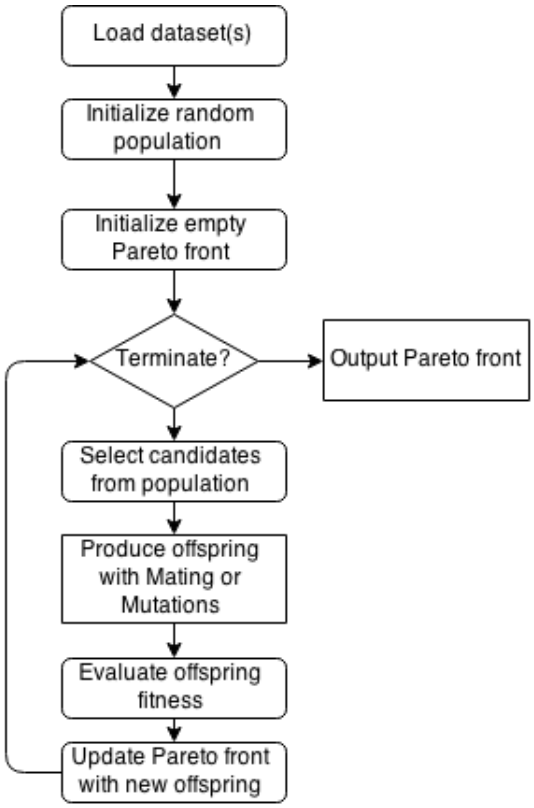


Figure 3.2: High-level view of the evolutionary loop in MotifGP.

3.2.1 Generational evolution

To evolve candidates according to multiple fitness objectives, we use two different multiobjective optimization algorithms; SPEA2 [81] and NSGA-II$_R$ [28] (a revised version of NSGA-II [21]). Both algorithms are reported to be good baseline methods for multiobjective evolution, and they share many similarities in implementation. Each requires an initial population and evolves it over successive generations by generating offsprings created in a mating/reproduction step. Where the algorithms differ is mainly in their selection mechanisms outlined in Chapter 2. NSGA-II successively combines non-dominated fronts into a population for the next generation, while truncation of the candidates is based on their crowding distance. SPEA2 inserts the first non-dominated front, and then inserts additional dominated solutions based on a fitness score determined by the non-dominated front rank first, and then by distances to k neighbours.

In both algorithms, this step can be abstracted as a selection operator. To generalize the use of multiple methods, the *Mu plus Lambda*(or $\mu + \lambda$) algorithm for evolutionary

algorithms is used as the MotifGP main evolutionary algorithm, with different selection operators for different multiobjective optimization algorithms. The generational evolution requires two specific parameters; *lambda*, which is the number of offspring to be produced in each generation, and *mu* which is the population size.

Algorithm 2 Mu plus Lambda pseudocode.

$ngen = 0$
$evaluate(Population)$
$ParetoFront = ParetoFront(Null)$
while $ngen < MAX\_GEN$ **do**
  $Offspring \leftarrow vary(population, lambda, cxpb, mutpb)$
  $evaluate(Offspring)$
  $Population = select(Population + Offspring, mu)$
  $ParetoFront.update(Population)$
  $ngen++$
end while
return ParetoFront

The function handling selection requires an input set of individuals (typically the population), and a number of candidates to select for the purpose of reproduction. The selection operators handle the ranking of multiobjective solutions, and the selection procedure for mating and selection the next generation's population. These steps are implemented through Environmental and Mating selection.

While a population is evolved, as described in Algorithm 2, a candidate selection is applied by retaining the non-dominated solutions . In SPEA2, this is done by selecting candidates to add an external "archive" of size *mu*, while NSGA-II keeps *mu* candidates from the top ranking non-dominated fronts of solutions between the population and *lambda* new offsprings. In SPEA2, the archive and population are used as a mating pools of candidates, with strong candidates for reproduction in the archive, and diversity in the population. This archive has a constant size of *lambda*: If the archive is smaller than *lambda*, then randomly picked individuals from the population are added to the archive. If it is larger than *lambda*, then the archive deletes candidates by means of a truncation operator [81]. In the latter case, the most crowded solution is eliminated from the archive in favour of other non-dominated solutions in lesser crowded regions of the Pareto front. Offspring are children of archived solutions after mutation or crossovers with other individuals from the population.

The NSGA-II algorithm uses a selection operator based on mating the population and offspring, and then preserving the non-dominated fronts. The combined candidate solutions are sorted using a *non-dominated sort* [21]. This sorting algorithm will order and group the solutions in fronts according to their relative non-domination. This means that the first front (F_O) will contain a set of solutions that are strictly non-dominated by the rest of the solutions. The next front, F_1, would contain solutions that may be dominated by solutions of F_O, but are non-dominated amongst each other and by fronts of subsequent ranks. There can be as many fronts up until there are *mu* individuals in all counted fronts.

In the last front, the individuals in the front are sorted according to their crowding distance within the front, which gives a slight penalty to solutions with similar fitness values. The *mu* individuals from the retained fronts are assigned as the new population for the next generation.

Pre-selection The NSGA-II$_R$ [28] algorithm is an improved version of NSGA-II that solves issues with the crowding distance operator on candidates with equal fitnesses. Before candidates are selected for reproduction (the *vary()* function in Algorithm 2), each candidate is assigned a crowding distance computed on their fitness. This will allow two things: 1) the selection of candidates for reproduction prioritize solutions of different fitnesses, and 2) Solutions that are selected from the last non-dominated front are not penalized if other solutions have the same fitness. Selection is also done on a unique fitness first basis. This works as a diversity preservation mechanism and avoids the same solutions from being picked up more than once in the same generation (before different solutions are selected first).

3.2.2 Genetic operators

In each generation, the modifications applied over the current population for offspring creation is handled by genetic operators. They are functions responsible for exploration and exploitation of potential solutions. This implementation uses a value of $cxpb \in [0.0, 1.0]$ as the probability percentage for the crossover, and implicitly a mutation probability of $mutpb = (1.0 - cxpb)$. The generational loop produces random values and checks them with the probabilities to determine if a given offspring should be produced via a crossover or a mutation on the parent(s). At this point, MotifGP implements only basic genetic operators to simplify the evaluations of our methods. The current operators are the following.

One-point crossover In this reproduction operator, the candidates are often represented as a string, list or array for simplified interpretation. However, our STGP implementation (Section 3.4.3) represents solutions as trees where elements of the candidates are nodes. In this case, the index is mapped to a tree node. The crossover is done by exchanging the subtrees at the index node between the two solutions. The methods returns a new tree representation for each of the two new children.

Let two candidates P_1, P_2 of length l_1, l_2 be chosen for reproduction. Randomly pick two indexes $idx_1 \in [1, l_1]$ and $idx_2 \in [1, l_2]$. Split P_1 and P_2 at idx_1 and idx_2 respectively, into 2 sets of half solutions P_1a, P_1b and P_2a, P_2b. Two new offsprings are created by concatenating $P_1a + P_2b$ and $P_2a + P_1b$.

Uniform mutation For a given candidate P_1 of length l_1 chosen for mutation, randomly chose an index $idx \in [1, l_1]$. Copy the candidate as a new child C_1 and modify the element at $C_1[idx]$ to a random value (compatible with the STGP from Section 3.4.3). More exactly, the value should be a new expression generated by a Primitive or Terminal function, such that the candidate program still produces a grammatically valid solution.

3.2.3 Termination and reproducibility

Due to the stochastic nature of evolutionary algorithms, the termination conditions are not explicitly defined. Moreover, since DNA motif discovery is NP-Hard [41], there is no guarantee of finding a global optimum for a given runtime. An EA can run for an infinite number of generations if termination is not set. The same EA can also produce varied results simply depending on the initial random state of the system. MotifGP employs a few measures to study reproducibility and proper termination.

Pseudo random number generator seeds MotifGP takes in an integer value as the *seed* for the engine's pseudo-random number generator's initial state. This ensures that two runs with the same parameters using the same initial seed will produce the same outcome at each generation. This can be used to compare the results of a given set of parameters across different initial random states. Populations from different seeds/runs can also converge towards different solutions since the initial starting population can impact the solution exploration.

Termination condition The engine can terminate under different conditions. The most straightforward method is a maximum number of generations; the engine will only run a certain number of generations and return the latest Pareto front. The alternative is to set a time limit, where the engine checks that the time limit has not expired before each new generation.

Checkpointing An exploratory feature in MotifGP is the ability to create checkpoints that can be reloaded and resumed for extended termination. A checkpoint is created on termination, and also at specific generational steps. For example; a given run may create a checkpoint at each step of 10 generations.

3.3 Multiobjective optimization for motif discovery

Our design differs from existing MOEA motif discovery algorithms by its choice of multi-objective function. Our approach was initially designed to use only 2 objectives; Discrimination and Support. We later added a p-value calculation with Fisher's exact test [25] (a hypergeometric probability value) as an additional objective. This section describes the relevance of each objective and how to compute them in the context of Network Expression matching over DNA sequence sets.

3.3.1 Network expression fitness

Our approach involves matching network expressions (modelled as regular expressions) over a set of sequences. Each candidate network expression is matched against all sequences of

the input dataset and the control set. Our fitness objective functions requires us to count 4 values. For both the input and control datasets, the number of sequences that the network expression matches, and the total number of sequences in the dataset. The calculations for our fitness functions are described in Section 3.3.2.

Consistent with the multiobjective optimization framework, the fitness of our network expressions are computed by multiple functions and are not combined to a scalar value, but rather expressed as a tuple. Each value of the tuple is computed by an objective function and a weight as in Equation 3.1.

$$Fitness = (s_1, s_2, \ldots, s_n) = (F_1 \cdot w_1, F_2 \cdot w_2, \ldots, F_n \cdot w_n) \tag{3.1}$$

The weights have a value of either -1 or 1. This is used to ensure compatibility between minimization and maximization objective functions. Since multiobjective optimization algorithms require all objective to work towards the same criterion (all maximization functions, or all minimization functions), a weight is used to mix objectives that are minimizing and maximizing in the same fitness. In our implementation, we added a negative weight to minimization functions (i.e. where their best value is the lowest, including negatives) such that the best fitness values are all implicitly maximizations.

3.3.2 Evaluation functions for motifs

The required parameters to evaluate motifs with our objective functions are the following:

- P - Set of input sequences
- p - Set of input sequences matched by the network expression
- N - Set of control sequences
- n - Set of control sequences matched by the network expression

For our metrics, we only use the cardinality of each set. One could opt to use the matched sequence sets to define objectives using further information, such instances of multiple matches per sequence. At this stage, we implemented three objective functions: Discrimination, Support, and Fisher's Exact Test.

Discrimination

Computed by dividing the total number of input sequence matches ($|p|$) over the total number of input and control sequences matched ($|p| + |n|$). A perfect discrimination (a score of 1.0) means an expression matches one or more positive input sequence, and none of the control sequences were matched. A discrimination score of 0.0 means no positive input sequences were matched, regardless of how many negative sequences are matched.

Conversely, the higher the number of input sequence matched versus control sequences matched, the better the discrimination.

$$Discrimination = \frac{|p|}{|p| + |n|} \tag{3.2}$$

Support

Support represents the total number of input sequences matched out of the total number of input sequences. A perfect support (1.0) is achieved by an expression that successfully matches all the positive input sequences. A support of 0.0 means the pattern matches none of the input sequences.

$$Support = \frac{|p|}{|P|} \tag{3.3}$$

While a very general simple expression (such as $NNNN$) would match any sequence of at least four nucleotides, and therefore have a support of 100%, it is not useful to our motif finding problem. However, it can still serve in crossovers as mating partners, or mutate into a more specific motif. While discrimination can also attribute high scores to trivial solutions (e.g. a unique pattern found in a single sequence in the input dataset would have a discrimination of 100%), solutions that perform moderately well in both discrimination and support should avoid these pitfalls since high-scoring trivial expression in one metric cannot possibly do well in the other.

Fisher's exact test

After initial testing of the first two objectives, we implemented Fisher's exact test as a new objective function. This function computes a statistical test returning the probability (p-value) of observing a given arrangement of events in a contingency table (Table 3.1).

The p-value indicates the deviation from the null hypothesis behind the case distributions in the table. A matched pattern that does not appear significantly more frequently in the input dataset relative to the control would not be accepted.

Each case in the table is presented by the number of observed successes and failures in a given instance of a candidate evaluation. We defined Fisher's exact test in terms of DNA motif discovery as follows:

Definition 1. *Let* $\boldsymbol{p}$ *be a pattern. Given two cases, a set of input DNA sequences and an independent set of control sequences; the contingency table* $\boldsymbol{M}$ *is formed based on the count of success in each case.*
Let a success be a given sequence where $\boldsymbol{p}$ *matches, and failures be sequences where* $\boldsymbol{p}$ *did not match. All sequences are evaluated into either a success or a failure given each pattern. The null hypothesis is that* $\boldsymbol{p}$ *matches the same proportion of sequences in both cases. If*

the null hypothesis is rejected, and $\boldsymbol{p} \leq 0.5$*, the p-value measures how highly represented patterns in the input DNA sequence set given a controlled environment. The lower the p-value, the more confident the test is.*

This method can be used to infer the probability of observing **p** in larger sets of the same type of sequences. MotifGP uses Fisher's exact test as an objective function that determines the significance of a given pattern according to its distribution in the dataset. Fisher's exact test is computed for each individual pattern, and requires the number of success for both case (input and control dataset), as well as the total number of success plus failures for each case (or total sequences in each dataset). The fitness values selected by the multiobjective algorithm should try to minimize the *p*-value, as opposed to other maximization functions (such as Support). This is corrected by assigning a negative weight to the Fisher's exact test's *p*-value in the fitness tuple.

We can adapt Fisher's work on 2x2 contingency tables *p*-value calculation [25] for sequence matching problems. Using our previous notations for parameters, the table is computed as as follows:

Table 3.1: Contingency table for discriminatory sequence matching.

	Input	**Control**	**Total**
Matched	p	n	p + n
Not matched	q = P - p	m = N - n	q + m
Total	P	N	S = p + n + q + m

Therefore, the *p*-value can be obtained by Equation 3.4.

$$Fisher = \left(\frac{\binom{p+n}{p}\binom{q+m}{q}}{\binom{s}{p+q}}\right) = \frac{(p+n)!(q+m)!(p+q)!(n+m)!))))}{p!n!q!m!S!} \tag{3.4}$$

Fisher's exact test has been used in other DNA motif discovery context, including DREME [7].

3.4 Strongly Typed Genetic Programming for regular expressions

While the multiobjective optimization algorithms ensure that we promote solutions that are diverse according to their fitness, they also implicitly influence what kinds of solutions can be expected. Herein, we use STGP to model what kinds of patterns are useful to our experiment. Our goal is to use the regular expression syntax (for pattern matchings) and reduce it to the minimum subset needed to express TFBSs. This subset syntax contains the basic elements required to draft a grammar for our GP approach. In order to generate expressions from this subset, we further divide the grammar into *Primitives* functions and *Terminals* values, so that they can be combined to form a valid expression from our

grammar. Finally, we also added type-checking constraints to *Primitives* and *Terminals* to ensure no invalid expressions can be created, essentially enforcing our grammar on randomly generated candidates.

3.4.1 Regular expression versus the probability weight matrix

A popular model for TFBS motifs in such problems is the PWM [61]. They are 2-dimensional matrices that represent a probability score of each alphabet character in the pattern for each position. The GP implemented in this tool compiles REs, which also describes one or more expected characters per position, but does not consider any positional probability. From this relation we can take a RE and translate it into a PWM. The only caveat is that the RE, by default, do not have any positional weights. Multiple valid alphabet tokens shared on a single position are therefore assumed to have equal probability. In return, using REs greatly simplifies various aspects of candidate generation, reproduction, and pattern matching. The GP template grammar could be modified to evolve PWM, as long as there is an efficient way to compute pattern matching, since this has to be done over the entire dataset, for each candidate at each generation.

It is also possible to extend the syntax of RE in order to express concepts that are not possible with PWM. For example, a variable range operator (*".{3,6}"*, a token for a range of anything of at least 3 and at most 6) is something that is not easily expressed with a PWM. One can also think of modelling short dependencies with 'or', for example, $(CAT)|(TAC)$.

3.4.2 Defining regular expressions motifs

The model used to represent motifs is based on the RE construct. To build such REs that are useful, a grammar is designed to represent how valid candidates can be programmatically produced. The previously mentioned notion of grammar is built by having functions as rules (*Primitives*), and an alphabet of values (*Terminals*). For our scope, the alphabet is the Deoxyribonucleic acid (DNA)'s nucleotide alphabet, which is represented by 4 characters. Rules are simply ways to combine theses terminals (or other primitives) into more complex expression. A basic primitive, *Concatenate*, two arguments of type *String* and concatenates them. Given two terminals A and G as leaves in a depth-2 tree where *add* is the root node, the resulting compiled solution would be the regular expression AG.

To create a grammar that can output valid individuals in the GP algorithm, we need to express two types of words:

1. Characters: single letter value representing the four basic nucleotides (Adenine, Cytosine, Guanine, Thymine) under the tokens A,C,G and T. Other valid IUPAC Nucleic acid notation can be created using a combination of these tokens and other RE modifiers.

2. Options from multiple characters: A token where a set of nucleotides options are valid tokens in the overall pattern.

This syntax is powerful enough to express motifs with varying degree of complexity. At this time, the produced solutions should also be fixed length patterns. For example:

Valid expressions:

```
ACCTACG; ACTT[AG]TT[AC]A, [ACTG]
```

Non-valid expressions:

```
AKTT (Should be A[GT]TT); A*CC.?T (Invalid words);

AA.{3,6}GT (Variable ranges not supported yet)
```

3.4.3 Template Grammar for DNA Network Expressions

Individuals in GP are composed with functions and parameters named *Primitives* and *Terminals* respectively. These are the basic components used to produce solutions. By randomly assembling *Primitives* into a tree where the 0-arity (leaf) nodes are *Terminals*, we can structure individuals into GP linear trees[37].

All created individuals share the same top-level type (a *string*). Below is an overview of the template grammar.

Strongly Typed Network Expression Template Grammar

```
# Root type
<String> Individual -> <String>

# Primitive
<String> Concatenation -> <String><String>
<String> CharacterClass -> <Char><Char><Char><Char>

#Terminals
<String> -> <Char>
<Char> -> A | C | G | T
```

Primitives

The Primitive nodes are functions using child nodes as type checked input parameters. They can be recursively used as parameters as long as typing constraints are followed.

- Concatenation: Combines two strings. Returns the concatenation as a string.
- Character Class: Represents a choice of possible characters as described by section 3.4.2.

Terminals

Terminal nodes are unary values the GP tree leaves. In order for the *Primitives* to produce REs, String and Char type terminals are used as parameters.

- String: Any combination of one or more Character. A single Character is also a String.
- Char: Single nucleotides, 'A', 'C', 'G', 'T'. This Terminal type differs from Strings as they are the only terminals that can be passed to the Character Class primitive.

3.4.4 Candidate Memoization

The most expensive part of the evolutionary engine can easily be the fitness evaluation, depending on the dataset size. A 4 MB ChIP-seq sequence dataset can contain around 40,000 sequences, which need to be matched against each offspring in each generation (and an equal number of sequences for the control dataset). An optimization for this problem is fitness memoization. Memoization is the process of storing a function's computational result in a cache according to its parameters. If the same parameters are called on the function subsequently, we can return the cached result.

Algorithm 3 Memoized Fitness Evaluation.

$regex = compile(Individual)$
if $regex \notin FitnessCache$ **then**
  $fitness = evalFitness(regex)$
  $FitnessCache_{regex} = fitness$
end if
return $FitnessCache_{regex}$

Grammar-based evolutionary algorithms have already been shown to be accelerated when using memoization [42]. Our approach is to cache the compiled candidates (as opposed to partial tree calculations) in order to avoid matching entire datasets for pre-computed solutions. Moreover, since identical patterns can be represented differently across two individuals (by having a different tree structure), we can avoid recalculating the fitness more than once.

Chapter 4

Methods and Experiments

This chapter describes experiments designed or adapted for the purpose of this research. The goal of each experiment is to evaluate the performance of MotifGP against ChIP-seq datasets under limited runtime. This is to ensure that this tool is efficient and relevant for practical uses, rather than using arbitrary test sequences. Moreover, the benchmarking datasets are the same as those from the original DREME publication in order to keep performance assessment fair and relevant. By comparing results reported from both tools using the same datasets and sought motifs, we can directly compare each method and identify their strong or weak points.

