

Preface

This report is the result of a master project for the algorithms group, IDI, Norwegian University of Science and Technology. The project ran from January to July, 2005. The project is written for Interagon, with research scientist Pål Sætrum as main supervisor. The author of the report is Ola Sætrum.

I would like to thank everyone at Interagon for their guidance during this project, especially Pål Sætrum for his invaluable input on the project, and guidance on writing this report. I would also like to thank Ola Snøve for inputs on the project, and help when writing the article. Last, I would like to thank professor Arne Halaas for putting me in touch Interagon and their tasks.

Trondheim, July 19, 2005

Ola Sætrum

Contents

1	Introduction	1
2	Biological background	3
2.1	DNA and RNA	3
2.2	MicroRNAs	5
2.3	MicroRNA regulation	5
2.4	Identifying miRNA targets	6
3	Technical background	9
3.1	Learning theory	9
3.1.1	Training strategies	10
3.1.2	Scoring functions	12
3.2	Genetic programming	13
3.2.1	Individuals	14
3.2.2	Genetic operators	15
3.2.3	Fitness and selection	17
3.2.4	Introns	18
3.3	Boosting	18
3.4	Pattern Matching Chip	19
4	Previous work	21

5	Methods	23
5.1	Training the classifier	23
5.1.1	Architecture	23
5.1.2	Fitness function	24
5.1.3	Training parameters	26
5.2	MicroRNA target-site screener	26
5.2.1	Program interface	26
5.2.2	The TargetBoost algorithm	29
5.2.3	System overview	31
5.2.4	Scoring algorithm	33
6	Results	39
6.1	The TargetBoost classifier is a state of the art miRNA target site predictor	39
6.2	Efficacy comparison of different architectures	40
6.3	The average of a set of TargetBoost classifiers creates a better final classifier	42
6.4	The TargetBoost scoring is best for finding single mRNA-regulating target sites	44
6.5	The TargetBoost classifier specializes in finding 3' compensatory target sites	47
7	Discussion	51
7.1	Scoring function	51
7.2	Dataset handling	51
8	Further work	55
A	TargetBoost system	57
B	Article	59

List of Figures

2.1	The DNA molecule	4
2.2	Simple figure of the transition from DNA to protein [2].	4
2.3	A verified miRNA target site	7
3.1	Training strategies used in machine learning.	11
3.2	A sample program in a tree representation.	14
3.3	Mutation operator	15
3.4	Cross-over operator	16
5.1	TargetBoost command line user interface.	27
5.2	TargetBoost web user interface.	28
5.3	TargetBoost main result page.	28
5.4	Detailed target site alignment.	29
5.5	Translation of a miRNA-specific query	30
5.6	Correlation miRNA length and target site score	34
5.7	Maximum TargetBoost score	35
6.1	Overall ROC-curves for algorithms.	40
6.2	Performance comparisons on individual miRNAs.	41
6.3	Performance comparison for different grammars.	43
6.4	Efficacy comparison ensemble classifier	44
6.5	Correlation curves for scoring functions	45
6.6	ROC curves for scoring functions	46

6.7	Score vs document size	47
6.8	Venn diagrams	48
A.1	UML diagram TargetBoost.	58

List of Tables

3.1	IQL-query examples.	20
5.1	Grammar used in the genetic programming.	25

List of Algorithms

1	The genetic programming algorithm.	13
2	The AdaBoost algorithm	19
3	The TargetBoost algorithm	30
4	Needleman-Wunsch algorithm [5].	33
5	Optimal binding site set finder	36

Chapter 1

Introduction

MicroRNAs are a recently discovered large gene-family of short non-encoding RNAs. They have a regulatory role in the protein synthesis, targeting mRNAs for either cleavage or translation repression. While many miRNAs have been identified, the actual function of most remain unknown. As regulators for other RNAs, their function is closely connected to the mRNAs they regulate, and with no high-throughput experimental method discovered, computational methods are used to predict the regulation.

MicroRNAs regulate by binding reverse complementary to their targets, with complementarity in certain miRNA-regions being more important than in other regions. We used hardware-accelerated boosted genetic programming to create weighted sequence motifs that find a pattern in the miRNA-mRNA binding. This classifier was compared to other target site predictors and found to be state of the art.