4.1 Mouse Embryonic Stem Cells ChIP-seq datasets

The main experiment's goal is to determine if MotifGP succeeds as a viable motif discovery tool. In order to do this, we compare MotifGP's motif predictions against those of DREME, a tried and tested motif discovery solution. Our assumption is that the datasets used in the original DREME publication constitutes a fair data mining challenge, and should not favour MotifGP (as opposed to selecting datasets that could be biased against DREME). This section describes the data representations for the input datasets and output motif predictions. We also list what is expected to be found as the primary targeted motifs.

4.1.1 Input data format

FASTA files are a standard format for DNA sequences. It was introduced with the FASTA software suite [56, 57] and is a widely adopted file format for sequencing data. Lines in FASTA files are paired by header and sequence. A header is a line starting with the character >, and all the following lines up until the next header constitutes a sequence (from DNA or proteins). For example, the first few lines representing the first two sequences of one of the datasets (CTCF):

```
>chr18:54419318-54419418
ATGCCGGCAAAAAATGTCATTATCATGAAGGCTGTTATAGTCAAGAGTGC
```

```
CCTCTGGTGGCCATTTAGACTGGCTATCCAGAGTGTACTGGGCCCAGGCT
>chr8:87308070-87308170
CTCTATTCCCTTTTTGGCCCGAAGAATCAAACACTGTTATACAGCCCCAC
TCCAGAAGGGAGCGCTTCCCATCCCAGACCCAGCAGTCTCCAAGCCAGGT
```

In this example, the first title header shows the DNA sequence comes from the 18th chromosome (*chr18*), and is found between genomic coordinates *54419318* and *54419418*. In the context of *de novo* motif discovery, the header can and should be disregarded as the algorithm should not fall back on any information other than the sequence at this time. This can, however, be used for post-analysis once motif predictions have been obtained. ChIP-seq peaks are generally expected to be at least 100bp in length [45].

4.1.2 Outputting the Network Expression fronts

The output of a MotifGP run is a list of REs and their fitness computed according to the input dataset and control sequences. Solutions are sorted on multiple fitnesses, and they are non-dominated solutions. The output represents a pareto-optimal front of solutions from the final generation of the population.

Our custom format to represent the output is the *.nef* file (for Network Expression Front). The format is tab-separated values, ordered from left to right by: Ranking in the sorted front, the network expression, and values for each fitness objectives used for evolution. For example, a compacted output of a network expression front using Discrimination/Fisher's exact test as the multiobjective function:

```
Rank         NE          Discrimination  Fisher
0          AAA[T]GACA          1.0          -6.940
1          AAA[CT]GACA         1.0          -6.940
                          ...
15   [CT][ACG][G]GAA         0.703        -111.0857
16   [CT][ACGT][G]GAA[ACGT]  0.688        -131.513
```

The files can be parsed or read as ordinary text files, or handled by most spreadsheet applications (such as LibreOffice Calc, Apple Numbers[1], Microsoft Excel[2]). A column-based representation for the .nef file is intended to facilitate interpretation of the solutions using sorting and filtering on multiple objectives.

4.1.3 Expected transcription factors binding sites

Due to the nature of the ChIP-seq protocol, different binding sites may be present in a dataset sequenced for a single TF. There can be many reasons for this, such as 1) co-factor

[1]Numbers® is a registered trademark of Apple Inc.
[2]Excel® is a registered trademark by the Microsoft Corporation.

motifs; gene expression can be regulated by multiple TFs [13]. 2) binding site affinity; the antibody in a ChIP-seq experiment may have bound to a TF which binds to weaker affinity sites, returning sequences containing different TFBS [63]. 3) A certain motif can have multiple representations. For example, a database may have different versions of the same motifs. Different databases may each have their own representation of a certain motifs. Sequences with different DNA orientations (normal or reverse) may also contain complemented representations of the same TFBS.

While primary target motif identification is important to understand transcription regulation, it is still difficult to determine with certainty which ones are functionally essential to a given protein [17]. With those biological constraints and uncertainties in place, we chose to evaluate transcription factor binding sites motif predictions in terms of the following priorities:

1. The more the query alignment to the database targets is significant (i.e. the lower the E-value attributed by Tomtom, the comparison tool), the higher the quality of the prediction.

2. The best prediction should not necessarily be the primary factor motif expected, as long as the E-value is < 0.01.

3. Finding multiple motifs that align to different sites are also of interest as they can be important to gene expression and co-factor regulation.

The output of a given MotifGP run will be compared to DREME's top motif predictions. The tool able to yield the motif with the most significant alignment (lowest E-value) will be considered better for that given run. See Section 4.3 for more details on the tabulation of different runs.

While the authors of DREME have published[3] their results for each dataset, we ran our own executions of DREME for a few reasons. First, we had to obtain a runtime for DREME using the same hardware available for our MotifGP runs, to ensure fair computational capabilities. Also, the motif databases originally used for validation in the initial publications have been updated and may contain new versions of motifs profiles that may improve the alignment score.

By cross-referencing the TFBS identified by DREME [7] with what is available in JASPAR and UniPROBE, we looked for any profiles of binding sites for primary factor and family members that could be relevant for each dataset. Table 4.1 shows each dataset's primary targets. Those can be expected to be highly overrepresented in the dataset with respect to other motifs.

[3]Results for DREME predictions are hosted at http://research.imb.uq.edu.au/t.bailey/supplementary_data/Bailey2011/.

4.2 Motif comparison and scoring

Motif predictions are compared to existing databases of profiles (position-weight matrices representing binding sites) and are scored using a motif comparison tool. When a query motif is submitted, there is no information about which TFBS is expected, or which dataset was used.

4.2.1 Position-weight matrices motif database

A common representation for binding profiles is the PWM. A profile is a 2-dimensional representation of a pattern. The profile's PWM contains one column for each position of the pattern. There are also 4 rows representing each nucleotide (A,C,G,T), and the sum of all cells in a column should be a value of at most 1.0. The motif database we use to verify predictions are represented by such PWMs. The database we use for the species (Mouse/Vertebrates) sampled in our datasets are JASPAR and UniPROBE.

JASPAR (5th edition) JASPAR is the largest open-access database for transcription factor binding site profiles covering various species. The latest edition of the JASPAR CORE Vertebrates database is composed of 202 non-redundant transcription factor binding site profiles [48]. While the full JASPAR collection contains up to 590 motifs, those additional profiles are for other species (such as plants, insects, and fungi) and not needed for mouse embryonic stem cell samples. The motifs found in JASPAR are compiled from various experimental sources. The profiles are manually curated from Protein binding microarray (PBM), HT-SELEX and ChIP-seq experiments for consensus on TF binding preferences.

Note that for the rest of this piece, any reference to a JASPAR database refers to the CORE Vertebrates version only, as other libraries of profiles would be irrelevant for our datasets and purposes.

UniPROBE (Mouse) The Universal PBM Resource for Oligonucleotide Binding Evaluation (UniPROBE) [51] is another open-access database of TFBS. It differs from JASPAR as it was created by compiling PBM experiments as opposed to ChIP-seq and motif discovery. UniPROBE does report some common motifs that are similar to those found in JASPAR. As with JASPAR, we are using the *mouse* library of profiles, and not other species (such as worms). The latest version of UniPROBE we obtained for our experiments contains 386 profiles.

4.2.2 Primary motifs per dataset

Table 4.1 shows the expected primary TFBS in 11 of the ChIP-seq datasets available in the aforementioned databases. While JASPAR covers most of the datasets, there are a few that require UniPROBE Motifs, such as *Oct4*. Since there's no Oct4 motif in either database,the

TF	Access #	Names	Database
CTCF	MA0139.1	CTCF	JASPAR
cMyc	MA0104.3	Mycn	JASPAR
	MA0059.1	MYC:MAX	JASPAR
	UP00060_1	Max	UniPROBE
E2f1	MA0024.2	E2F1	JASPAR
	UP00003_1	E2F3_primary	UniPROBE
Essrb	MA0141.2	Essrb	JASPAR
Klf4	MA0039.2	Klf4	JASPAR
nMyc	MA0104.3	Mycn	JASPAR
	MA0059.1	MYC:MAX	JASPAR
	UP00060_1	Max	UniPROBE
Oct4	UP00179_1	Pou2f3	UniPROBE
STAT3	MA0144.2	STAT3	JASPAR
Sox2	MA0143.3	Sox2	JASPAR
	UP00030_1	Sox11_primary	UniPROBE
	MA0515.1	Sox6	JASPAR
Tcfcp2l1	MA0145.2	Tcfcp2l1	JASPAR
Zfx	MA0146.2	Zfx	JASPAR

Table 4.1: List of known primary motifs (or similar representations) [7] from JASPAR and UniPROBE databases. For *Sox2*, we include any Sox family motif as valid predictions. We omitted some members (such as MA0078.1, Sox17) from the table if they were not relevant to results of predicted motifs. We also consider all Myc-binding profiles from JASPAR or UniPROBE as valid matches for the cMyc and nMyc experiments.

Pou2f3 motif was chosen since it is a related TF family member [7]. In addition to the reported primary targets from the DREME authors, we also parsed the databases for other relevant primary factor motifs that were available in the current (2014) [48] edition of JASPAR.

4.2.3 Motif comparison

To compare predictions from MotifGP, we use TOMTOM, a motif to motif similarity algorithm [33] compatible with JASPAR and UniPROBE. Given an input motif, we want to obtain the best alignments with profiles from all motifs in the given databases. It returns a set of candidate matches ranked by E-Value. The lower the E-Value, the more likely the reported motif is correctly identified by the given prediction. The published DREME results also referred to TOMTOM as their motif comparison tool [7]. We chose the same, again ensuring we are not selecting resources that favour MotifGP.

For compliance between TOMTOM's expected input format and network expression representation, we convert each of MotifGP's output motifs to individual position-weight matrices with equal positional probability for multiple characters in our *Character classes*

(see Section 3.4.2). Each matrix is individually evaluated by TOMTOM, and the score is mapped back to the original network expression.

An example query output to TOMTOM returns the following information:

```
#Query ID                  Target ID         Optimal offset
[AG]CCACGTG[AG][CG]          MA0104.3          0

#p-value             E-value             q-value
8.84146e-09          5.2253e-06          1.02615e-05

#Overlap          Query consensus          Target consensus          Orientation
8                  ACCACGTGAC                 GCCACGTG                  +
```

The output is separated in a three-line format for facilitated reading. The Query ID is the motif to align, while the Target ID is the name of the profile aligned from motif databases. The rest of the output describes the relation between the query and the database profile in terms of alignment and scores. For more detailed information, see the official documentation[4].

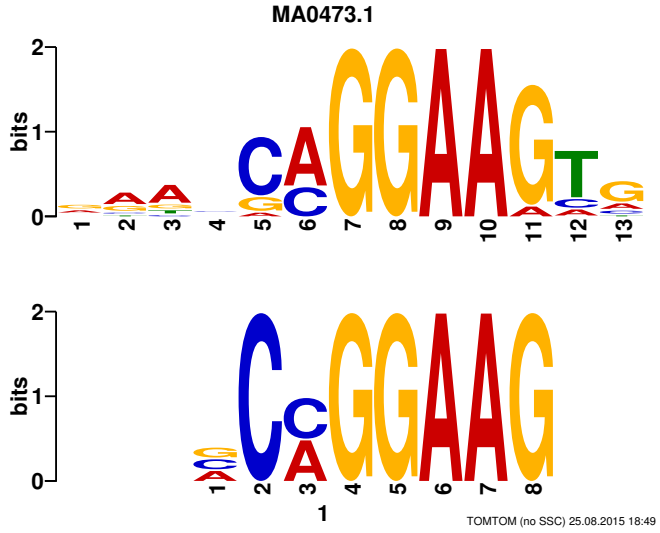


Figure 4.1: An example logo alignment.

TOMTOM also produces a graphical logo alignment like in Figure 4.1. The graphic contains the following items: Two profiles of nucleotides represent the database target, and the submitted query (i.e. the motif prediction). The database target, which is on top, also has its database accession number above (i.e. MA0473.1). The two strings of nucleotide logos are placed so they align in terms of positions. In this case, the position 1 of the query

[4]Official TOMTOM documentation from the MEME Suite http://meme-suite.org/doc/tomtom.html

is aligned to position 4 of the target. This means there is an *offset* of 4. The target also only spans 8 positions out of the possible 13, so the *overlap* is of 8. Stacked nucleotides represent multiple options at the same position. The relative size of each stacked character is related to the probability each nucleotide option is assigned. The database profile can have very different positional probabilities, meaning there can be uneven stacked characters such as position 11, where we strongly expect G, or a slight probability for an A. MotifGP's prediction uses equal probabilities for each nucleotide sharing a position (e.g. 50% split on position 3 of the query).

4.3 Experimental setup

Since EAs rely strongly on randomly generated values to initialize and execute the search process, it is important to test the program outcome according to the different initial value of the system's Pseudorandom number generator (PRNG). That value is called a *seed*.

Through the different experiments, we did multiple runs of the same set of parameters (multiobjective optimization algorithm, objective function, dataset) with a different seed to initialize the PRNG. By each experiment's criteria of success, we look to see if said successes replicate across different seeds. A consistent set of successes through all or most of the seeds indicates that the system is consistently performing well on the problem with the given parameter set. Situations where the algorithm performs poorly (little to zero successes) are also of interest as they indicate where the methods clearly fail, which can point to shortcomings and trade offs of some methods and parameters, or the evolutionary approach to the problem as a whole. Cases where overall success is ambiguous (say a rate of success around 10—50%) can indicate datasets where there are multiple TFBS, requiring more time to fully exploit the solutions.

Each experiment will require 10 seeded runs for each multiobjective algorithm. The runtime is determined by letting DREME terminate with its default settings on each dataset. We use that runtime as the termination time limit for each MotifGP run.

4.3.1 Experiment: Comparing alignment scores from DREME

As we mentioned, different methods of motif discovery on the same dataset may target different database profiles. In this experiment's context, we will disregard the profile itself, and instead look at the best alignment score. By verifying DREME's prediction with TOMTOM, we take the top scoring (E-value) prediction for each dataset. This value is used as the threshold of success; meaning a successful run for MotifGP is one where a predicted Motif outscores DREME's best alignment E-value on the same dataset. By tabulating the outputs of each 10 seed run for each parameter set, we will count how many successes occur over 10 runs. We also keep track of prediction with scores that are within a 3 order of magnitude from the success threshold. The DREME runtimes for each dataset are listed in Table 4.2.

Table 4.2: Runtimes per datasets for DREME using the tools' default parameters. The peaks are the number of unique sequences in each ChIP-seq dataset.

Dataset	Size	Peaks	Time limit (s)
CTCF	4.8M	39609	7691
Tcfcp2l1	3.3M	26910	5004
Esrrb	2.6M	21647	4054
E2f1	2.6M	20699	2769
Klf4	1.4M	10875	1504
Nanog	1.3M	10343	1987
Zfx	1.3M	10338	1258
nMyc	891K	7182	1418
Sox2	561K	4526	427
Oct4	467K	3761	362
cMyc	425K	3422	1418
STAT3	316K	2546	164
Smad1	140K	1126	71

4.3.2 Experiment: Primary factor profile identification

In the context of this experiment, we are interested in primary factor identification as the measure of success. A successful run is one where MotifGP can find the appropriate primary motif for a given dataset matching those described in Table 4.1. In this experiment, we do not necessarily expect to find the primary target in all of the 10 seed runs of a given parameter set, but rather if the primary factor occurs in at least one of the runs. Colloquially, these experiments look whether or not it is possible for MotifGP to discover a primary factor motif.

4.3.3 High performance computing for parallel benchmarks

Table 4.3: Listing of the different parameter and datasets combinations required for testing.

Algorithms	Objectives	Seeds	Runs	Datasets	Total
SPEA2 NSGA-II$_R$	Support Discrimination Fisher	[0 .. 9]			
2	**4 (DS, FDS, FD, FS)**	**10**	**80**	**13**	**1040**

With the different parameters to test, we end up with a total of 80 different combinations for 13 dataset. This requires 1040 runs in total (Table 4.3). In order to streamline the execution and evaluation process, we leverage high-performance computing to schedule all the jobs on single dedicated processor instances. After each run terminates (based on the DREME runtimes from Table 4.2), the TOMTOM analysis is launched so the resulting

motifs are aligned with known database profiles. The query motif and objective scores outputs (described in 4.1.2 as the *.nef* output) and the top-scoring target alignment from TOMTOM (from section 4.2.3) for each query are concatenated to a *.neft* file (*nef* + TOMTOM). The output is still a tab separated file now showing motifs, their fitness, and scores from external software. The intent is to keep the information format easy to sort and compare. The different runs were executed on the HPCVL (High Performance Computing Virtual Laboraties) Linux clusters. Each run is assigned a single, dedicated CPU to execute. We also executed DREME's runs on the same cluster. The technical specifications of the computing nodes are listed in Table 4.4.

Table 4.4: Observed hardware specs from the HPCVL High Performance Computing Linux cluster.

Operating System	**CPU**			**Memory**
Linux Red Hat 4.4.7-11	Vendor	Architecture	Observed clock rate	RAM
	Intel	x86_64	3059.043 MHz	64.0 Gb

4.3.4 Control sequence generation

Our evaluation process expects two equal size datasets; one as input, and a second one used as discriminative background for control over predictions. In our cases, the control dataset is not only used to validate predictions, but it is a requirement of our fitness function. It follows that the choice of control set can greatly improve (or hinder) predictions depending on what kind of information it shares with the input. For example, this work [53] studies the transcriptional networks of the TAL1 gene by discriminating in samples from leukaemic T-Cell lineages against TAL-1 expressing Jurkat cell lines (red blood cells).

While choosing proper control sequences needs consideration of the input datasets, we opted to use a random shuffling algorithm to produce similar sequences (in terms of dinucleotide content) but that differ in their ordering. Being able to produce such control datasets eliminates the need to manually select a control dataset. Instead, we create a shuffled set of sequences for control, on-the-fly, as the input dataset is loaded. This is good for usability since this algorithm works on any set of DNA sequences. This is the same approach employed by DREME [7] when no control set is provided (default behaviour). This is a more meaningful background than randomly generated sequences of DNA because this emphasizes discrimination on the nucleotide ordering.

Altschul and Erickson proposed a shuffling algorithm where they model DNA sequences as a directed multigraph G where vertices are nucleotides [2]. Two connecting edges represent a *doublet* (a pair of nucleotides in the sequence). The sequence can be expressed as an open walk (or path) on G. A shuffle of the sequence where doublets are preserved is equivalent to computing an Eularian walk on the graph [26]. The proposed algorithm works by shuffling the internal order of the traversed edged from the walk of the original sequence. By extension, a permutation of the edge ordering from the original sequence's

walk is a shuffled sequence that preserves dinucleotide usage. Our version of this dinucleotide shuffle is based on and nearly identical to the original implementation [18], except we handle DNA instead of RNA, requiring a switch from Uracil (U) nucleotides to Thymine (T). This algorithm also includes the nucleotides N, for any base, which is used within our datasets.

Chapter 5

Results

In this Chapter, we use the System described in Chapter 3 and discuss the results obtained when using the methodology described in Chapter 4.

We computed MotifGP's fitness score averages, minimum and maximum values for the Pareto fronts found over the 10-runs per parameter set and displayed the tabulation in Section 5.1. Section 5.2 is an analysis of the multiobjective scores for the Discrimination and Fisher's exact test fitness objective combination as it was the best performing combination. We first compare the scores obtained from our MotifGP runs, and we computed what DREME would obtain for its top predictions if it had been evaluated with these objectives.

Section 5.3 describes the combinations of parameters tested and their relative performances compared to DREME's top results. In each step, we compare a combination of multiobjective functions (e.g. Discrimination and Fisher's exact test). We display a tabulation of MotifGP's successes over 10 independent runs per parameter set, illustrating how often MotifGP offers more significant predictions than DREME. This is done separately for the NSGA-II$_R$ and SPEA2 algorithms. To further understand if there is a trend for success behind a certain objective function, we also plotted a graph with the summary scores between different summaries of objective functions.

Following the comparison between the independent seeded runs, we also explore the primary factor identification in Section 5.4, and we outline some anomalies we found, such as valid, yet unexpected TFBS predictions. We also explore the relation between queries and the database target to make sense of unexpected results.

5.1 Pareto front fitness values

After executing multiple runs of MotifGP, we were interested in what kind of fitness values can be observed over multiple executions over the same dataset. Looking at a single run and calculating the mean of values would not tell much about consistence. Different runs are not expected to have Pareto fronts with the same fitnesses, or even the same amount of

candidates. The goal of this section is to better our understanding of the relation between an objective score and the datasets we used. We can also compare the performance between multiobjective optimization algorithms to see if a certain method generally produces high or low scoring solutions with respect to a given objective.

Since we have 10 differently seeded runs per for two multiobjective algorithms, we decided to apply basic statistics over the compilations of fitness values from network expression front candidates. We calculate the maximum, minimum, and average values from the 10 combined runs, and the standard deviation for the average. As with other analysis, we chose the (Discrimination, Fisher's exact test) multiobjective function.

Table 5.1: A tabulation of the Discrimination objective found in all of the output network expression fronts, for each dataset and algorithm. We combined the Pareto fronts from the 10 individual runs and calculated the maximum, minimum and average values.

Dataset	**Algorithm**	**Max**	**Min**	**Mean**	**Std**
CTCF	NSGA-II$_R$	1.0	0.88	0.982	0.018
	SPEA2	1.0	0.921	0.982	0.018
Oct4	NSGA-II$_R$	1.0	0.621	0.896	0.093
	SPEA2	1.0	0.786	0.947	0.047
Smad1	NSGA-II$_R$	1.0	0.703	0.898	0.091
	SPEA2	1.0	0.598	0.858	0.111
cMyc	NSGA-II$_R$	1.0	0.669	0.913	0.068
	SPEA2	1.0	0.63	0.905	0.088
Zfx	NSGA-II$_R$	1.0	0.674	0.814	0.081
	SPEA2	1.0	0.597	0.804	0.093
STAT3	NSGA-II$_R$	1.0	0.75	0.93	0.055
	SPEA2	1.0	0.601	0.864	0.116
Tcfcp2l1	NSGA-II$_R$	1.0	0.677	0.917	0.067
	SPEA2	1.0	0.706	0.913	0.07
Nanog	NSGA-II$_R$	1.0	0.664	0.854	0.079
	SPEA2	1.0	0.65	0.873	0.075
Esrrb	NSGA-II$_R$	1.0	0.819	0.966	0.026
	SPEA2	1.0	0.914	0.971	0.018
Sox2	NSGA-II$_R$	1.0	0.726	0.912	0.063
	SPEA2	1.0	0.77	0.927	0.053
E2f1	NSGA-II$_R$	1.0	0.583	0.763	0.122
	SPEA2	1.0	0.583	0.77	0.119
Klf4	NSGA-II$_R$	1.0	0.578	0.93	0.069
	SPEA2	1.0	0.597	0.931	0.09
nMyc	NSGA-II$_R$	1.0	0.61	0.871	0.114
	SPEA2	1.0	0.61	0.881	0.104

Table 5.1 shows the compilation for the Discrimination objective. Systematically, MotifGP is able to find motifs with 100% Discrimination. Under both multiobjective opti-

mization algorithms. The minimum values are harder to interpret as they vary; they are likely the result of solutions with very significant (low, thus high-scoring) p-values from the other objective. What we can say, at least, is that Discrimination scores must be over 50% for them to be considered in the Pareto front. This is, again, a side effect of using Fisher's exact test as the partnered objective, since a low discrimination ($< 50\%$) would give a poor p-value, no matter what, and such solution would not make the Pareto-optimal front.

On average, there does not seem to be a major difference between multiobjective optimization algorithms and the resulting discrimination score. The mean score when comparing algorithms on the same dataset is fairly close. Except for Oct4, STAT3, and Smad1, all other datasets have only about a 1—2% gap between algorithms on the average discrimination scores.

E2f1 obtains an average score closer to the minimum than the maximum. Since the dataset produced unexpected predictions (see Section 5.4.2), it is likely that the disparity between Discrimination scores are caused by the primary target not being outstandingly overrepresented in the dataset.

Table 5.2: A tabulation of the p-value scores from the Fisher's exact test objective found on the output network expression front for each dataset and algorithm. We combined the Pareto fronts from the 10 individual runs of the Discrimination/Fisher's exact test objective function and calculated the maximum, minimum and average values.

Dataset	**Algorithm**	**Max**	**Min**	**Mean**	**Std**
CTCF	NSGA-II$_R$	7.064e-112	3.730e-2712	6.889e-1140	1.886e592
	SPEA2	6.299e-85	6.738e-2605	3.923e-981	4.433e577
Oct4	NSGA-II$_R$	3.738e-6	1.916e-124	3.239e-45	1.495e29
	SPEA2	1.129e-17	2.624e-113	4.213e-58	4.665e19
Smad1	NSGA-II$_R$	9.572e-4	2.452e-17	5.438e-10	3.215e3
	SPEA2	1.179e-4	5.889e-17	4.714e-9	1.739e3
cMyc	NSGA-II$_R$	7.479e-6	1.286e-199	2.571e-89	2.246e58
	SPEA2	1.499e-5	2.098e-199	9.733e-72	1.909e56
Zfx	NSGA-II$_R$	1.892e-6	8.450e-300	6.242e-127	8.098e89
	SPEA2	2.434e-4	8.450e-300	9.791e-109	5.677e90
STAT3	NSGA-II$_R$	1.202e-4	5.942e-132	4.256e-56	3.611e32
	SPEA2	9.680e-4	5.300e-143	1.656e-44	2.950e31
Tcfcp2l1	NSGA-II$_R$	2.419e-26	2.676e-1917	1.895e-444	1.640e315
	SPEA2	1.680e-18	7.931e-1364	3.261e-370	1.075e284
Nanog	NSGA-II$_R$	2.828e-11	3.025e-150	5.477e-62	2.599e38
	SPEA2	1.467e-8	1.018e-150	1.870e-60	4.661e36
Esrrb	NSGA-II$_R$	4.148e-35	1.616e-1143	1.608e-701	1.463e330
	SPEA2	3.137e-50	4.419e-1140	9.714e-690	1.489e286
Sox2	NSGA-II$_R$	5.781e-8	7.323e-212	2.466e-85	4.406e58
	SPEA2	2.546e-14	1.501e-201	9.550e-91	1.793e53
E2f1	NSGA-II$_R$	2.371e-7	1.715e-113	4.719e-55	1.463e29
	SPEA2	2.959e-8	2.127e-107	1.484e-53	2.885e27
Klf4	NSGA-II$_R$	7.582e-6	4.146e-505	7.208e-235	3.232e156
	SPEA2	7.331e-9	1.375e-509	1.242e-223	1.648e145
nMyc	NSGA-II$_R$	4.699e-7	5.250e-288	2.293e-95	2.185e86
	SPEA2	4.699e-7	1.873e-289	1.739e-97	2.493e85

Fisher's exact test is a bit more difficult to interpret: We do not have any knowledge of what is the optimal p-value, and how low can it go. Even when DREME motifs have strong p-values after Fisher's exact test (i.e. 6.7e-3472 on a CTCF prediction), a motif with a weaker p-value, such as those predicted by MotifGP, can still align with a stronger E-value once it is compared to existing databases of motifs. Our best prediction only has a p-value of 9.43E-817, yet its TOMTOM alignment is superior. This is also verifiable using our other predictions, which all obtained lesser p-value than DREME, but still manage (in most cases) to align better when using TOMTOM.

Another thing we have to consider about the p-values from Fisher's exact test is that they cannot be compared directly. The calculation of the p-value is dependent on the control sequences, which are randomly shuffled input sequences. While we can say the

same for Discrimination, it is actually possible to find a Discrimination of 100%; with a p-value, we have no way to tell what is the lowest p-value that can be reached. We are not able to compare motifs directly on their p-values, though a trend towards minimizing the p-value shows the optimization is working as intended. Discrimination seems to converge toward the 90—100% mark and may risk idling there if the other objective stagnates. While we cannot expect the best motif always to have a Discrimination of 100%, we know that a Pareto optimal solution with such a score must be amongst the most discriminative patterns.

Instead of looking for *max* values in Table 5.2, we are interested in the minimums (since Fisher's exact test is used as a minimization function). We notice that the mean value for most datasets is not close to either the minimum and maximum values, but rather "average". Also, since the p-value is an exponential distribution, the standard deviation does not describe a linear variation between the average p-values, but rather their variation in terms of order of magnitude.

What we can say for sure about from p-value of Fisher's exact test is that we cannot judge the fitness of a Pareto front according to this measure since the boundaries are dependent on the inputs.

5.2 Multiobjective fitness analysis

This section explores the fitness scores obtained by output Pareto-optimal solutions. We first look at the top-scoring motif in terms alignments results and look back on their fitness scores. We also computed what scores DREME would obtain if it had required the Discrimination metric. The section aims to understand the requirements and relation between the fitness objectives and a *good* prediction, something that is not fully understood by existing motif discovery software.

5.2.1 Descriptive analysis of top predictions

We are primarily interested in the results produced by the Discrimination and Fisher's exact test objective function as it is our best candidate multiobjective function for general purpose ChIP-seq dataset motif discovery. We gathered the best alignment scores and listed them in this manuscript. Table 5.3 (MotifGP) and 5.4 (DREME) are listings of the best motif predictions observed for each dataset. Tables 5.5 through 5.17 are side-by-side comparisons for both tools' graphical alignment logo produced by TOMTOM for each of their top predictions.

While the logos obtained by MotifGP's alignments are visibly longer than DREME, we can confirm that by comparing the *overlap* value of Table 5.3 and 5.4. This indicates how many bases the motif spans, therefore the length of the predictions. The overlap value in the tables also reveals that some of MotifGP's predictions are longer than the actual TFBS. The first indication of this comes from alignments with negative *offset*. Tcfcp2l1 and Klf4 predictions both to have this property.

We confirm that the binding site target from the database is actually smaller than the prediction by the logo from Table 5.11 and 5.16. Tcfcp2l1's prediction has 2 extra bases on the 5' end (towards the left, assuming normal orientation of the DNA) and 2 more on the 3' (therefore towards the right) end, versus the database target. This means the prediction is a total of 4 bases longer than the best know reference for the Tcfcp2l1 TFBS. In Klf4's case, we only have 1 extra base on the 5' end, but a full overlap otherwise.

Otherwise, there are predictions that span beyond the 3' end only. In Smad1, (Table 5.7), MotifGP predicts a motif with 2 additional positions, and DREME predicts an extra base as well. MotifGP and DREME produce similar motifs (the latter actually produces a subset of the former), meaning this common extra base is likely part of this dataset's motif, at the least in the samples sequences. They, however, map to different targets, MA0515.1 and MA0442.1 (Table 5.7). We observe the same phenomenon with Sox2 (Table 5.14): 2 extra bases for MotifGP's prediction and 1 for DREME . It is also worth mentioning that for the Zfx dataset, DREME predicts an additional base on the 3' end while MotifGP does not.

5.2.2 Relative enrichment and Discrimination of top predictions

Compared to MotifGP, DREME always find a top motif with a more significant Fisher's exact test p-values. Table 5.18 indicates that DREME likely finds the best p-value obtainable via Fisher's exact test, while our approach manages its prediction by allowing this metric to be compromised in favour of an alternative measure of success, which contributes to our goal.

Table 5.18: Enrichment (p-value) for top motif predictions. Motif representation from MotifGP have been converted to the IUPAC nucleotide alphabet from their original Network Expression notations.

Fisher's exact test	DREME		MotifGP	
Dataset	Top prediction	p-value	Top prediction	p-value
CTCF	AGRKGGCR	**6.7e-3472**	YGCCMYCTRSTGG	9.430e-817
Tcfcp2l1	ADCCRG	**1.8e-1049**	DNCCRGHNNVADCCRGHH	6.932e-167
Esrrb	RAGGTCR	**2.2e-2325**	TGACCTTGRVC	3.421e-193
E2f1	VGGAAR	**3.3e-177**	BCACTTCCGS	1.938e-29
Klf4	GGGYGKGK	**4.4e-892**	VGCCMCRCCCH	6.395e-254
Nanog	ACAAWRS	**2.1e-247**	RRRACAAWGV	1.612e-76
Zfx	GGCCTNB	**6.0e-366**	GGCCTVGGCS	3.728e-29
nMyc	CACGTG	**1.9e-289**	SCACGTG	1.156e-206
Sox2	ACAAWRGV	**4.4e-334**	RRRACAAWGGVV	1.666e-52
Oct4	ATGBWAA	**2.4e-130**	VCMGGAAG	7.590e-13
cMyc	CACRTGS	**3.6e-194**	SCACGTGG	7.434e-71
STAT3	CCRGDAA	**6.8e-109**	TTCCNRGAA	1.824e-83
Smad1	ACAAWGV	**2.7e-028**	VRACAAWGVV	1.669e-13

Table 5.19: Scores of the Discrimination objective for top motif predictions. DREME's discrimination has been calculated by using the DREME output files and applying the Discrimination metric on its reported input and control sequence match counts. Motif representation from MotifGP have been converted to the IUPAC nucleotide alphabet from their original Network Expression notations.

Discrimination	DREME		MotifGP	
Dataset	Top prediction	Score	Top prediction	Score
CTCF	AGRKGGCR	0.931	YGCCMYCTRSTGG	**0.998**
Tcfcp2l1	ADCCRG	0.692	DNCCRGHNNVADCCRGHH	**0.996**
Esrrb	RAGGTCR	0.913	TGACCTTGRVC	**0.997**
E2f1	VGGAAR	0.597	BCACTTCCGS	**0.947**
Klf4	GGGYGKGK	0.855	VGCCMCRCCCH	**0.968**
Nanog	ACAAWRS	0.740	RRRACAAWGV	**0.890**
Zfx	GGCCTNB	0.660	GGCCTVGGCS	**0.849**
nMyc	CACGTG	0.911	SCACGTG	**0.918**
Sox2	ACAAWRGV	0.894	RRRACAAWGGVV	**0.979**
Oct4	ATGBWAA	**0.832**	VCMGGAAG	0.764
cMyc	CACRTGS	0.874	SCACGTGG	**0.968**
STAT3	CCRGDAA	0.841	TTCCNRGAA	**0.986**
Smad1	ACAAWGV	0.764	VRACAAWGVV	**0.886**

We also tabulated the discrimination score for MotifGP and DREME's prediction. It is not surprising that MotifGP provides better scores than DREME since the latter does not explicitly take Discrimination into consideration for its optimization. Still, we see one instance (Oct4) where DREME actually finds a motif with higher discrimination than MotifGP. This happens to be a dataset where MotifGP fails to outperform DREME regardless of the objective function or multiobjective algorithm. We also notice that the DREME predictions that happen to have a high discrimination score is also amongst its highest scoring motifs with regards to the TOMTOM alignment (see Table 5.4, motifs CTCF, Klf4, Oct4 which have discrimination between 83–93%).

Overall, this illustrates the trade-off between using only Fisher's exact test and a multiobjective approach that includes it. Fisher's exact test is a good measure to calculate the relative enrichment for pattern occurrence in DNA sequences, but we suspect the nature of ChIP-seq datasets require a compromise between discrimination and enrichment, in order to capture more precise, fully developed predictions. Alternatively, this might not hold up if we were to use different kinds of control sequences. In further studies, this information can be used to gain insight on the relation between the objectives performance and the datasets. Perhaps certain objectives converge towards specific solutions, and runtime is mostly dependent on the dataset size, or its relative enrichment over certain types of control sequences.

5.3 Benchmarking alignment against DREME

In this section, we compare the different set of parameters (4 sets of objective functions) over the 13 datasets, on 10 independent seeds, for the SPEA2 and NSGA-II$_R$ multiobjective algorithms. We initially tested our original design, which is the Discrimination and Support objective function. Following that, we added the Fisher's exact test objective to make this a triple-objective function. We then tested Fisher and Discrimination as well as Fisher and Support, to see if one of the objectives of the triple-objective function is redundant with respect to the two others. The output for each 10 runs set is parsed independently to count how many obtained a better TOMTOM alignment than DREME's best prediction (with regards to the E-value). This is done for each algorithm for comparison between each method.

In the case where there are fewer than 10 successful runs, we looked for the highest-scoring alignment from unsuccessful run (called the runner-up), and listed the E-value in the column right of the DREME's benchmark. This should represent if valid prediction were close to the scored obtained by DREME. In this section's table, a score of **10** above DREME's benchmark means all the individual seed runs obtained a better prediction than DREME. **0** means there were no successful runs in that goal. Overall, we are seeking combinations of algorithms and objective function that produce high number of runs superior to DREME's for as many datasets as possible.

5.3.1 Discrimination and support

MotifGP discovered higher scoring motif alignments than DREME and all 13 datasets. The original objective function does not have any runs where we observed **0** successful runs. It also obtain better scores than DREME in more than half of the runs on most datasets. Another indicator that the objective function is a successful pairing is that there are datasets where we observe perfect scores (**10**) regardless of the multiobjective algorithm. NSGA-II$_R$ got a perfect 10 on 6/13 datasets, while SPEA2 gets 5/13 (coming close with a 9 out of 10 on the STAT3 runs, where the failure was within only one order of magnitude from DREME's). For 9 out 13 experiments, MotifGP outperforms DREME for 6 or more runs out of 10. Ergo, using this objective function, we are able to find improved predictions compared to DREME, on these specific datasets, using DREME's runtime.

Table 5.20: Comparing DREME and MotifGP using the Discrimination/Support objective function and the NSGA-II$_R$ multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	2	8.59888e-05	0.000127758
CTCF	1	3.50248e-07	4.44806e-07
E2f1	10	0.0385603	-
Esrrb	10	0.000574497	-
Klf4	5	3.28351e-08	4.76909e-08
Nanog	8	0.000967177	0.00123737
nMyc	10	0.00121468	-
Oct4	1	7.16982e-05	0.000363135
Smad1	6	0.00107949	0.00831152
Sox2	10	0.00539771	-
STAT3	10	0.0323896	-
Tcfcp2l1	6	0.00307791	0.00355356
Zfx	10	0.230923	-

Table 5.21: Comparing DREME and MotifGP using the Discrimination/Support objective function and the SPEA2 multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	5	8.59888e-05	0.000212988
CTCF	5	3.50248e-07	6.74086e-07
E2f1	10	0.0385603	-
Esrrb	10	0.000574497	-
Klf4	7	3.28351e-08	4.76909e-08
Nanog	8	0.000967177	0.00106304
nMyc	10	0.00121468	-
Oct4	1	7.16982e-05	0.00204809
Smad1	3	0.00107949	0.00207339
Sox2	10	0.00539771	-
STAT3	9	0.0323896	0.0838299
Tcfcp2l1	8	0.00307791	0.00414856
Zfx	10	0.230923	-

We found that 4 datasets with runs under both algorithms which had a success rate of 50% or less, though they are not exactly the same (cMyc, CTCF, Klf4, Oct4 for NSGA-II$_R$, and cMyc, CTCF, Oct4 and Smad1 for SPEA2). Referring to the dataset sizes from

Table 4.2, we see CTCF is the largest dataset, and MotifGP performs poorly compared to DREME with this objective function. However, the next three largest datasets (Tcfcp2l1, E2f1, Essrb) all pass the 50% success mark. On the Tcfcp2l1 dataset, MotifGP scores of 6/10 and 8/10 successful runs for NSGA-II$_R$ and SPEA2 respectively. On E2f1 and Essrb, it got 10 out of 10 on both multiobjective algorithms. With these results, we have evidence that Discrimination and Support is a well suited multiobjective fitness function for large datasets, though the largest, CTCF, remains problematic.

5.3.2 Triple-objective (Discrimination, Fisher's exact test, Support)

As we wanted to improve the former multiobjective function, we implemented an additional fitness objective to the original fitness functions of Discrimination and Support to make it a triple objective function that calculates the p-value as well.