We here present the program TargetBoost that allows for screening large mRNA datasets for miRNA target sites. Several miRNAs can be used in the screening, which allows for cooperative effects checking. TargetBoost uses the Interagon Pattern Matching Chip (PMC), a specialized circuit for pattern matching in large datasets. We found that TargetBoost is excellent in finding target sites that individually cause regulation, but not as good in finding cooperative effects between weak binding sites. We found this when using both synthetic and real datasets.

The first chapters give the reader the background needed for reading this paper, with Chapter 2 presenting the necessary biological background, and Chapter 3 presenting the background on the machine learning, genetic programming, boosting, and the PMC. Chapter 4 presents previous work published on predicting miRNA targets. Chapter 5 presents the algorithms and methods used in TargetBoost, first presenting the details around the training of the classifier,

thereafter presenting the TargetBoost system. We compared the efficacy of TargetBoost against two other published methods for miRNA target prediction. We also tested TargetBoost's ability to predict cooperative effects, and compared it to cooperative effects predicted by seed regions. These results are presented in Chapter 6. Discussions are found in Chapter 7, and further work are found in Chapter 8

Chapter 2

Biological background

To understand some of the details presented in this paper, the reader need to have some knowledge of molecular biology. This chapter gives a short introduction to the subject.

2.1 DNA and RNA

All information about a living cell is encoded in its DNA. The building blocks of DNA, the nucleotides, consist of four bases: adenine (A), cytosine (C), guanine (G), and thymine (T). The DNA molecule can therefore be described as a sequence S with the alphabet $\mathcal{A} = \{A, C, G, T\}$. Because the sequence has a direction, one of the ends is labeled 5' and the other 3'. The basic structure of the DNA-molecule is a double-helix: a strand of nucleotides is paired with another strand running in the opposite direction (see Figure 2.1). The two strands are complementary, which means the nucleotides from the two strands form Watson-Crick pairs with each other. A binds to T and C binds to G in the Watson-Crick pairs, and they bind to each other using hydrogen bonds.

Because of the complementarity of the double helix, the complete DNA molecule can be generated from a single strand. This means a DNA molecule can duplicated itself (replication). The DNA encodes all the information required by an organism to function [2], and this information is transferred from parents to offsprings within a species. DNA is used to create proteins, the building block of living cells. A simplified overview on how DNA is used to create proteins can be seen in Figure 2.2. As depicted, in addition to duplicate itself, the DNA can encode RNA, a molecule similar to DNA, but where thymine (T) has been replaced by urasil (U). Similarly to thymine, urasil binds with adenine. Additionally can urasil form weak bindings with guanine. This is called a GU-wobble, and

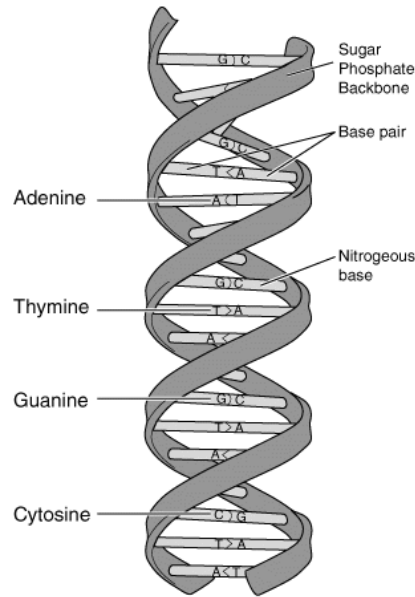


Figure 2.1: The DNA molecule. Two strands running in opposite directions form a double helix [1].