Table 5.22: Comparing DREME and MotifGP using the Discrimination/Fisher's exact test/Support objective function and the NSGA-II$_R$ multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	7	8.59888e-05	0.000160341
CTCF	4	3.50248e-07	1.22103e-06
E2f1	10	0.0385603	-
Esrrb	9	0.000574497	-
Klf4	3	3.28351e-08	4.76909e-08
Nanog	7	0.000967177	0.00100131
nMyc	10	0.00121468	-
Oct4	1	7.16982e-05	0.000160021
Smad1	5	0.00107949	0.00218454
Sox2	10	0.00539771	-
STAT3	10	0.0323896	-
Tcfcp2l1	9	0.00307791	0.006329
Zfx	10	0.230923	-

Table 5.23: Comparing DREME and MotifGP using the Discrimination/Fisher's exact test/Support objective function and the SPEA2 multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	3	8.59888e-05	0.000160989
CTCF	2	3.50248e-07	3.58707e-06
E2f1	10	0.0385603	-
Esrrb	10	0.000574497	-
Klf4	5	3.28351e-08	4.76909e-08
Nanog	9	0.000967177	0.00209382
nMyc	10	0.00121468	-
Oct4	0	7.16982e-05	0.000185176
Smad1	2	0.00107949	0.00192347
Sox2	10	0.00539771	-
STAT3	9	0.0323896	0.0575364
Tcfcp2l1	6	0.00307791	0.0065291
Zfx	9	0.230923	-

Looking at the runs that obtained a score of **9** or **10** in the triple-objective function, we get the same number as with the Discrimination/Support function. This function also

yields 4 datasets with a success rate of 50% or less for NSGA-II$_R$, and 5 for SPEA2.

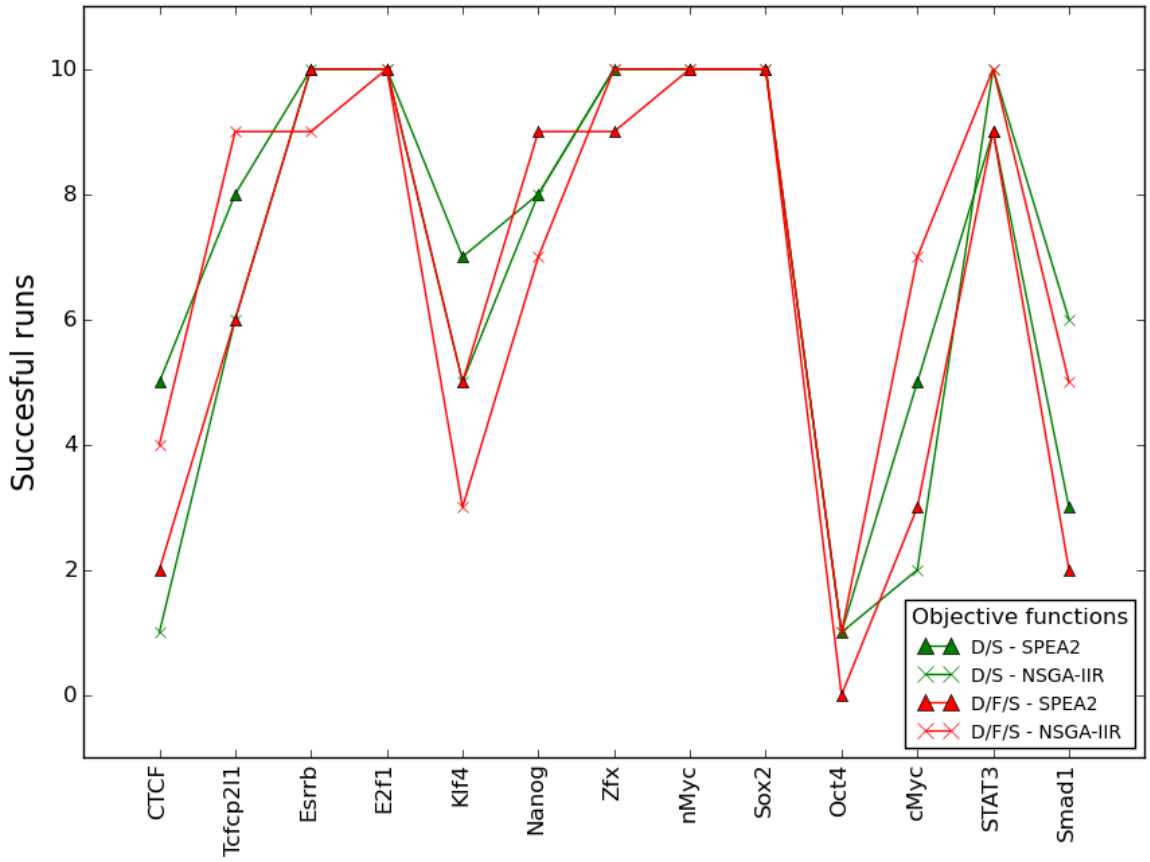


Figure 5.1: Comparison of dataset successes of Discrimination/Support versus Discrimination/Fisher/Support. The datasets are listed from largest to smallest in sequence count.

Using the results presented from configurations so far, we plotted the success counts (Figure 5.1) to visually represent the performance of each method per dataset. The plot shows an inconsistency between the optimization algorithms: In cMyc, we see the best and worst method uses the NSGA-II$_R$ algorithm while SPEA2 is more conservative yet consistent.

While the tangents are mostly close together, indicating similar performances, we can look at the most successful algorithm and objective function to determine the best approach per dataset. The Discrimination/Fisher/Support (red) variants obtain the best scores on cMyc, Nanog, Tcfcp2l1. For Discrimination/Fisher, the triple-objective function is stronger on CTCF, Klf4 and Smad1. Unfortunately, this does not seem like enough information to correlate datasets with methods. There does not seem to be a preference for a specific multiobjective algorithm on the basis of datasets size alone.

If we compare multiobjective algorithms alone, we can say that SPEA2 performs better under the basic Discrimination/Support schema, as it outperforms the triple-objective in 7

datasets (cMyc, CTCF, Klf4, Oct4, Smad1, Tfcp2l1 and Zfx) and is only outperformed in one dataset (Nanog). When comparing the NSGA-II$_R$ algorithm, the Discrimination/Support outperforms the triple-objective runs 4 to 3. This assessment suggests when looking at optimization algorithms alone, the triple objective function does not necessarily improve the simple Discrimination/Support objective function. However, the total number of successes (Table 5.24) is higher with the Triple-objective, suggesting that while their performance on a dataset basis is variable, the Triple-objective is a more stable, general-purpose method. For that reason, we will compare further results in Section 5.3.3 to both the methods as it is not clear if one method is definitely better than the other.

Table 5.24: Number of successful runs for the multiobjective functions Discrimination/-Support and Discrimination/Fisher's exact test/Support.

	Discrimination/Support	**Discrimination/Fisher/Support**
NSGA-II$_R$	89	**95**
SPEA2	**96**	**96**
Total	185	**191**

5.3.3 Fisher and discrimination

In this section, we are investigating whether the triple-objective fitness function is redundant. First, we removed the Support function to have a Discrimination/Fisher's exact test fitness function.

Table 5.25: Comparing DREME and MotifGP using the Fisher's exact test/Discrimination objective function and the NSGA-II$_R$ multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	6	8.59888e-05	9.87354e-05
CTCF	10	3.50248e-07	-
E2f1	10	0.0385603	-
Esrrb	10	0.000574497	-
Klf4	6	3.28351e-08	2.57034e-07
Nanog	9	0.000967177	0.00105348
nMyc	9	0.00121468	0.00121468
Oct4	1	7.16982e-05	0.000523792
Smad1	5	0.00107949	0.00395272
Sox2	10	0.00539771	-
STAT3	10	0.0323896	-
Tcfcp2l1	7	0.00307791	0.00324994
Zfx	10	0.230923	-

Table 5.26: Comparing DREME and MotifGP using the Fisher's exact test/Discrimination objective function and the SPEA2 multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	6	8.59888e-05	9.87354e-05
CTCF	9	3.50248e-07	4.65744e-06
E2f1	10	0.0385603	-
Esrrb	10	0.000574497	-
Klf4	6	3.28351e-08	2.57034e-07
Nanog	9	0.000967177	0.00165201
nMyc	9	0.00121468	0.00453747
Oct4	0	7.16982e-05	0.000138666
Smad1	1	0.00107949	0.00151819
Sox2	10	0.00539771	-
STAT3	10	0.0323896	-
Tcfcp2l1	6	0.00307791	0.00398977
Zfx	10	0.230923	-

The combination of Fisher's exact test and the discrimination metric boasts strong results in most datasets. Both algorithms obtain a success rate $> 50\%$ (ie. greater or equal to 6/10 successful runs) on 11 of the 13 datasets. Both algorithms also get a success

rate of 9 out of 10 runs or better for 8 of the 13 datasets, which is more than any other configuration thus far. A very interesting property of this configuration is that it actually outperforms DREME on the CTCF dataset, which is not something that was observed with other configurations. The only datasets where this configuration has a 50% failure rate of more is on Oct4 and Smad1 which, as mentioned in Section 4.2.2, do not have profiles in our motif databases.

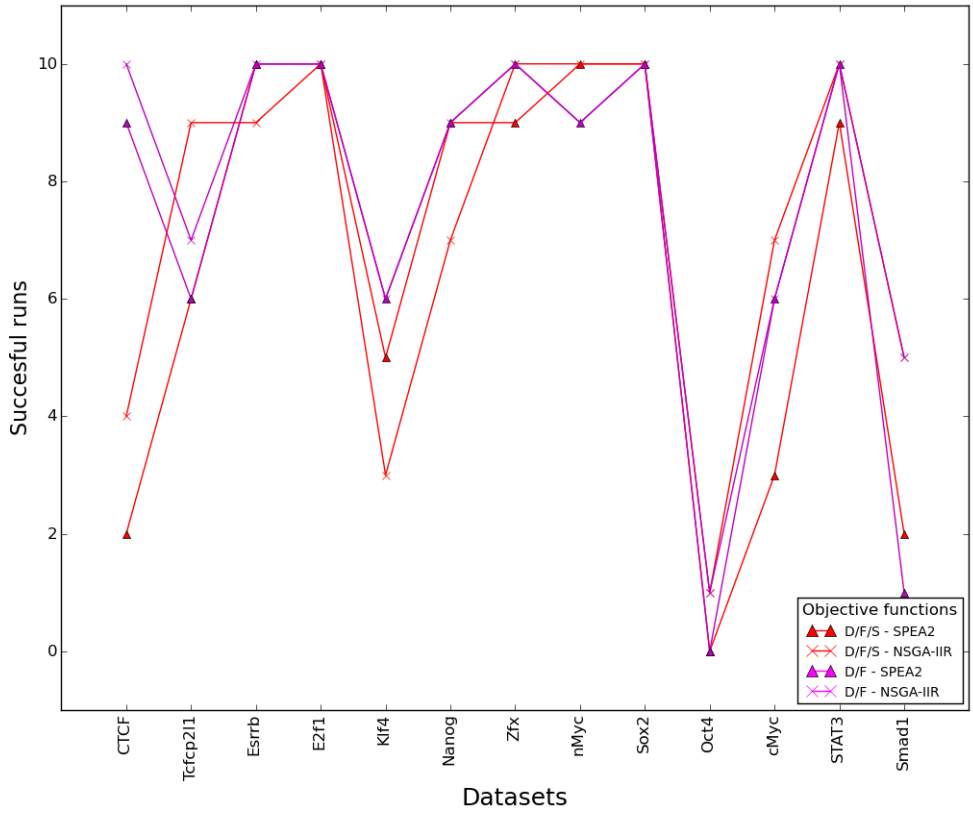


Figure 5.2: Comparison of dataset successes of Discrimination/Fisher/Support versus Discrimination/Fisher. The datasets are listed from largest to smallest in sequence count.

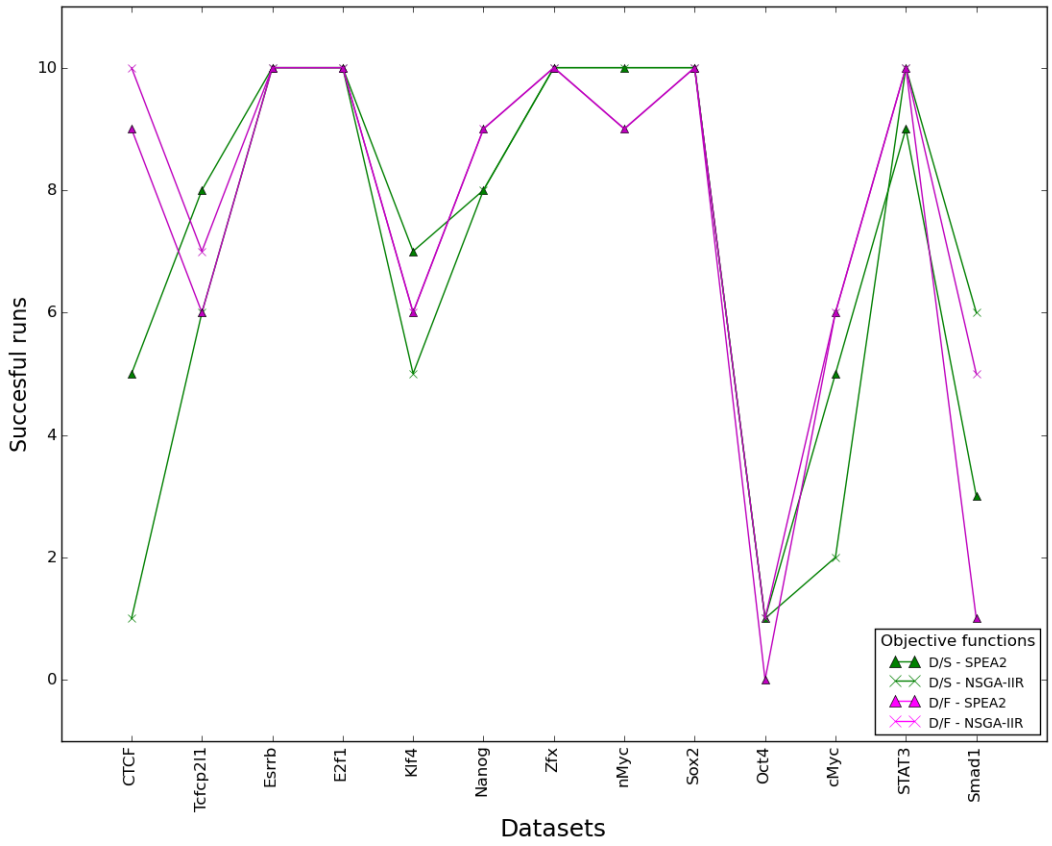


Figure 5.3: Comparison of dataset successes of Discrimination/Support versus Discrimination/Fisher. The datasets are listed from largest to smallest in sequence count.

We plotted the triple-objective success per dataset along with the Discrimination/-Fisher results (Figure 5.2). Both algorithms' runs under Discrimination/Fisher's Exact Test outperform the triple-objective in CTCF, Esrrb, Klf4, Nanog, Smad1, STAT3 and Zfx. The only instances where the triple-objective may outperform the Discrimination/-Fisher combination is nMyc, cMyc and Tcfcp2l1, and by only a margin of 1 success in each of theses datasets.

We also plotted the original Discrimination/Support method against Discrimination/-Fisher in Figure 5.3. When looking at both algorithms combined, Discrimination/Fisher outperforms Discrimination/Support in cMyc, CTCF, Nanog, STAT3, and underperforms in nMyc, and Tcfcp2l1.

Looking at individual multiobjective algorithms, the SPEA2 algorithm works best under Discrimination/Support for Klf4, nMyc, Oct4, Smad1, Tcfcp2l1 (notice when the green triangle is above the magenta triangle). SPEA2 under the Discrimination/Fisher performs best on cMyc, CTCF, Nanog, STAT3.

With the NSGA-II$_R$ algorithm, the Discrimination/Fisher method is superior over cMyc, CTCF, Klf4, Nanog, Tcfcp2l1. Discrimination/Support is only better in the nMyc

and Smad1 datasets. There seems to be an improvement over NSGA-II$_R$ runs when using Fisher's exact test instead of Support.

The main advantage of the Discrimination/Fisher method is that rarely attains poor scores on any of the datasets, with the exception of Oct4 and Smad1 datasets. It is also the only objective function able to obtain a score of 10 on the largest dataset (CTCF), let alone obtain a success threshold $> 50\%$. It also seems to have the most consistent performance between multiobjective algorithms, with the success rate differing by a margin of 1 (except on the Smad1 dataset). This feature can perhaps reduce concerns in our problem regarding the choice of multiobjective optimization algorithms, since they both perform favourably.

5.3.4 Fisher and support

The Fisher/Support objective function is by far the worst as it fails (i.e. a total of 0 successful runs) in 6 out of 13 datasets for both algorithms. Only 2 datasets yielded a success rate greater than 50% with the NSGA-II$_R$ algorithm, and only 3 for SPEA2. Plotting the results (Figure 5.4) shows that the Discrimination/Fisher variant outperforms the Fisher/Support runs in every algorithm and dataset.

Table 5.27: Comparing DREME and MotifGP using the Fisher's exact test/Support objective function and the NSGA-II$_R$ multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	0	8.59888e-05	0.000691122
CTCF	0	3.50248e-07	3.58707e-06
E2f1	8	0.0385603	0.0673758
Esrrb	0	0.000574497	0.00127559
Klf4	0	3.28351e-08	7.43606e-07
Nanog	1	0.000967177	0.00263183
nMyc	4	0.00121468	0.0019604
Oct4	0	7.16982e-05	0.0333258
Smad1	0	0.00107949	0.00263183
Sox2	3	0.00539771	0.00626948
STAT3	1	0.0323896	0.0481495
Tcfcp2l1	3	0.00307791	0.00590892
Zfx	10	0.230923	-

Table 5.28: Comparing DREME and MotifGP using the Fisher's exact test/Support objective function and the SPEA2 multiobjective algorithm. For each dataset, the #Above column represents the number of runs which reported a motif that aligned better than the best alignment for motifs reported by DREME. The "Next runner-up" shows the alignment E-value for the best reported motif amongst unsuccessful MotifGP runs.

Dataset	#Above	DREME	Next runner-up
cMyc	0	8.59888e-05	0.0019604
CTCF	0	3.50248e-07	2.34302e-05
E2f1	9	0.0385603	0.0650715
Esrrb	0	0.000574497	0.00127559
Klf4	0	3.28351e-08	7.43606e-07
Nanog	2	0.000967177	0.00179227
nMyc	6	0.00121468	0.00121468
Oct4	0	7.16982e-05	0.0290943
Smad1	0	0.00107949	0.00872448
Sox2	2	0.00539771	0.00626948
STAT3	1	0.0323896	0.0474564
Tcfcp2l1	1	0.00307791	0.0076521
Zfx	10	0.230923	-

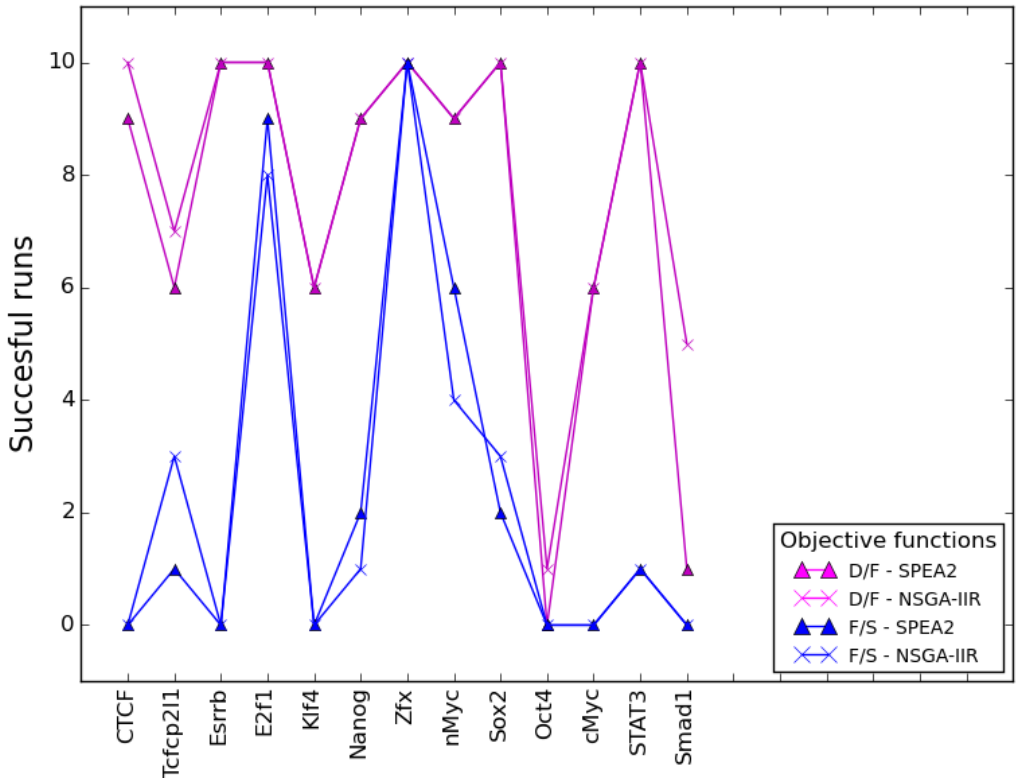


Figure 5.4: Comparison of dataset successes of Discrimination/Support versus Discrimination/Fisher/Support. The datasets are listed from largest to smallest in sequence count.