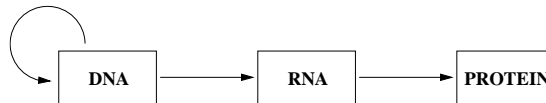


Figure 2.2: Simple figure of the transition from DNA to protein [2].

is often allowed in calculations [5]. RNA is also single stranded, while DNA is double-stranded. RNA is created by using one of the DNA strands as a template for a single strand RNA in a process called transcription. Certain RNAs, called messenger RNAs (mRNAs), can then again be translated into proteins. This process is called translation. There are also other RNAs that do not directly encode proteins, but contribute to the protein creation either by enhancing or repressing the translation process (tRNAs, rRNAs, etc). Only a part of the mRNA sequence called the coding region, or coding sequence (cds), encodes the protein. Before and after the coding sequence there are regions that do not directly encode the protein. These are called 5' nontranslated leader, or 5' untranslated region (5'UTR), and 3' nontranslated trailer, or 3' untranslated region (3'UTR) [6].

2.2 MicroRNAs

MicroRNAs or miRNAs are a relatively new family of RNAs. The first miRNA, *lin-4*, was discovered in 1993, but seven years went before similar sequences with similar properties were found and identified as a family. MicroRNAs are short, non-coding RNAs, usually 21-23 nucleotides long. They are non-coding, because they do not directly contribute in the production of proteins. Instead, miRNAs have a regulatory role in the protein production, by inhibiting the protein production from mRNAs [7].

The miRNA family is a large gene family: it is estimated that nearly 1% of the genes in the human genome, amongst others, are miRNA genes [7]. They are also conserved across species, i.e. miRNAs found in the human genome can often be found in the mouse genome. MicroRNAs can even be found in plants. This suggests that the regulatory role of the miRNAs is a basic mechanism in the production of proteins in higher species [7].

A large number of MicroRNAs have been identified using experimental or computational approaches. The identified miRNAs are registered in the miRNA registry [8]. The actual function of most of these miRNAs is, however, unknown. Since miRNAs are regulators for mRNAs, their function are closely connected to the mRNAs they regulate.

2.3 MicroRNA regulation

MicroRNAs regulate protein production by either targeting mRNAs for cleavage or for translational repression [7]. This depends on the degree of complementarity between the miRNA and target.

MicroRNAs target mRNAs by binding reverse complimentary to the mRNA, with target sites found in both the 3'UTR and coding region of mRNAs. Whether or not the regulation occurs by cleavage or translational repression, is not decided by the miRNA or the mRNA, but by the degree of complementarity between them. If the miRNA-mRNA pair has sufficient complementarity, the regulation will be in the form of cleavage. If the complementarity is of lesser degree, the regulation will be in the form of translational repression [7]. The difference between cleavage and translational repression is that the former destroys the mRNA by cleaving the sequence, while the latter only represses translation of the mRNA. Both mechanisms, however, block production of proteins.

2.4 Identifying miRNA targets

To find the actual function of a miRNA, the mRNAs that the miRNA regulates must be found. No easy, high-throughput, experimental methods for finding miRNA targets have been found yet. Computational approaches are therefore used to identify candidate target sites, which later can be confirmed using experimental approaches. Whereas miRNA-mRNA pairs in plants often have a high degree of complementarity across the whole target region, miRNA target interactions are much more complex in animals. The miRNA-mRNA pairing in animals contain only short complementary sequence stretches, interrupted by gaps and mismatches, and is therefore far from perfect [9]. This complicates the target identification considerably.

Some properties of miRNA target sites in animals have been observed. The 5' end of the miRNA are often perfectly complimentary to 3'UTR sections in the target site [7, 10, 11]. In particular nucleotides 2-8 counted from the 5' end are often perfectly complementary to the target site, and is therefore often called the nucleus or seed of the miRNA-mRNA target pair. The seed region is also the most conserved region among homologous miRNAs [7]. With this in mind, two target site categories have been identified: "5' dominant" sites and "3' compensatory" sites [9]. While the first category pairs well in the seed region, the second category pairs weakly with the seed region. It has been found that the first category requires little or no pairing in the 3' end to regulate the target. The second category is dependent of pairing in the 3' end, but even extensive pairing on the 3' end is not enough to regulate targets if there is not some 5' complementarity present. GU-wobbles in the seed region is always detrimental [9, 12].