Curiously, Discrimination/Fisher seems to perform generally better than other methods, while Fisher/Support is arguably the worst. A possible reason may be because Support is too similar to Fisher's exact test, and combining them is redundant. Conversely, the solutions explored by Discrimination compared to those sought by Fisher's exact test can be very different. The lower the support, the higher (worse) the p-value from Fisher's exact test will be. Note that this is not the case of Discrimination, as a good Discrimination can have a bad p-value, and vice-versa. Consider this example where we have only one sequence matched, in the positive dataset:

Let p = 1, P = 10 000, n = 0, N = 10 000

$$\begin{aligned} Discrimination &= \frac{1}{(1+0)} = 1.0 \\ Fisher &= \frac{(1+0)!(9999+10000)!(1+9999)!(0+10000)!))))}{1!0!9999!10000!20000!} = 0.5 \end{aligned} \tag{5.1}$$

Since there are no matched control sequences, we still obtain a 100% discrimination, but due to the large sample size, Fisher's exact test returns a p-value = 0.5, which means we cannot reject the null hypothesis and is an undesirable prediction. This case illustrates

how Discrimination is not necessarily correlated with Fisher's exact test, therefore they can exploit different solutions.

5.4 Primary factor profile identification

While the previous section emphasizes the stability (given that it's evolutionary computing), we will here observe the top prediction identified across each dataset across different runs. By referring to Table 4.1, we can tell how accurate the predictions of MotifGP are with respect to an identified primary motif. The only motif considered in this section are from the Discrimination/Fisher's exact test objective function runs.

5.4.1 Primary factor motif detection

We parsed the output of the different runs and compiled (Table 5.3) a list of the best scoring motif predictions obtained on each dataset. We compared the primary targets from Table 4.1 resulting in Table 5.29.

Table 5.29: Tabulation of datasets where MotifGP motif identifies a primary target. The * denotes instances where the best aligned motif is not the primary factor, but the primary factor can still be found in the output predictions.

Dataset	**Is the Primary target found?**
CTCF	Yes
cMyc	Yes
E2f1	No
Essrb	Yes
Klf4	Yes
nMyc	Yes
Oct4	Yes*
Stat3	Yes*
Sox2	Yes
Tcfcp2l1	Yes
Zfx	Yes

Table 5.30: Tabulation of datasets where a DREME identifies a primary target motif. The * denotes instances where the best aligned motif is not the primary factor, but the primary factor can still be found in the output predictions. The double * denotes the same as the single *, but the alignment E-value of the motif was ≥ 0.5.

Dataset	Is the Primary target found?
CTCF	Yes
cMyc	Yes
E2f1	Yes**
Essrb	Yes
Klf4	Yes
nMyc	Yes
Oct4	Yes*
Stat3	Yes*
Sox2	Yes
Tcfcp2l1	Yes
Zfx	Yes

5.4.2 Dataset and transcription factor specific anomalies

Oct4 While the Oct4 dataset it one where MotifGP struggles to outperform DREME, it does identify the same primary target motifs. DREME identifies one motif which aligns equally well to two different profiles: Pou2f2, and Pou2f3 with E-value = 7.16982E-005. The motif in question is ATG[CGT][AT]AA (or ATGBWAA). MotifGP also identifies instances of Pou2f2 with motif [AGT][AT]ATGC[AT]AAT, and an E-value = 0.000180293, and Pou2f3 with motif [CGT]ATGCA[AT]AT and an E-value = 0.0220317. Although, the alignment E-value of the motif produced by MotifGP has lower E-value compared to that of DREME, both systems identify the same motif.

The confusion between Pou2f2 and Pou2f3 is likely due to the high similarity between both motif. The two UniPROBE profiles for both TFBS are in Figure 5.5 and 5.6.

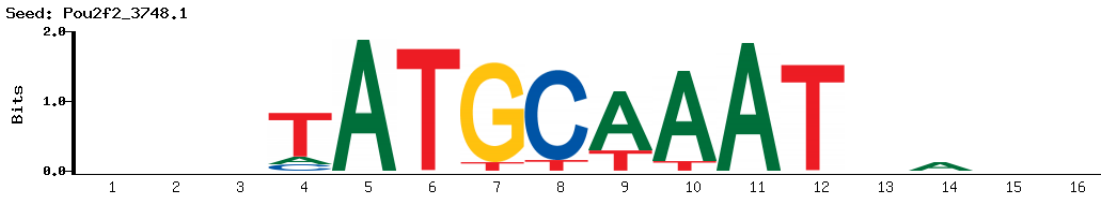


Figure 5.5: Pou2f2 binding profile from UniPROBE.

Other than some slight variation to the probability for positions with more than one nucleotide option, the main difference between the two profiles is the T nucleotide at position 1 of Pou2f3. Predictions containing any other nucleotide at this position would be likely align to Pou2f2 instead of Pou2f3. Since DREME only predicts patterns of length

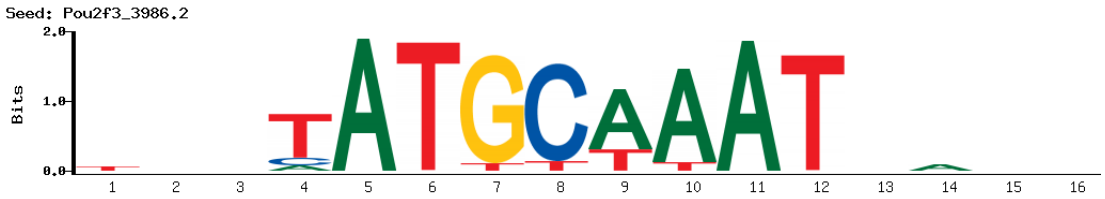


Figure 5.6: Pou2f3 binding profile from UniPROBE.

8 at most, it likely found the common pattern consensus between the two profiles that would not include the uncertain position, thus maximizing its p-value by matching more sequences. This also explains the equal alignment E-value, since the overlapped portion of the alignments is a common for both motifs.

STAT3 If we look in the top motifs identified by MotifGP (see Appendix A.2), the STAT3 motif can be found in some of the output Pareto front. DREME obtains a low significance TOMTOM alignment E-value of 1.32308 for the motif CCRGDAA (or CC[AG]G[AGT]AA in our network expression syntax). Meanwhile, MotifGP's variant scores an E-value of 1.33301E-005 for the motif TT[C]C[ACT][AG]G[AC][A] (or TTCCHRGMA in IUPAC nucleotide notation). MotifGP's STAT3 motif alignment is even more significant than DREME's best scoring prediction.

DREME and MotifGP also both identify STAT1 (MA0137.3) as their top scoring motif. It is a STAT-family member, so it is not surprising that it would be in common regulatory modules as the STAT3 TF. MotifGP's STAT1 prediction alignment is for the query motif TTCC[ACGT][AG]GAA. In a different seed, we obtained TTCC[ACT][AG]G[AC]A which actually aligns with STAT3. The STAT1 motif's objective score had a Fisher p-value of 1.824e-83 and a Discrimination of 98.6%, while the STAT3 targeting motif had a Fisher p-value of 1.978e-95 and a Discrimination of 97.2%. While *Support* was not part of this multiobjective function, we can still calculate it by counting the number of matches for the query in the dataset divided by the total number of sequences in the dataset. The STAT1 predictions occur in 262 sequences. Out of 2546 sequences in total, this gives us a Support of 10.2%. The STAT3 prediction (which has a lower alignment score) occurs in 325 out of 2546 sequences (i.e. a Support of approximately 12.7%).

This likely emphasizes the difference between a prediction and what is found in the actual dataset. Though none of the predictions are incorrect (as they appear frequently amongst the given sequences), the ratio of sequences matched (Support) can degrade as the search gets more discriminative (specific). Finally, we can also see that both motifs have very similar representations, differing by only two characters; the first Character Class at position 5 has an extra T for STAT1, and the STAT3 motif at position 8 has a Character Class $[AC]$ instead of a single nucleotide A as in STAT1.

E2f1 Curiously, none of the algorithms identifies a proper match for the E2f1 dataset. While we have a JASPAR reference for E2f1[1], the species is for Homo sapiens (humans), and not *Mus Musculus*. A high quality prediction of the Homo sapiens E2f1 in a Mus Musculus dataset would be of interest, but we could not identify such a binding site. DREME does predict a match for E2f1, but with an E-value = 4.14056 meaning it is not a confident alignment. It is unclear why both algorithms identify plenty of other motifs with higher significance than DREME's prediction of E2f1. Further evaluation of this specific dataset is required to fully understand why this happens. More information about the Mus Musculus E2F transcription factor version 1 and the genomic coordinates of the E2f1 gene can be consulted via the NCBI gene browser[2].

We can still see that the optimization provided by the multiobjective function employed by MotifGP works as intended. Table A.1 shows a summary of the best 15 TOMTOM alignment MotifGP predicted over the E2f1 dataset. We see a clear cutoff in the multiobjective fitness values: All the predicted motifs which aligned to UP00407 (Elf3 factor[3]) have very low discrimination relative to those which align to a different target. The best overall alignments that align to different Target IDs all have high discriminative scores, above the 80%. While these high-scoring alignments are relatively enriched like the Elf3 predictions, they are longer and by definition, discriminate strongly against the background. The same kind of cutoff between predictions can be observed over the prediction length: All Elf3 motif predictions are 8bp in length or shorter, while other alignments with high discrimination are of length 10 or 11.

Overall, the E2f1 dataset is challenging because it has many transcription factor binding sites that are fit for both MotifGP's and DREME's search methods. The primary factor then gets drowned out by other predictions, making it more difficult keep it as a prediction after a long evaluation period.

[1]JASPAR reference for MA0024.2 (E2f1) available at http://jaspar.genereg.net/cgi-bin/jaspar_db.pl?ID=MA0024.2&rm=present&collection=CORE

[2]E2F transcription factor 1 NCBI Genbank entry http://www.ncbi.nlm.nih.gov/gene/13555#genomic-regions-transcripts-products

[3]UniProbe reference for the Elf3 factor http://thebrain.bwh.harvard.edu/uniprobe/detailsDef.php?id=407

Table 5.3: Summary of best predictions per dataset for Fisher/Discrimination.

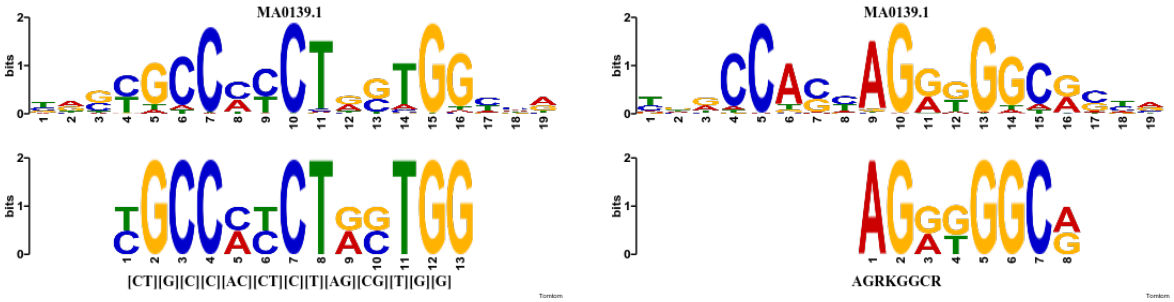
	Algorithm	Query ID	Target ID	Offset	P-Value	E-value	Overlap	
CTCF	NSGA-II$_R$	YGCCMYCTRSTGG	MA0139.1	3	9.18055E-015	5.42571E-012	13	-
Oct4	NSGA-II$_R$	VCMGGAAG	MA0473.1	3	5.68618E-008	3.36053E-005	8	+
Smad1	NSGA-II$_R$	VRACAAWGVV	MA0515.1	1	1.67564E-007	0.00009903	9	-
cMyc	SPEA2	SCACGTGG	MA0059.1	2	4.0761E-010	2.40897E-007	8	+
Zfx	SPEA2	GGCCTVGGCS	MA0146.2	2	1.97704E-008	1.16843E-005	10	-
STAT3	SPEA2	TTCCNRGAA	MA0137.3	1	7.83542E-009	4.63074E-006	9	-
Tcfcp2l1	NSGA-II$_R$	DNCCRGHNNVADCCRGHH	MA0145.2	-2	7.48357E-012	4.42279E-009	14	+
Nanog	SPEA2	RRRACAAWGV	MA0515.1	0	9.21128E-011	5.44386E-008	10	-
Esrrb	NSGA-II$_R$	TGACCTTGRVC	MA0141.2	0	6.96819E-011	4.1182E-008	11	-
Sox2	NSGA-II$_R$	RRRACAAWGGVV	MA0515.1	0	9.59515E-011	5.67073E-008	10	-
E2f1	SPEA2	BCACTTCCGS	MA0062.2	1	1.24915E-009	7.38246E-007	10	-
Klf4	NSGA-II$_R$	VGCCMCRCCCH	MA0039.2	-1	7.20229E-012	4.25656E-009	10	-
nMyc	NSGA-II$_R$	SCACGTG	MA0059.1	2	8.42227E-009	4.97756E-006	7	+

Table 5.4: Summary of best predictions per dataset for DREME.

	Algorithm	Query ID	Target ID	Offset	*p*-value	E-value	Overlap	
CTCF	DREME	AGRKGGCR	MA0139.1	8	5.92636e-10	3.50248e-07	8	+
Oct4	DREME	ATGBWAA	UP00191_1	4	1.21317e-07	7.16982e-05	7	+
Smad1	DREME	ACAAWGV	MA0442.1	0	1.82656e-06	0.00107949	6	-
cMyc	DREME	CACRTGS	UP00060_1	6	1.45497e-07	8.59888e-05	7	-
Zfx	DREME	GGCCTNB	MA0146.2	8	0.000390733	0.230923	6	+
STAT3	DREME	CCRGDAA	MA0137.3	3	5.48048e-05	0.0323896	7	+
Tcfcp2l1	DREME	ADCCRG	MA0145.2	8	5.20797e-06	0.00307791	6	+
Nanog	DREME	ACAAWRS	MA0514.1	3	1.63651e-06	0.000967177	7	-
Esrrb	DREME	RAGGTCR	MA0141.2	5	9.72077e-07	0.000574497	7	+
Sox2	DREME	ACAAWRGV	MA0143.3	1	9.13318e-06	0.00539771	7	-
E2f1	DREME	VGGAAR	UP00085_1	5	6.52459e-05	0.0385603	6	+
Klf4	DREME	GGGYGKGK	MA0039.2	1	5.55585e-11	3.28351e-08	8	+
nMyc	DREME	CACGTG	MA0104.3	2	2.05529e-06	0.00121468	6	+

Table 5.5: CTCF transcription factor binding site alignment scores.

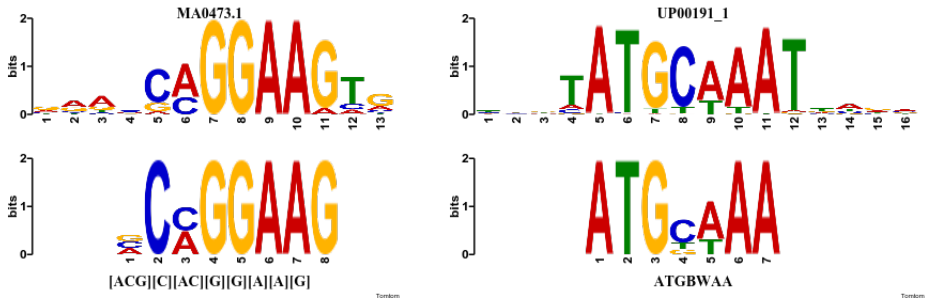
CTCF



MotifGP DREME

Table 5.6: Oct4 transcription factor binding site alignment scores.

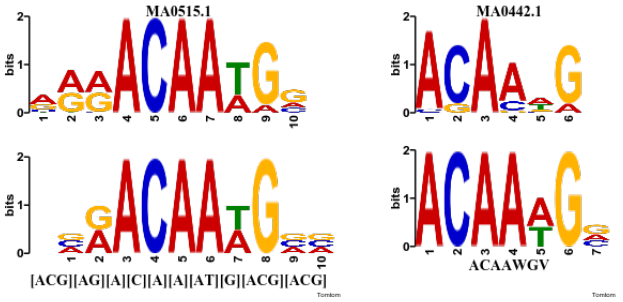
Oct4



MotifGP DREME

Table 5.7: Smad1 transcription factor binding site alignment scores.

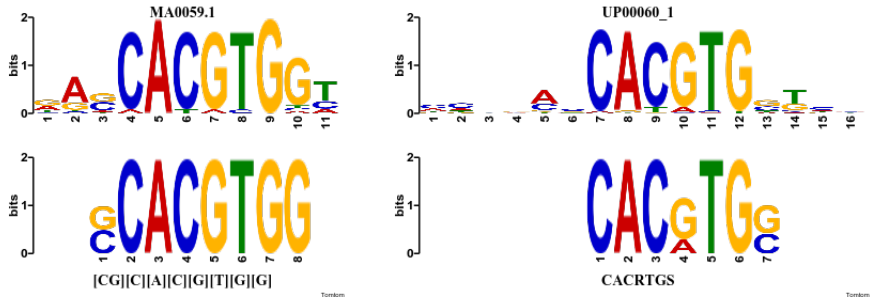
Smad1



MotifGP DREME

Table 5.8: cMyc transcription factor binding site alignment scores.

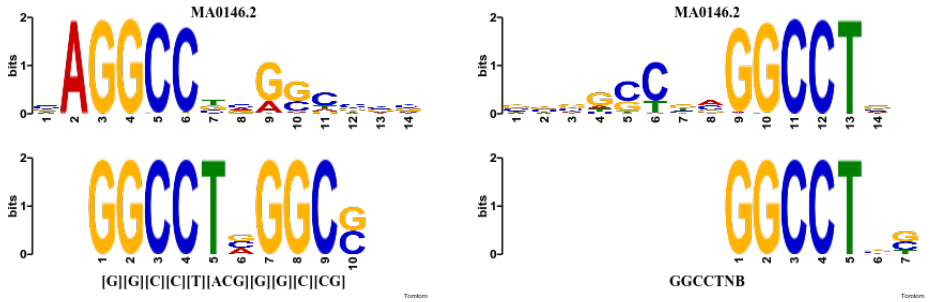
cMyc



MotifGP DREME

Table 5.9: Zfx transcription factor binding site alignment scores.

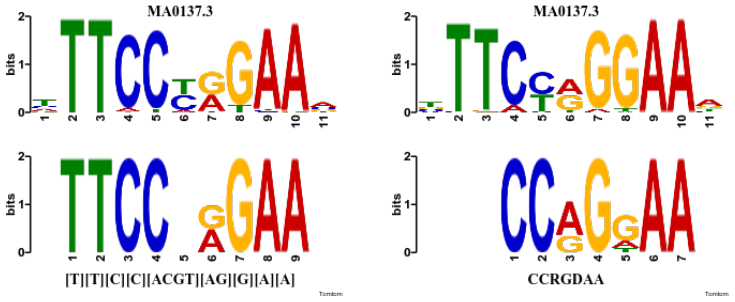
Zfx



MotifGP DREME

Table 5.10: STAT3 transcription factor binding site alignment scores.

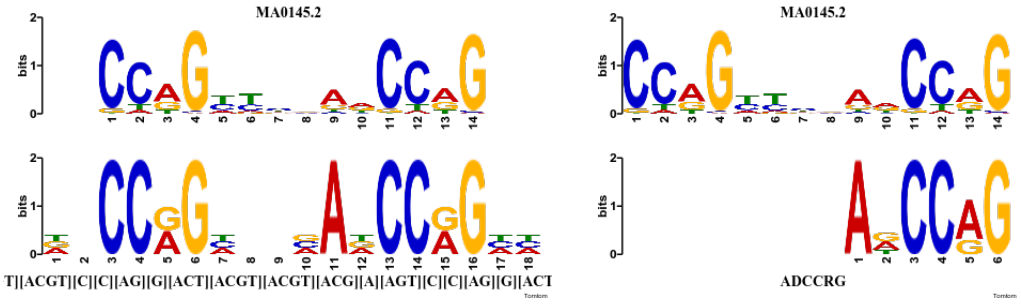
STAT3



MotifGP DREME

Table 5.11: Tcfcp2l1 transcription factor binding site alignment scores.

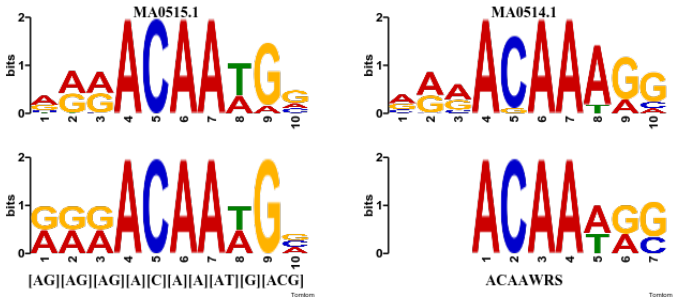
Tcfcp2l1



MotifGP DREME

Table 5.12: Nanog transcription factor binding site alignment scores.

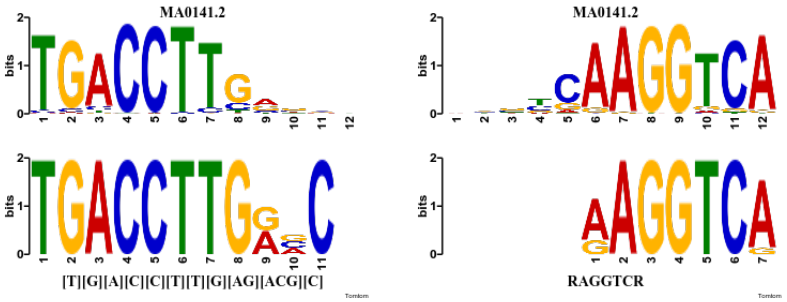
Nanog



MotifGP DREME

Table 5.13: Esrrb transcription factor binding site alignment scores.

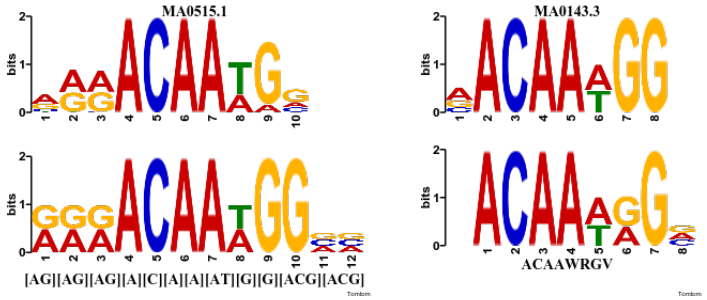
Esrrb



MotifGP DREME

Table 5.14: Sox2 transcription factor binding site alignment scores.

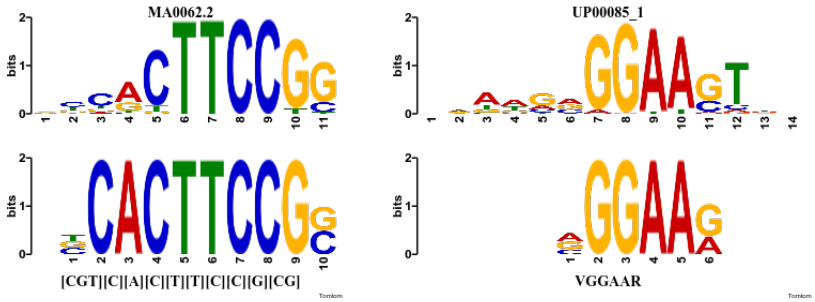
Sox2



MotifGP DREME

Table 5.15: E2f1 transcription factor binding site alignment scores.

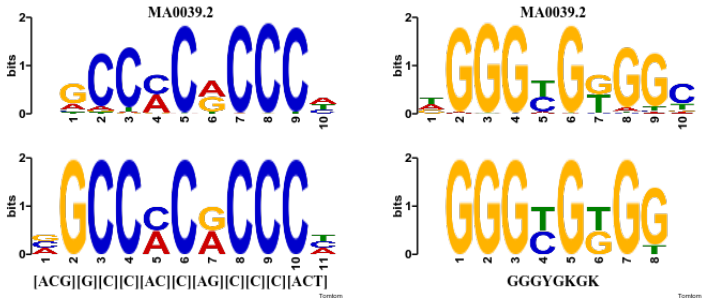
E2f1



MotifGP DREME

Table 5.16: Klf4 transcription factor binding site alignment scores.

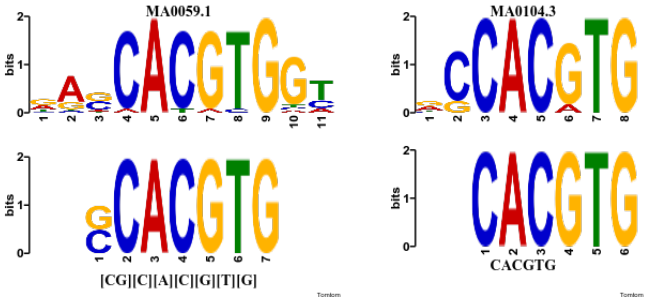
Klf4



MotifGP DREME

Table 5.17: nMyc transcription factor binding site alignment scores.

nMyc



MotifGP DREME

Chapter 6

Discussion

MotifGP offers predictions that can span an entire transcription factor binding site. Several tools impose a restriction on the length of motifs to keep the runtime under control. We did not impose such a constraint. Herein, we empirically show that an evolutionary search is powerful enough to uncover motifs spanning the entire length of a motif. Although the length of the motifs is not explicitly restricted, two of the three objective functions employed, namely Fisher's exact test and Support, force the motifs to be terse. We explain why MotifGP tends to obtain improved predictions over DREME's methodologies.

Using results from the best performing multiobjective algorithm (Discrimination and Fisher's exact test), Section 6.1 further explores the qualitative properties of MotifGP-produced predictions. We discuss a few use cases where MotifGP could offer some practical solutions depending on what is expected from a given motif discovery task, as a computational biologist would.

The next section covers and discusses the impact of fitness functions selection for ChIP-seq datasets. Specifically, we observed that Support does not seem essential to produce high-scoring prediction alignments. We are interested in why Support does not increase prediction alignment scores, and if there are any potential benefits to using this objective within our methods.

Finally, one of the goals of this research is to propose new methods to engineer biological motif discovery solutions. For example, one such consideration that was not in the initial design is the ability to match the *reverse complement* of motifs during the pattern matching process. In Section 6.3, we reproduced our experimental runs with an implementation of MotifGP using reverse complement matching in order to see if an evolutionary motif discovery algorithm is necessarily enhanced by this. These sections not only tests the result of this extension over ChIP-seq datasets, but also shows the flexibility of the MotifGP engine by customizing additional biological constraints in the evolutionary mining process.

6.1 Predictive capabilities of MotifGP

Evolutionary algorithms have been shown to be difficult to evaluate in formal contexts because solution convergence is highly dependent on the problem [69]. Instead, our experimental design was focused on the quality of results obtained via post-analysis, and not simply by our own standards such as solution fitness. Our primary goal is to offer an analogous solution to DREME. For these reasons, the resulting MotifGP predictions are difficult to bound; we are not guaranteed the same results on two differently initialized runs. It is, however, possible to evaluate if our current implementation can offer superior alignments to DREME as a relative benchmark.

Indeed, our primary factor identification tabulation (Section 5.4) shows that there is a strong overlap between what can be identified by DREME and MotifGP. Both algorithms find common primary target motifs, and both algorithms are consistent on irregular predictions, such as multiple identification of STAT-family members (as described by the anomalies in Section 5.4.2).

If we consider consistency across different runs, our Discrimination/Fisher's exact test combination (Section 5.3.3) shows very competitive results compared to DREME. The algorithm can generally be expected to produce better predictions than DREME, and the only datasets where this is not the case are for motif datasets that are not well represented in the JASPAR or UniPROBE databases.

By looking at the type of predictions and output format MotifGP produces, we can narrow down potential niche use cases a computational biologist could require:

- If we are looking for specific transcription factor binding site in a large dataset of ChIP-seq, MotifGP should be able to identify its presence. This is the case for all experiments where we have a known TFBS in the database (Table 5.29), except E2f1, which DREME also does not yield a prediction aligned with a significant E-value.
- If a strongly represented transcription factor binding site exists in a dataset, but other subtle patterns are also of interests, then they can be explored by running multiple initializations of MotifGP.
- Transcription factor binding sites that are longer than 8 bp can be completely covered by predictions from MotifGP, whereas DREME cannot (for example, see Figure 5.10 for the STAT3 experiment).
- Extending the runtime for MotifGP can improve and refine a prior discriminative motif discovery search, while DREME already outputs its best possible predictions once it reaches a certain stage in its algorithm.
- MotifGP can uncover motifs where the specificity of the motif is distributed along the length of the motif, whereas other approaches assume that a motif consist of a conserved core (contiguous conserved positions). In contrast, DREME will not expand on predictions if the p-value does not improve, and usually predicts only up to 8 bp, which does not leave a lot of room for predicting gaps within a larger motif.

The motif length is especially noteworthy consideration when it comes to using MotifGP over DREME. While DREME's best predictions are typically limited to short core fragments (8 bp) of the whole binding site, they still obtain a significant alignment score. Consider the CTCF dataset, where the best DREME motif spans from positions 9 to 16, with an E-value of approximately 3.5E-07. MotifGP's top prediction spans positions 4 to 16 (an overlap of 13 positions), and reaches an E-value as low as 5.40E-12. Predictions that have more overlap have fewer chances of being a false positive, since we have more information to align against known profiles. This can feasibly become an important aspect of motif validation as more motif variants becomes available, and similar family members become harder to distinguish. Our experiment does show that it is favourable to have long, thoroughly covered binding site predictions, as this enhances the confidence of our predictions.

There are also cases where 8 bp may not be sufficient. As we observed with the Tcfcp2l1 motif, DREME fails to capture most of the profile. It covers positions 9 to 14, which is only 6 characters. While the covered portion is strongly conserved, it is no surprise DREME fails to capture nucleotide information between positions 5 and 8. The reasoning is that since this portion of the profile is very lightly conserved, DREME fails to iteratively expand regular expressions into to this area of the TFBS. DREME uses Fisher's exact test to compare potential predictions and refines them; it is likely that adding the poorly conserved portion to the predicted regular expression would only hinder the resulting p-value, so the algorithm halts before reaching the other extremity of the motif.

MotifGP does not explore the motif predictions on the basis of its length. The two distinct, conserved, portions of the Tcfcp2l1 motif can be evolved as population candidates until they either mutate the remaining part of the profile, of crossover with their missing adjacent portion. MotifGP obtains stable prediction scores (over 10 seeded runs of the Tcfcp2l1 datasets), though not as stable as other datasets with TFBS that do not contain such a weakly conserved centre. The interior portion of the motif likely requires more runtime before the proper crossover occurs since it is guided by a randomized selection process. In that light, it is not surprising to see a few unsuccessful runs from MotifGP where the top-scoring motif falls within a few orders of magnitude from DREME. Given more runtime, better predictions would surely be produced in those unsuccessful runs.

6.2 Fitness function impact over ChIP-seq dataset predictions

By comparing the performance of different combinations of multiobjective fitness functions, it becomes apparent that the choice of function is crucial to ensure a certain level of success under restricted runtime. The combination of fitness functions that did not include Discrimination was the worst; while those that included Discrimination all performed favourably.

The performance of the Discrimination/Fisher's exact test combination also brings to question the relevance of Support for discriminative motif discovery. It is worth asking

whether this function, despite its popular use in multiobjective motif discovery algorithms, is relevant to real ChIP-seq datasets where no one knows what the support of the sought motifs should be; this number is dependent on the peaks resulting from the sequencing process at large. An irrelevant peak (one that would not contain any known TFBS common to other peaks) can be the caused by unknown consequences or errors in sequencing, and will impact a Support score. Discrimination would not be affected, and Fisher's exact test would be less severely penalized than Support. Furthermore, any peak that contains a co-factor, a reverse complement or an unknown variant of the sought TFBS are relevant to the dataset, but still penalize the Support.

Moreover, Support will favour general motifs that may not represent the full-binding site, as long as they cover many sequences in the dataset. While Discrimination seems to correct this in multiobjective evolution, Fisher's exact test does not seem to provide a strong enough discriminative penalty on the control dataset for it to be a useful pair with Support. Discrimination does pair well with Support, and even better with Fisher's exact test, suggesting that Support and Fisher's exact test has similar effect in their purpose as objective functions.

Support has been shown to be useful for other datasets. It performed favourably when combined with *Similarity* and *Length* functions [31, 30] over datasets from a motif discovery benchmark collection [72]. While we briefly evaluated the datasets in question, we found them to be incompatible with our methods. Our methods were developed with the mouse embryonic stem cell ChIP-seq datasets [17] in mind, which at the very least contains a thousand sequence per dataset. The benchmarks from [72] typically have fewer than 10 sequences per dataset, and recent work [30] filter to maintain only predictions with a Support of 50% or more, which is not something we designed for.

Our methods also strongly rely on the control sequences. Both Discrimination and Fisher's exact test consider matches on the control sequences in their calculation. For example, if we were to use the same dataset for input and control, predictions would always have a Discrimination of 50% and Fisher's exact test would return $p = 0.05$ as its p-value. Our results show that the dinucleotide shuffling function [18] provides meaningful background for accurate predictions, which does not require any input from the user to produce a background for control.

6.3 Zfx, the Reverse Complement, and Discriminative Search

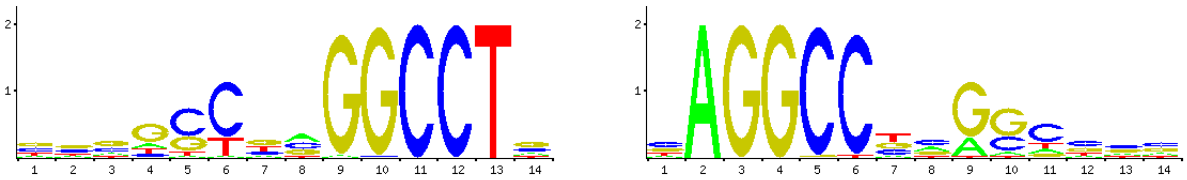
A particular feature in our result that drew our attention is how the Zfx predictions from MotifGP and DREME map to different strands (DNA orientations) for its motifs. DNA is structured as a double-helix, two sequences of nucleotides that are complementary. At the position where we have an *adenine* (A), the same position (though relatively reversed) typically has the complement nucleotide, *thymine* (T). Table 6.1 lists the complement value for each base.

Following that table, a reverse complement of a DNA sequence can be computed by:

Table 6.1: Complement table for nucleic acids.

Base	Complement
A	T
C	G
G	C
T	A

Table 6.2: The Zfx binding site (left) and its reverse complement (right) from JASPAR.



- Rewriting the sequence from end to beginning, essentially reversing its order.
- Changing each occurrence of a *Base* to its *Complement* (See Table 6.1).

With that in mind, a query to the motif database that aligns to the reverse strand of another means their alignment logo is the reverse complement of one another. One would expect the prediction that aligns to different strands to be also complement. They are not in our results; they actually share a common core aligned at different positions. This can be observed in Figure 5.9, when comparing DREME and MotifGP's alignments.

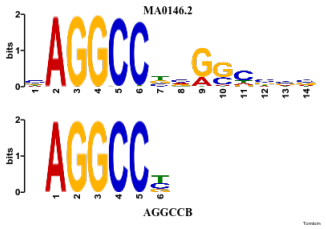
Consider the alignment logo for both strands of the Zfx binding site from JASPAR [48]: Both profiles have a strongly conserved **GGCC** portion (see Table 6.2), since its reverse complement is also **GGCC**. Furthermore, both motif could align with **[AT]GGCC[AT]**, which is an expansion on the previous core. We have here a particular case where a prediction can align well for both a certain binding site and its reverse complement, but at different positions. This is relevant to DREME, because by default, the tool searches regular expression predictions as well as their reverse complement, and DREME's search process extends regular expressions until they stop increasing in significance (from Fisher's exact test). Ergo, any sequence that included the reverse complement motif instead was not matched prior to this extension. Adding this feature can only increase the overall number of matched sequences per motif.

To disable search on the reverse complement, DREME has a *-norc* parameter which according to the documentation will: "search the given strand only for motifs (not reverse complement)". We ran DREME with the *-norc* parameter, and while the runtime is reduced, so is the prediction significance. Table 6.3 shows the prediction scores of the best DREME *-norc* Zfx prediction.

While the *–norc* switch generally downgrades the results, the runtime for the Zfx dataset is more than halved (compared to reported times from Table 4.2) and the predictions obtain better alignment scores than our previous DREME benchmark. This shows an instance

Table 6.3: Top TOMTOM alignment (AGGCC[CGT]) for DREME predictions on the Zfx dataset when disabling reverse complement search.

Target	MA0146.2 (Zfx)
E-value	0.00536535
Overlap	6
Orientation	- (Reverse)
DREME runtime (seconds)	547

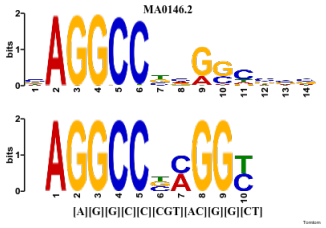


where DREME's prediction not improved by allowing search of their reverse complement. Fisher's exact test will yield a more significant p-value when provided a higher number of matches on the input sequences. Thus disabling reverse complement search can directly affect the p-value of predictions, therefore which motifs should be expanded in the search process.

Comparing results between each configuration tells us where DREME draws the line for its top predictions depending on reverse-complement search. We also implemented a variant of MotifGP's evaluation function where the compiled solution would also generate its reverse complement before being matched to input sequences. Any sequences that matched the given motif or its reverse complement would be counted as a matched sequence in the objective function.

Table 6.4: Top TOMTOM alignment (AGGCC[CGT][CA]GG[TC]) for MotifGP predictions on the Zfx dataset when enabling reverse complement search.

Target	MA0146.2 (Zfx)
E-value	1.33394e-05
Overlap	10
Orientation	- (Reverse)
DREME runtime (seconds)	1258



The top prediction from MotifGP when also considering the reverse complement is not better than the reported motif with the default settings (Table 6.4). Searching for a network expression and its reverse complement requires more time: Under a fixed runtime, MotifGP have less time to search the solution space. This suggests MotifGP should not necessarily search the reverse complement when using the *(Discrimination/Fisher's exact test)* objective function. This can also be credited to the Discrimination function which could be favoured by not using the reverse complement search. Since there are fewer possible matches, there is a lesser chance for false positives, which is the only aspect of a motif that will negatively impact discrimination.

Our next step in this analysis is to check how many times each pattern occurs. We searched the dataset for patterns of interest to narrow down how each algorithm draws its predictions. Table 6.5 breaks down the number of matches for a few given *substrings* of motifs and some end result predictions. The first pair of patterns we checked where

Table 6.5: Zfx sequence match counts using various patterns and its reverse complement. The maximum number of matches for a given pattern is the number of sequences in the dataset (10338).

Query	Matches	
GGCC	8155	The shared core.
CCGG	4668	The shared core (non-reverse, complemented)
[AT]GGCC[AT]	3466	Extended shared core.
GGCCT	4550	Extended core (normal orientation)
AGGCC	4581	Extended core (reverse orientation)
AGGCC[CGT]	4082	DREME (-norc) top prediction alignment.
GGCCT[ACGT][CGT]	3935	DREME (default) top prediction alignment.
AGGCC[CGT][CA]GG[TC]	200	MotifGP's (–revcomp) top prediction alignment.
GGCCT[ACG]GGC[CG]	197	MotifGP's (default) top prediction alignment.

the shared core (the motif fragment from the Zfx profile that is common in the normal orientation and its reverse complement) and its complemented version. *GGCC* is much more frequent than its reverse complement, even though both representations can align to both representation of TFBS. While the complement fits the alignment logo, it does so at weakly conserved core of *CCGG*, so it can be reasoned that this core is not as frequent as *GGCC* due to the weak representation at matching positions of the pattern. This also explains why predictions would extend this version and not the complement. This can be seen in the last two pairs of patterns from Table 6.5: All top predictions from DREME or MotifGP are extensions of this core.