Even if a single miRNA target site within a gene has not enough complementarity to regulate it, the miRNA might still regulate the gene if the gene has more target site candidates. Several miRNAs might also regulate target sites by cooperating together [13]. Even if a single miRNA does not have enough complementarity to singlehandedly regulate a targetsite, it can contribute to the regulation if other miRNAs also bind to the target gene.

Figure 2.3 shows an example of an experimentally verified target site.

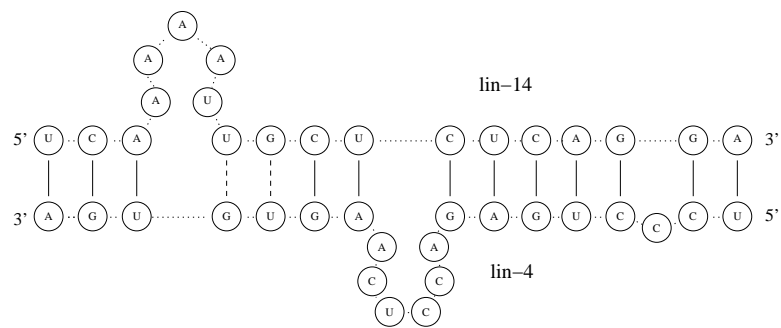


Figure 2.3: A verified target site of miRNA lin-4 within mRNA lin-14 [3].

Chapter 3

Technical background

Automatic generation of computer programs have been a goal for many years. As computers become more and more complex, it becomes more and more difficult to fully utilize their potential, without making any mistakes. To help the programmers, new programming paradigms have been introduced, including structured programming, object oriented programming among others. Even so, the quality of the programs developed are closely connected to the programming skills of their developers.

Machine learning tries to solve the complexity problem by letting the computers create the programs. The original goal was to let the computers do the programmers job: writing programs using the same code as human programmers. This is an unrealistic goal, so the focus of machine learning was changed to letting computer programs learn and getting better through their own experience. As a program runs, it uses past experience to create better predictions in its current context[14]. Having this goal, there exist several methods for achieving machine learning: support vector machines, neural networks, Bayesian networks, hidden markov models, and genetic programming. Machine learning is especially useful in classification problems, because the computer might pick up small differences in datasets that are invisible to humans.

3.1 Learning theory

In binary classification, the goal is to create a hypothesis h that correctly places every sample with a given property within one class. Samples without this property are placed in a different class. Every sample belongs to the input space X , and the output from the hypothesis belongs to the output space Y . The hypothesis can therefore be described as a function $f : X \rightarrow Y$. This function is to

be created from a set of randomly selected samples from the input-space. The function must then be able to generalize onto unseen samples in the input-space, such that future samples also can be classified correctly.

In supervised learning, the training set comes with a corresponding output. The set is randomly chosen from the unknown probability distribution $P(\mathbf{x}, \mathbf{y})$. The classifiers generalization error is given by Equation 3.1,

$$L(f) = \int \lambda(f(\mathbf{x}), \mathbf{y}) d\mathbf{P}(\mathbf{x}, \mathbf{y}) \quad (3.1)$$

where λ is a suitable loss-function [4]. The goal of the classifier training, is to minimize this generalization error. However, since $P(\mathbf{x}, \mathbf{y})$ is unknown, this generalization error cannot be calculated directly, but must be estimated based on the available data. One possible way of training the classifier, is to minimize the empirical risk (see Equation 3.2).

$$\hat{L}(f) = \frac{1}{N} \sum_{n=1}^N \lambda(f(\mathbf{x}_n), \mathbf{y}_n) \quad (3.2)$$

As $N \rightarrow \infty$, $\hat{L}(f) \rightarrow L(f)$. By minimizing the empirical risk in the training process however, chances are that the final classifier will be specialized on the training set, and will therefore generalize poorly to future samples (overfitting). To prevent overfitting, other techniques must also be used.

3.1.1 Training strategies

Several different training strategies have been developed to reduce the risk of overfitting.

Training and test set approach

In this approach the set of samples are divided into a training set and a test set (see Figure 3.1(a)). The classifier is trained using the training set, and the test set is used to estimate its generalization error [15]. The main drawback with this method lies in that in order to estimate the generalization error in a statistically significant manner, many samples must be placed in the test set. Often as many as 1/3 of the samples are used as test samples. This means fewer samples can be used in the training, and a less accurate classifier will be made. As smaller training sets increase the risk of overfitting, the best classifiers will often also be overfitted to the training set.

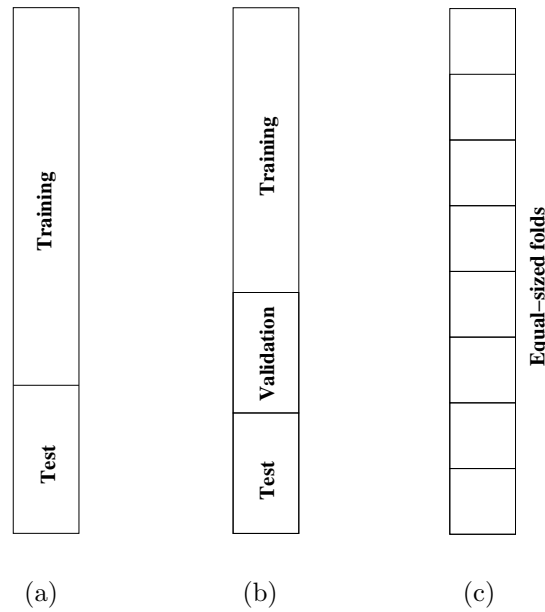


Figure 3.1: Training strategies used in machine learning.

Training, validation, and test set approach

This strategy tries to remove the overfitting problem of the training and test set approach. In this strategy, the samples are divided into a training set, a test set, and a validation set (see Figure 3.1(b)). In this approach, instead of choosing the expression scoring highest in the training set, the expression scoring highest in the validation set is chosen as the final classifier. As long as higher scores in the validation set are achieved, the training procedure continues. When the scores in the validation set drops, the training procedure is stopped, because this indicates that the classifier starts overfitting to the training set. This is called early stopping. The drawback with this strategy is that even fewer samples are used in the actual training, which means it can rarely be used on small training sets.

Cross-validation

In this strategy, the samples are randomly partitioned into a predetermined number of folds (see Figure 3.1(c)). A separate training procedure is performed for every fold, using one fold as test set and the rest as training sets. If early stopping is used, one of the folds is used as a validation set. The final classifier consists of the set of classifiers resulting from the individual training runs, which gives a much more reliable classifier. The main drawback is the increased training time.

3.1.2 Scoring functions

To compare classifiers, their performance must be quantifiable. For binary classifiers, the evaluation samples can be divided into two sets, a positive and a negative set. Based on these two sets, the output from a classification of the evaluation set consists of four numbers: the number of samples correctly placed into the the positive set (TP), the number of samples incorrectly placed into the positive set (FP), the number of samples correctly placed into the negative set (TN), and the number of samples incorrectly placed into the negative set (FN). Based on these four values, several scoring methods have been developed.

Correlation

The correlation score returns a value between -1 and 1: a score of -1 indicates that the classifier totally disagrees with the evaluation set, and a score of 1 means every sample was classified correctly. A score of 0 indicates a random classification. Using TP, FP, TN, and FN, the correlation can be calculated using Equation 3.3 [16].

$$C = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \quad (3.3)$$

Sensitivity and specificity

The goal is to find a classifier that finds as many samples in the positive set as possible (a sensitive classifier), without cluttering this with many samples from the negative set (a specific classifier). This can be expressed using sensitivity (3.4) and specificity (3.5) [16].

$$S_e = \frac{TP}{TP + FN} \quad (3.4)$$

$$S_p = \frac{TN}{FP + TN} \quad (3.5)$$

The relationship between sensitivity and specificity is usually displayed using receiver operating characteristics (ROC) curves. Different tests on the ROC-curves can be used to measure the quality of the classification. Area-tests calculate the area beneath the ROC-curve: an area of 1, indicates a perfect classification, while an area of 0.5 indicates a random classification. ROC₅₀-tests calculate the area beneath the ROC-curve plotted until 50 false positive samples are found [17].