Searching for the reverse complement has more effect after this core is extended. Next, the core is padded on its extremities to match a more discriminative pattern (via p-value of Fisher's exact test). Maintaining matches for both versions of the profile (Table 6.2) returns fewer matches (3466 matches for [AT]GGCC[AT], versus GGCCT or AGGCC which match 4550 and 4581 sequences respectively). At this point, DREME must refine the patterns to align to one orientation, not both.

In the *-norc* variant (6th row in the table), the top prediction extends AGGCC, as it is more prominent. Note that if *-norc* was not activated, the prediction GGCCT could also search for AGGCC, as they are complement. This can be seen in the next row, where the motif prediction for the default DREME settings contains that motif. Finally, we noted that the configurations order which obtained the best TOMTOM alignments (MotifGP (default), MotifGP (*–revcomp*), and DREME (*-norc*)) have an inverse relation to the number of sequences matched. This furthers the hypothesis that overall sequences matched (*Support*) is not necessarily indicative of alignment confidence.

Table 6.6: Summary 10 independent runs of MotifGP predictions benchmarked against the top scoring DREME TOMTOM alignment, listed for each 13 datasets using Discrimination/Fisher's exact test, the NSGA-II$_R$ multiobjective optimization algorithm and the –revcomp switch.

Dataset	#Above	DREME	Next runner-up
cMyc	0	8.59888e-05	0.0019604
CTCF	0	3.50248e-07	2.34302e-05
E2f1	9	0.0385603	0.0650715
Esrrb	0	0.000574497	0.00127559
Klf4	0	3.28351e-08	7.43606e-07
Nanog	2	0.000967177	0.00179227
nMyc	6	0.00121468	0.00121468
Oct4	0	7.16982e-05	0.0290943
Smad1	0	0.00107949	0.00872448
Sox2	2	0.00539771	0.00626948
STAT3	1	0.0323896	0.0474564
Tcfcp2l1	1	0.00307791	0.0076521
Zfx	10	0.230923	-

Table 6.7: Summary 10 independent runs of MotifGP predictions benchmarked against the top scoring DREME TOMTOM alignment, listed for each 13 datasets using Discrimination/Fisher's exact test, the SPEA2 multiobjective optimization algorithm and the –revcomp switch.

Dataset	#Above	DREME	Next runner-up
cMyc	5	8.59888e-05	0.000106361
CTCF	7	3.50248e-07	4.80112e-07
E2f1	10	0.0385603	-
Esrrb	10	0.000574497	-
Klf4	6	3.28351e-08	2.57034e-07
Nanog	10	0.000967177	-
nMyc	10	0.00121468	-
Oct4	1	7.16982e-05	0.000180293
Smad1	3	0.00107949	0.00412219
Sox2	10	0.00539771	-
STAT3	10	0.0323896	-
Tcfcp2l1	4	0.00307791	0.00433643
Zfx	9	0.230923	0.283353

We listed the 10 seeded runs' results with *–revcomp* evaluation as in the previous section for comparison (Table 6.7 and 6.6). While the results are similar to the default settings (with the exception of CTCF which turned out worse under both SPEA2 and NSGA-II$_R$), we cannot fully compare the performance of this method at this point. This would require a separate analysis comparing both method's results (instead of using DREME as a benchmark), and it would be favourable to compare them under generations to make sure

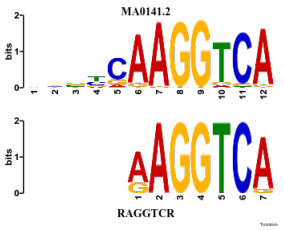
runtime saving mechanisms (such as memoization) does not favour a certain method. Still, these results can be taken as an indication that, in MotifGP's case, searching the reverse complement does not severely impact the prediction stability (under restricted runtime).

Essrb's complemented predictions

Similarly to the Zfx experiments, we also noted MotifGP and DREME predict different motifs of orientations on the Essrb dataset under their default settings. In this case, the DREME *-norc* top prediction still maps to Essrb (normal orientation), with comparable alignment scores (Table 6.8).

Table 6.8: Top TOMTOM alignment (RAGGTCR) for DREME predictions on the Zfx dataset when disabling reverse complement search.

Target	MA0141.2 (Zfx)
E-value	0.000490087
Overlap	7
Orientation	+ (Normal)
DREME runtime (seconds)	1693



For MotifGP, we obtained the same exact top prediction whether we used *–revcomp* search or the default setting.

From this, we gain further insight on how crucial reverse complement search is too good, highly significant predictions. It has strong implication over Support and Fisher's exact test. Again, it does not seem to be necessary to search for complements with MotifGP, whereas DREME may or may not perform better with it. This time, with Essrb, the orientation does not change depending on whether or not the reverse complement is searched for, possibly because the sought TFBS does not contain an equivalent reverse complement core as we had with Zfx (remember **GGCC**). This shows that it is possible for MotifGP and DREME to predict complementary results, simply because they differ in search approach, and not explicitly because of the reverse complement search.

Chapter 7

Conclusion and future work

Over the course of this manuscript, we have described the design goals and considerations of an extensible motif discovery engine built on strongly type genetic programming. Considering the state of the system and the results we were able to evaluate, we can attribute a few achievements to MotifGP:

- MotifGP is capable of a powerful discriminative search which predicts patterns that are relatively overrepresented versus a controlled or randomized background set of sequences. This discrimination also implicitly allows predictions to become longer and more detailed throughout evolution, which is desirable.
- Multiobjective optimization is a driver of solution diversity: High-scoring solutions in a certain objective typically identify different binding sites that other solutions in the same Pareto front who scored higher in another objective.
- MotifGP is capable of predicting discriminative motifs on real ChIP-seq datasets. The produced transcription factor binding sites alignments have potential for a higher significance than those of the competing tool when compared database references such as JASPAR/UniProbe.
- While DREME was developed to provide a time efficient solution to the motif discovery problem at the cost of shorter length predictions, MotifGP predicts binding sites of unrestricted lengths under the same runtimes, which mostly score higher alignments than those of DREME. Since longer motifs are inherently less frequent than short general predictions, DREME will score short motifs higher due to their number of input matches (as illustrated in Table 6.5).

After assessing the performance of MotifGP in tackling the motif discovery problem, we are confident the engine could be used to help understand new datasets, even in its current state. Conversely, we also put a great deal of effort in keeping the solutions modular, generic and extensible. Using the ground work established by MotifGP, there are many directions that could be explored for further development of this software, and we hope to see some of these questions addressed in the near future.

Modular matching

Evaluating candidate solutions is undoubtedly expensive. MotifGP will try to match a new candidate that needs a fitness by matching against each input and shuffled (control) sequence. In the worst case, we have N individuals matched against S sequences twice. We already took initiatives to minimize the runtime spent under matching using caching (memoization), but a more elaborate matching strategy could be developed to avoid searching the entire dataset twice for each new individual that is not cached.

In earlier stages of development, the matching component [5] was intended to use a trie structure or a compressed text index. The conceptual idea is that the datasets could be stored in their own trie or index, such that once the structure is populated with the input sequence, we could match substrings linearly. Such a structure would allow faster evaluation of population fitness, thus less runtime required for convergence towards optimal solutions.

Modeling individuals and predictions as PSPMs

TOMTOM evaluates PSSM for pattern matching. MotifGP provides such matrices by converting its output network expressions converted into PSSM with equal probability distribution for nucleotides (0.25 each). This current method has the flexibility that comes with generating network expressions (as opposed PSSMs, which are more elaborate and complicated) and test whether a candidate solutions matches any given sequences in the dataset. An alternative method would be to evolve candidates as PSPM. A PSPM is another type of matrix with probabilities $\in [0, 1]$. The .MEME files used for computing the TOMTOM alignment is a such a matrix, except the file also contains a background distribution probability for the nucleotide, which is used evaluate predictions as a PSSM.

Since we already convert network expressions to a PSPM, it would be of interest to see how candidate evolution is affected when the GP optimizes matrices with probabilities rather than a network expression. While the solution search space with undoubtedly be larger, as the GP will mutate and crossover on probabilities as well as nucleotides order/position, it would add the ability to express patterns that are not possible with a network expression (e.g. a position probability [0.7, 0.15, 0.10, 0.05] for A,C,G,T). The matching mechanism would have to be changed to accommodate probabilities, rather than an exact string as a network expression does, and the fitness resulting from a match would likely be closer to a value of significance (p-value) instead of using a count of sequence matched. This approach might be beneficial if it shows to be able to uncover patterns with low probability distributions for certain positions, in contrast to a network expression which assumes contiguous nucleotides.

We already have implemented the necessary components, such as a strongly typed grammar producing PWM in turn converted to PSPMs, and integration with a fast PWM matcher [39]. There still needs to be more work done in adapting and creating new fitness functions that will work as well as the ones we have for the network expression matching. Since Discrimination and Fisher's exact test rely on sequence counts, it is not yet clear how

p-values scores from PSSM matching over sequences will affect the multiobjective fitness function. Alternatively, the probabilities could be extracted from the matched sequences. Recent works also suggest methods to infer binding energy by using a scaling parameter between PWM values [43].

Refining a Pareto front into a consensus PSSM

When MotifGP terminates its execution, it returns a set of solutions as predictions. These predictions are ranked as a non-dominated front of Pareto-optimal solutions. It can be desirable to look at the ranking and pick specific solutions for post-analysis based on their fitness scores or appearance. The most standard evaluation method we used was to pass the Pareto front individuals to TOMTOM and to then combine the alignment results with the output.

Predictions could be refined by making use of multiple solutions in the front to cluster similar solutions. This would require Pareto front solutions to be sampled into a common PWM and then converted to a PSPM for TOMTOM analysis. The challenge with this approach is finding how to optimally cluster predictions, and which ones should be clustered if not all.

Another approach could make use of regular expression groups. Essentially, each token for a nucleotide in the regular expression should make use of the *group* modifier. From example, the expression ACA would become $((A)|(.))((C)|(.))((A)|(.))$. Using groups on each regex elements would allow us to know what are the exact tokens are matched in a expression and count individual matches. This would provide a practical way to build a PWM consensus representing multiple predictions.

Sequential covering of multiple datasets

When we study the output Pareto front of a given MotifGP run, we find motifs that align to different transcription factor binding sites at various degrees of affinity. Predictions which score favourably in Fisher's exact test or Support are generally widely represented in the dataset despite sometimes having a weaker alignment due to lack of specificity, similar to what DREME provides. Predictions who scored high Discrimination values are typically the opposite, as a fully developed, highly discriminative motifs will be rarer than a shorter, more commonly found prediction.

The strength of MotifGP comes from its ability to balance and refine these different characteristics of motifs in a shared population. However, we have not explored what would happen if the top predictions on a given run would be excluded from a subsequent run. For example, let the Pareto-optimal front of motif predictions be a set of rules that are learned in the first pass of a two-step search process. These rules are what classifies the first set of Pareto-optimal motif predictions. In the second step, we would re-execute the same run with the same parameters, but this time excluding any motifs that are supersets of the prior Pareto-optimal solutions, since these would fit the set of previously learned rules. By

eliminating solutions that classify under the previous rules, we obtain a new classification of motifs that could not be identified otherwise. In data mining, this is called *Sequential Covering*. We believe new motifs on previously trialled datasets could be uncovered by this method, and additional subsequent runs could also reveal even more hidden motifs.

We implemented a basic version of sequential covering that requires an output *.nef* file and uses its motif to blank matches on the input dataset. Any nucleotide that match an existing motif would be changed to N nucleotides, such that the next run could not predict previously suggested motifs.

While our results do not suggest this kind of pre/post processing is required in any way to obtain high-scoring predictions, it could become useful when dealing with subtle differences between sequences and datasets. In early stages of MotifGP, datasets for the TAL1 transcription [53] factor were studied. We analyzed two datasets: A set of peaks from a jurkat cells known to express the TAL1 transcriptional network, contrasted to peaks of erythroid cells in which TAL1 is known to promote T-cell differentiation. The two datasets are compared as a way to better our understanding of the role of TAL1 in different genetic functions and environments.

Using sequential covering, we ran some preliminary tests on the TAL1 datasets. We ran two runs for the first phase of the experiment. The erythroid peaks versus the jurkat peaks, and vice-versa. For the second phase, we reran the same experiments as the first pass, but this time, the Erythroid predictions would be deleted in the second phase of the Jurkat versus Erythroid run, and vice-versa. The hypothesis is that this will provide two tiers of discriminative search. First one being the use of a control dataset, and the second tier by using optimal predictions from the control sequences in order to erase potential motifs in the input sequences.

Erythroid with erased Jurkat predictions

Table 7.1: Phase 1: Jurkat versus Erythroid.

NE	**Discrimination**	**Fisher**	**Target ID**	**E-value**
DCAGGAARY	0.576	2.416e-51	MA0156.1	0.000520097
WCAGGAARY	0.600	1.596e-37	MA0156.1	0.000522498
DCCGGAARY	0.671	1.199e-11	MA0076.2	0.00139116
WCCGGAARY	0.702	2.449e-9	UP00408_1	0.00546214
WCAGGAAGY	0.623	3.577e-25	MA0474.1	0.0059881
AGGAARY	0.455	1.855e-65	MA0156.1	0.00832922
CGTSKTKAA	0.728	1.811e-5	UP00084_2	0.0798516
CGTSDTKAAA	1	3.932e-5	UP00050_2	1.14247

Table 7.2: Phase 2: Erythroid versus Jurkat with motifs from Table 7.1 erased in the dataset.

NE	Discrimination	Fisher	Target ID	E-value
SGSCGGG	0.844	4.19E-011	MA0470.1	0.0205395
SGSCGSG	0.842	6.23E-012	MA0470.1	0.038739
SGCCGGG	0.881	2.67E-008	UP00009_2	0.0785293
GGRCGCGG	1	9.07E-006	MA0079.3	0.127282
SGCCGGS	0.858	6.04E-009	UP00009_2	0.208287
GKRCGCGG	0.978	6.11E-006	UP00007_1	0.25928
GGRCGCSG	0.967	6.61E-007	MA0516.1	0.325864
GKRCGCSG	0.948	2.88E-007	MA0162.2	0.755771

In Table 7.1, we have the standard Jurkat versus Erythroid predictions, while Table 7.4 shows what happens to Jurkat predictions after deleting high-scoring Erythroid predictions which we want to discriminate against. In this case, we see the lesser alignment scores after deletion, but what is interesting is that we see new motifs being uncovered. Indeed, none of the motif from the basic Jurkat versus Erythroid run are present after deleting Erythroid predictions, and relevant new motifs are also discovered. We find RUNX1 (MA0002.2) within the Jurkat dataset with an alignment E-value $\doteq 0.009$. This is a relevant prediction, as TAL1 is shown to interact with RUNX1 [53], and it was not discovered on its initial pass. However, the prediction only holds a Discrimination of 60%. This suggests that the motif is highly represented in both datasets, and may not have been identifiable by our prior methods without sequential covering. It is also worth mentioning that this second pass also yields 11 different predictions that have E-value < 0.05, while the first pass only has 3.

Jurkat with erased Erythroid predictions

Table 7.3: Phase 1: Erythroid versus Jurkat.

NE	Discrimination	Fisher	Target ID	E-value
CMCRCCC	0.799	2.65E-014	MA0039.2	2.83E-007
GSGCRCCCC	1	1.26E-005	UP00088_1	0.00802496
SSRCRCCCCD	0.936	6.83E-010	UP00043_2	0.015994
GSGCRCCCS	0.981	4.96E-007	UP00035_2	0.0227087
VSACRCCC	0.823	1.71E-011	MA0493.1	0.026703
SSRCRCCCSD	0.906	3.48E-010	UP00043_2	0.0412463
SSRCACCCCR	0.969	1.97E-007	UP00024_1	0.0595984
BSRCRCCCVD	0.846	3.08E-011	UP00043_2	0.0841413
SSRCRCCCCR	0.963	8.54E-009	UP00024_1	0.084703
SSRCRCCCVD	0.862	7.18E-011	UP00043_2	0.0931838
VSRCRCCC	0.811	9.83E-013	MA0493.1	0.113739
SSRCACCCVD	0.884	1.49E-010	MA0493.1	0.272815

Table 7.4: Phase 2: Jurkat versus Erythroid with motifs from Table 7.3 erased in the input.

NE	Discrimination	Fisher	Target ID	E-value
NMNWAACCAC	0.644	1.303e-50	UP00017_1	0.00106954
NNVNWAACCAC	0.62	4.181e-66	UP00228_1	0.00187369
VAAACCAC	0.628	5.073e-52	MA0511.1	0.00621312
NWAACCAC	0.6	4.926e-77	MA0002.2	0.00924382
NNVNWAACCRC	0.603	7.494e-69	UP00228_1	0.0125815
HAACCAC	0.56	2.650e-83	MA0511.1	0.0185491
VVDWAACCAC	0.663	1.441e-44	UP00228_1	0.0265504
NVVNAAACCAC	0.674	1.288e-43	MA0002.2	0.0304205
NNHNWAACCRC	0.604	5.112e-67	UP00017_1	0.0321825
CVNHAACCAC	0.696	5.220e-32	MA0073.1	0.03247
NWAACCRC	0.587	1.353e-82	MA0002.2	0.0462323
MVNAAACCAC	0.694	1.638e-35	MA0002.2	0.05788
NHAACCRC	0.553	2.787e-92	MA0002.2	0.089628
HAACCRC	0.553	2.047e-92	MA0002.2	0.089628
SVNWAACCAC	0.69	1.155e-38	UP00228_1	0.115167
CVNWAACCAC	0.736	3.211e-26	MA0073.1	0.137178
SVDWAACCAC	0.702	3.399e-30	MA0002.2	0.139919
CVDAAACCAC	0.813	1.169e-18	MA0002.2	0.149017
CVDWAACCAC	0.784	1.100e-22	MA0002.2	0.16174
CVWAAACCAC	0.833	8.013e-12	MA0002.2	0.179096
NAACCRC	0.527	1.869e-92	MA0002.2	0.265944
CSRAAACCAC	0.909	2.543e-5	UP00040_1	0.282865
CDDWAACCRCAC	1.0	1.399e-4	MA0002.2	0.288191
CDNWAACCRCAC	0.909	2.543e-5	MA0002.2	0.294919
CVAAAACCAC	0.875	1.261e-6	MA0073.1	0.372434
CVRAAACCAC	0.818	1.805e-13	MA0002.2	0.381558
SRNWAACCAC	0.717	3.148e-29	MA0002.2	0.384224
CRNWAACCAC	0.803	3.704e-21	MA0002.2	0.438688
CSDAAACCAC	0.864	2.031e-8	UP00040_1	0.526117
HWACCRC	0.502	2.794e-93	MA0002.2	0.561989
NDACCRC	0.453	1.738e-95	MA0002.2	0.605297
NWACCRC	0.483	1.606e-94	MA0002.2	1.41109

Table 7.3, shows the standard Erythroid versus Jurkat predictions, while Table 7.2 shows Erythroid predictions after deleting high-scoring Jurkat motifs. In this case, it seems like erasing these predictions only negatively affected the subsequent run. Only two motifs in Table 7.2 align with an E-value < 0.05, both for MA0470.1 (E2F4)[1]. Indeed, E2F4 is a

[1]See JASPAR reference for MA0470.1 (E2F4): http://jaspar.genereg.net/cgi-bin/jaspar_db.pl?ID=MA0470.1&rm=present&collection=CORE

transcription factor involved in cell cycles[2] and is found in Homo sapiens.

Overall, this experiment shows that it is possible to use solutions produced from GP to discriminate further against certain known solutions. This is only presented as plans for future works as we did not control a few aspects of this experiment, for example: 1) We did not run multiple seeded runs of this experiment, this is only a single trial. 2) We did not address any considerations about dataset sizes. Since the Erythroid dataset has 5706 input sequences, and the Jurkat dataset only has 2238, there might be some class imbalance issues as the GP is learning motifs from these datasets. 3) We chose to execute each run for 150 generations on the NGSA-II$_R$ algorithm. A smaller or larger generation count would impact the results for either phase of the algorithm. A different multiobjective algorithm (such as SPEA2) could also have impact. This also goes for the multiobjective fitness function. Still, despite these considerations, we still show an instance where sequential covering gives light to new predictions which were not identifiable before, thus making this a good avenue for future work.

7.1 Known issues

MotifGP uses the Python *str*.search() (a regular expression matching function) for its network expression evaluation. In our datasets, we have sequences that have N nucleotides, representing *any* of the bases represented by A,C,G,T. We did not consider that [ACGT] would not match the N wildcard base. This can be easily fixed at the phase where expressions are compiled by simply adding a N character alternative option (i.e. $T \rightarrow (T|N)$ or $[AC] \rightarrow [ACN]$). What remains to be investigated is whether or not this property is desirable. Perhaps excluding N from the search alphabets allow the solution search to be more precise and discriminate. This requires further investigation, though at this stage, the N symbol is disregarded from the prediction search (though they are handled by the dinucleotide shuffling algorithm).

[2]Gene reference for E2F4: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene&cmd=Retrieve&dopt=full_report&list_uids=1874

APPENDICES

Appendix A

Additional top motifs summary from combined experiments

This section lists top additional top motifs identified by the combined MotifGP seeded runs. Predictions are ordered by TOMTOM alignment E-value, with the # column listing the best ranking being 1 up until 15. The Seed column represents the random number generator initial value (or the run ID) and rank shows the motifs rank within the Pareto front according to Discrimination and Fisher's exact test within other motif of its seed.

A.1 E2f1

Table A.1 shows the top motifs predicted for the E2f1 dataset.

Table A.1: Best predictions fom MotifGP over the E2f1 dataset.

#	Algorithm	Seed	Rank	NE	Fisher	Discrimination	Target ID	E-value	Span
1	SPEA2	6	31	BCACTTCCGS	1.938e-29	0.947	MA0062.2	7.38246E-007	10
2	SPEA2	6	28	YACTTCCKSB	5.726e-44	0.881	MA0473.1	2.89194E-006	10
3	SPEA2	5	7	YTTCCKK	1.314e-75	0.664	UP00407_1	3.43168E-006	7
4	SPEA2	9	59	VGDANVCGGAA	3.050e-22	0.886	UP00015_1	4.98975E-006	11
5	SPEA2	1	44	VAVCMGGAAG	1.359e-33	0.870	MA0473.1	6.01377E-006	10
6	SPEA2	1	45	VAVCMGGAAG	1.359e-33	0.87	MA0473.1	6.01377E-006	10
7	NSGA-II$_R$	7	28	BBRCTTCCGG	5.930e-35	0.870	MA0062.2	7.30206E-006	10
8	SPEA2	1	5	VMGGAAR	4.363e-95	0.646	UP00407_1	8.46323E-006	7
9	SPEA2	1	6	VMGGAAR	4.363e-95	0.646	UP00407_1	8.46323E-006	7
10	NSGA-II$_R$	4	6	VMGGAAR	3.255e-78	0.630	UP00407_1	8.46323E-006	7
11	SPEA2	5	4	YTTCCKB	6.498e-93	0.646	UP00407_1	8.46323E-006	7
12	SPEA2	7	5	NYTTCCKB	1.466e-85	0.640	UP00407_1	8.46323E-006	8
13	SPEA2	7	6	NYTTCCKB	1.466e-85	0.640	UP00407_1	8.46323E-006	8
14	NSGA-II$_R$	7	6	YTTCCKB	2.185e-85	0.639	UP00407_1	8.46323E-006	7
15	SPEA2	8	12	VMGGAAR	2.763e-93	0.639	UP00407_1	8.46323E-006	7

A.2 STAT3

Table A.2 shows the top motifs predicted for the STAT3 dataset.

Table A.2: Best predictions fom MotifGP over the STAT3 dataset.

#	Algorithm	Seed	Rank	NE	Fisher	Discrimination	Target ID	E-value	Span
1	SPEA2	1	10	TTCCNRGAA	1.82E-083	0.986	MA0137.3	4.63074E-006	9
2	SPEA2	1	11	TTCCNRGAA	1.82E-083	0.986	MA0137.3	4.63074E-006	9
3	SPEA2	9	4	TTCCHRGMA	1.98E-095	0.972	MA0144.2	1.33301E-005	9
4	SPEA2	9	5	TTCCHRGMA	1.98E-095	0.972	MA0144.2	1.33301E-005	9
5	FORTIN	0	9	TTCYNGGAA	1.96E-076	0.979	MA0137.3	2.31504E-005	9
6	FORTIN	0	10	TTCYNGGAA	1.96E-076	0.979	MA0137.3	2.31504E-005	9
7	FORTIN	3	11	TTCYNGGAA	1.96E-076	0.979	MA0137.3	2.31504E-005	9
8	SPEA2	9	3	TTCCNRGMA	8.29E-107	0.97	MA0144.2	5.17863E-005	9
9	FORTIN	3	8	TKCYNGGAA	1.42E-096	0.966	MA0144.2	6.51852E-005	9
10	FORTIN	0	14	TTCYVGGAA	9.97E-060	0.995	MA0137.3	0.000118353	9
11	FORTIN	0	15	TTCYVGGAA	9.97E-060	0.995	MA0137.3	0.000118353	9
12	FORTIN	3	12	TTCYVGGAA	4.74E-055	0.98	MA0137.3	0.000118353	9
13	FORTIN	3	13	TTCYVGGAA	4.74E-055	0.98	MA0137.3	0.000118353	9
14	FORTIN	3	14	TTCYVGGAA	4.74E-055	0.98	MA0137.3	0.000118353	9
15	FORTIN	3	15	TTCYVGGAA	4.74E-055	0.98	MA0137.3	0.000118353	9

A.3 Other datasets

Other dataset results available upon request.

References

[1] Firoz Ahmed, Vagner A Benedito, and Patrick Xuechun Zhao. Mining functional elements in messenger RNAs: overview challenges, and perspectives. *Frontiers in plant science*, 2(NOV), 2011.

[2] Stephen F Altschul and Bruce W Ericksont. Significance of nucleotide sequence alignments: A method for random sequence permutation that preserves dinucleotide and codon usage1. *Mol. Biol. Evol.*, 2(6):526–538, 1985.

[3] Stephen F Altschul, E Michael Gertz, Richa Agarwala, Alejandro A Schäffer, and Yi-Kuo Yu. PSI-BLAST pseudocounts and the minimum description length principle. *Nucleic Acids Res.*, 37(3):815–824, February 2009.

[4] Wanyuan Ao, Jeb Gaudet, W James Kent, Srikanth Muttumu, and Susan E Mango. Environmentally induced foregut remodeling by PHA-4/FoxA and DAF-12/NHR. *Science*, 305(5691):1743–1746, 17 September 2004.

[5] Ricardo A Baeza-Yates and Gaston H Gonnet. Fast text searching for regular expressions or automaton searching on tries. *J. ACM*, 43(6):915–936, November 1996.

[6] Timothy L Bailey. Discovering sequence motifs. *Methods in molecular biology (Clifton, NJ)*, 452:231–251, 2008.

[7] Timothy L Bailey. DREME: motif discovery in transcription factor ChIP-seq data. *Bioinformatics*, 27(12):1653–1659, 15 June 2011.

[8] Timothy L Bailey, Mikael Boden, Fabian A Buske, Martin Frith, Charles E Grant, Luca Clementi, Jingyuan Ren, Wilfred W Li, and William S Noble. MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Res.*, 37(Web Server issue):W202–8, July 2009.

[9] Timothy L Bailey and C Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2:28–36, 1994.

[10] Timothy L Bailey, James Johnson, Charles E Grant, and William S Noble. The MEME suite. *Nucleic Acids Res.*, 43(W1):W39–49, 1 July 2015.

[11] Timothy L Bailey, Pawel Krajewski, Istvan Ladunga, Celine Lefebvre, Qunhua Li, Tao Liu, Pedro Madrigal, Cenny Taslim, and Jie Zhang. Practical guidelines for the comprehensive analysis of ChIP-seq data. *PLoS Comput. Biol.*, 9(11):e1003326, 14 November 2013.

[12] I Ben-Gal, A Shani, A Gohr, J Grau, S Arviv, A Shmilovici, S Posch, and I Grosse. Identification of transcription factor binding sites with variable-order bayesian networks. *Bioinformatics*, 21(11):2657–2666, 1 June 2005.

[13] Laurie A Boyer, Tong Ihn Lee, Megan F Cole, Sarah E Johnstone, Stuart S Levine, Jacob P Zucker, Matthew G Guenther, Roshan M Kumar, Heather L Murray, Richard G Jenner, David K Gifford, Douglas A Melton, Rudolf Jaenisch, and Richard A Young. Core transcriptional regulatory circuitry in human embryonic stem cells. *Cell*, 122(6):947–956, 23 September 2005.

[14] Alvis Brazma, I Jonassen, I Eidhammer, and Don Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of computational biology : a journal of computational molecular cell biology*, 5(2):279–305, 1998.

[15] Vince Buffalo. *Bioinformatics Data Skills: Reproducible and Robust Research With Open Source Tools*. Oreilly & Associates Incorporated, 25 May 2015.

[16] M Burset and R Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–367, 15 June 1996.

[17] Xi Chen, Han Xu, Ping Yuan, Fang Fang, Mikael Huss, Vinsensius B Vega, Eleanor Wong, Yuriy L Orlov, Weiwei Zhang, Jianming Jiang, Yuin-Han Loh, Hock Chuan Yeo, Zhen Xuan Yeo, Vipin Narang, Kunde Ramamoorthy Govindarajan, Bernard Leong, Atif Shahab, Yijun Ruan, Guillaume Bourque, Wing-Kin Sung, Neil D Clarke, Chia-Lin Wei, and Huck-Hui Ng. Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell*, 133(6):1106–1117, 13 June 2008.

[18] Peter Clote, Fabrizio Ferré, Evangelos Kranakis, and Danny Krizanc. Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. *RNA*, 11(5):578–591, May 2005.

[19] Peter J A Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J L de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 1 June 2009.

[20] Modan K Das and Ho-Kwok Dai. A survey of DNA motif finding algorithms. *BMC bioinformatics*, 8 Suppl 7:S21, 2007.

[21] K Deb, A Pratap, S Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6(2):182–197, April 2002.

[22] Patrik D'haeseleer. How does DNA sequence motif discovery work? *Nature biotechnology*, 24(8):959–961, August 2006.

[23] Kyle Ellrott, Chuhu Yang, Frances M Sladek, and Tao Jiang. Identifying transcription factor binding sites through markov chain optimization. *Bioinformatics*, 18 Suppl 2:S100–9, 2002.

[24] A V Favorov, M S Gelfand, A V Gerasimova, D A Ravcheev, A A Mironov, and V J Makeev. A gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length. *Bioinformatics*, 21(10):2240–2245, 15 May 2005.

[25] R A Fisher. On the interpretation of $\chi 2$ from contingency tables, and the calculation of P. *J. R. Stat. Soc.*, 85(1):87–94, 1 January 1922.

[26] W M Fitch. Random sequences. *J. Mol. Biol.*, 163(2):171–176, 15 January 1983.

[27] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *J. Mach. Learn. Res.*, 13(1):2171–2175, July 2012.

[28] Félix-Antoine Fortin and Marc Parizeau. Revisiting the NSGA-II crowding-distance computation. *ACM*, pages 623–630, 6 July 2013.

[29] Martin C Frith, Ulla Hansen, John L Spouge, and Zhiping Weng. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res.*, 32(1):189–200, 2 January 2004.

[30] David L González-Álvarez, Miguel A Vega-Rodríguez, and Álvaro Rubio-Largo. Multiobjective optimization algorithms for motif discovery in DNA sequences. *Genet. Program. Evolvable Mach.*, 16(2):167–209, June 2015.

[31] DL González-Álvarez, MA Vega-Rodríguez, JA Gomez-Pulido, and JM Sanchez-Perez. Predicting DNA motifs by using evolutionary multiobjective optimization. *IEEE Trans. Syst. Man Cybern. C Appl. Rev.*, 42(6):913–925, November 2012.

[32] W N Grundy, T L Bailey, and C P Elkan. ParaMEME: a parallel implementation and a web interface for a DNA and protein motif discovery tool. *Comput. Appl. Biosci.*, 12(4):303–310, August 1996.

[33] Shobhit Gupta, John A Stamatoyannopoulos, Timothy L Bailey, and William Stafford Noble. Quantifying similarity between motifs. *Genome Biol.*, 8(2):R24, 2007.

[34] Amine Heddad, Markus Brameier, and Robert M. MacCallum. Evolving regular Expression-Based sequence classifiers for protein nuclear localisation. In *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, pages 31–40. Springer Berlin Heidelberg, 1 January 2004.

[35] Wetterstrand KA. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). www.genome.gov/sequencingcosts, 2015-08-15.

[36] Wetterstrand KA. The NCBI Sequence Read Archive. www.ncbi.nlm.nih.gov/Traces/sra, 2015-08-15.

[37] Wolfgang Kantschik and Wolfgang Banzhaf. Linear-Tree GP and its comparison with other GP structures. In *Genetic Programming*, Lecture Notes in Computer Science, pages 302–312. Springer Berlin Heidelberg, 2001.

[38] Christopher Kauffman and George Karypis. Computational tools for protein-DNA interactions. *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery*, 2(1):14–28, 2012.

[39] Janne Korhonen, Petri Martinmäki, Cinzia Pizzi, Pasi Rastas, and Esko Ukkonen. MOODS: fast search for position weight matrix matches in DNA sequences. *Bioinformatics*, 25(23):3181–3182, 1 December 2009.

[40] Leping Li. GADEM: a genetic algorithm guided formation of spaced dyads coupled with an EM algorithm for motif discovery. *J. Comput. Biol.*, 16(2):317–329, February 2009.

[41] Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many sequences. *J. Comput. System Sci.*, 65(1):73–96, August 2002.

[42] Martin Luerssen and David Powers. Fast Grammar-Based evolution using memoization. In *Parallel Problem Solving from Nature, PPSN XI*, Lecture Notes in Computer Science, pages 502–511. Springer Berlin Heidelberg, 2010.

[43] Xiaoyan Ma, Daphne Ezer, Carmen Navarro, and Boris Adryan. Reliable scaling of position weight matrices for binding strength comparisons between transcription factors. *BMC Bioinformatics*, 16:265, 20 August 2015.

[44] Robert M MacCallum. Introducing a perl genetic programming system - and can meta-evolution solve the bloat problem? In *Genetic Programming*, Lecture Notes in Computer Science, pages 364–373. Springer Berlin Heidelberg, 1 January 2003.

[45] Philip Machanick and Timothy L Bailey. MEME-ChIP: motif analysis of large DNA datasets. *Bioinformatics*, 27(12):1696–1697, 15 June 2011.

[46] T K Man and G D Stormo. Non-independence of mnt repressor-operator interaction determined by a new quantitative multiple fluorescence relative affinity (QuMFRA) assay. *Nucleic Acids Res.*, 29(12):2471–2478, 15 June 2001.

[47] Tobias Marschall and Sven Rahmann. Efficient exact motif discovery. *Bioinformatics*, 25(12):i356–i364, 15 June 2009.

[48] Anthony Mathelier, Xiaobei Zhao, Allen W Zhang, François Parcy, Rebecca Worsley-Hunt, David J Arenillas, Sorana Buchman, Chih-Yu Chen, Alice Chou, Hans Ienasescu, Jonathan Lim, Casper Shyr, Ge Tan, Michelle Zhou, Boris Lenhard, Albin Sandelin, and Wyeth W Wasserman. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Res.*, 42(Database issue):D142–7, January 2014.

[49] J Mrázek. Finding sequence motifs in prokaryotic genomes - A brief practical guide for a microbiologist. *Briefings in bioinformatics*, 10(5):525–536, 2009.

[50] E W Myers. Approximate matching of network expressions with spacers. *J. Comput. Biol.*, 3(1):33–51, 1996.

[51] Daniel E Newburger and Martha L Bulyk. UniPROBE: an online database of protein binding microarray data on protein-DNA interactions. *Nucleic Acids Res.*, 37(Database issue):D77–82, January 2009.

[52] T T Nguyen and I P Androulakis. Recent advances in the computational discovery of transcription factor binding sites. *Algorithms*, 2(1):582–605, 2009.

[53] Carmen G Palii, Carolina PerezIratxeta, Zizhen Yao, Yi Cao, Fengtao Dai, Jerry Davison, Harold Atkins, David Allan, F Jeffrey Dilworth, Robert Gentleman, Stephen J Tapscott, and Marjorie Brand. Differential genomic targeting of the transcription factor TAL1 in alternate haematopoietic lineages. *EMBO J.*, 30(3):494–509, 2 February 2011.

[54] G Pavesi, G Mauri, and G Pesole. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, 17 Suppl 1:S207–14, 2001.

[55] Giulio Pavesi, Paolo Mereghetti, Giancarlo Mauri, and Graziano Pesole. Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res.*, 32(Web Server issue):W199–203, 1 July 2004.

[56] R Pearson. [s] rapid and sensitive sequence comparison with FASTP and FASTA. *Proc. Natl. Acad. Sci. U. S. A.*, 85:2444, 1988.

[57] W R Pearson and D J Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. U. S. A.*, 85(8):2444–2448, April 1988.

[58] G Pesole, N Prunella, S Liuni, M Attimonelli, and C Saccone. WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences. *Nucleic Acids Res.*, 20(11):2871–2875, 11 June 1992.

[59] P A Pevzner and S H Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 8:269–278, 2000.

[60] Yan Qi, Ping Ye, and Joel S Bader. Genetic interaction motif finding by expectation maximization–a novel statistical model for inferring gene modules from synthetic lethality. *BMC Bioinformatics*, 6:288, 6 December 2005.

[61] T D Schneider, G D Stormo, L Gold, and A Ehrenfeucht. Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188(3):415–431, 5 April 1986.

[62] Dustin E Schones, Pavel Sumazin, and Michael Q Zhang. Similarity of position frequency matrices for transcription factor binding sites. *Bioinformatics*, 21(3):307–313, 1 February 2005.

[63] Trevor Siggers and Raluca Gordân. Protein-DNA binding: complexities and multi-protein codes. *Nucleic Acids Res.*, 42(4):2099–2111, February 2014.

[64] B W Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1986.

[65] Saurabh Sinha, Mathieu Blanchette, and Martin Tompa. PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics*, 5:170, 28 October 2004.

[66] Saurabh Sinha and Martin Tompa. YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Res.*, 31(13):3586–3588, 1 July 2003.

[67] G D Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, January 2000.

[68] Vladimir B Teif and Karsten Rippe. Statistical–mechanical lattice models for protein–DNA binding in chromatin. *J. Phys. Condens. Matter*, 22(41):414105, 2010.

[69] A Ter-Sarkisov and S R Marsland. Convergence properties of (μ+ λ) evolutionary algorithms. *AAAI*, 2011.

[70] Gert Thijs, Kathleen Marchal, Magali Lescot, Stephane Rombauts, Bart De Moor, Pierre Rouzé, and Yves Moreau. A gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *J. Comput. Biol.*, 9(2):447–464, 2002.

[71] William Thompson, Eric C Rouchka, and Charles E Lawrence. Gibbs recursive sampler: finding transcription factor binding sites. *Nucleic Acids Res.*, 31(13):3580–3585, 1 July 2003.

[72] Martin Tompa, Nan Li, 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 Régnier, Nicolas Simonis, Saurabh Sinha, Gert Thijs, Jacques van Helden, Mathias Vandenbogaert, Zhiping Weng, Christopher Workman, Chun Ye, and Zhou Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.*, 23(1):137–144, January 2005.

[73] Ngoc Tam L Tran and Chun-Hsi Huang. A survey of motif finding Web tools for detecting binding site motifs in ChIP-Seq data. *Biology direct*, 9(1):1–22, February 2014.

[74] Marjan Trutschl, Phillip CSR Kilgore, Rona S Scott, Christine E Birdwell, and Urška Cvek. Detection and employment of biological sequence motifs. *Big Data Analytics in Bioinformatics and Healthcare*, page 86, 2014.

[75] J van Helden, B André, and J Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.*, 281(5):827–842, 4 September 1998.

[76] J van Helden, A F Rios, and J Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res.*, 28(8):1808–1818, 15 April 2000.

[77] Wyeth W Wasserman and Albin Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nat. Rev. Genet.*, 5(4):276–287, April 2004.

[78] E Wingender, P Dietze, H Karas, and R Knüppel. TRANSFAC: a database on transcription factors and their DNA binding sites. *Nucleic Acids Res.*, 24(1):238–241, 1 January 1996.

[79] C T Workman and G D Stormo. ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac. Symp. Biocomput.*, pages 467–478, 2000.

[80] E P Xing, W Wu, M I Jordan, and Richard M Karp. LOGOS: a modular bayesian model for de novo motif detection. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 266–276, August 2003.

[81] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Eckart Zitzler, Eckart Zitzler, Lothar Thiele, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm, 2001.