3.2 Genetic programming

Genetic programming (GP) [18] achieves machine learning by using the concepts of natural selection and the survival of the fittest. The principle of natural selection is that in a group of individuals in a given environment, only the strongest individuals in that environment will survive and procreate. This means that only the genes of the strongest individuals will be passed on to the next generations, and the general population will therefore be better suited for this environment as new generations are created.

The basic principle behind genetic programming is to have a population of individuals, where each individual is a separate program that tries to solve a given task. This population is subjected to evolution by using natural selection and genetic operators: each individual's ability to solve the given task is measured using a fitness function, and based on their fitness, individuals are chosen for next generations. Those chosen are subjected to genetic operators before being placed in the next generation. The genetic programming proceeds in a predetermined number of generations, or until an individual with a solution close enough to the wanted solution is found.

The genetic programming process can be seen as a search in the problem space. Each individual can be seen as a proposed solution for a given problem. The search is performed in parallel because of the population of individuals, and further searches are semideterministic: already existing solution proposals are used by the genetic programming process to create new solutions.

The general genetic programming algorithm is described by Algorithm 1.

-
1. **Input:** Size of population N , number of generations T
 2. **Initialize:** Set $p_n^{(1)}$ = randomly generated individual for all $n = 1, \dots, N$
 3. **Do for** $t = 1, \dots, T$,
 - (a) **Do for** $n = 1, \dots, N$, calculate $f(p_n^{(t)})$.
 - (b) **Do**
 - Select an genetic operator.
 - Based on the selection scheme, randomly select individuals from $P^{(t)}$ for the genetic operator.
 - Evaluate operator and add result to $P^{(t+1)}$
 - until** $|P^{(t+1)}| = N$
 4. **Output:** $p_n^{(t)}, p_n^{(t)} = \max_i(f(p_i^{(t)}))$

Algorithm 1: The genetic programming algorithm.

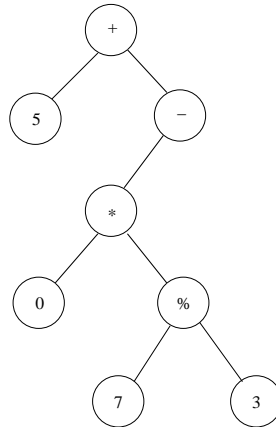


Figure 3.2: A sample program in a tree representation.

3.2.1 Individuals

The final solution is generated from a population of individuals that evolves from generation to generation until an individual with a high enough fitness score is found. Each individual is a program or an expression that is used to solve the problem at hand. The program or expression usually consists of two types of building blocks: a set of atoms and a set of functions. Atoms are constants and functions that does not have any parameters, and they can be evaluated into a single constant. Functions take one or more parameters and return a single value. Using these atoms and functions, each individual can be represented by a parse tree: atoms are leaf-nodes and functions are internal nodes (see Figure 3.2).

An important part of the genetic programming setup is to decide which atoms and functions that are to be used. If too few, or wrong atoms and functions are used in the process, it might not be possible to find an individual with an acceptable solution. On the other hand, if too many constants and functions are used, the search-space might become too large, and an acceptable solution might not be found because of this. You must also make sure that the semantics of the generated expressions are correct, that is, every possible randomly generated individual must be executable.

The genetic programming process randomly generates the first generation from the pool of atoms and functions. Later generations are evolved from the initial population using genetic operators.

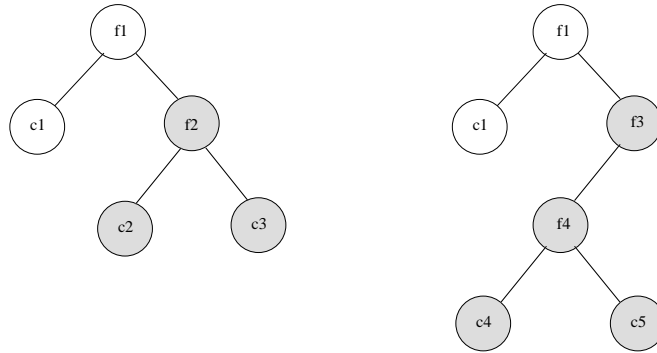


Figure 3.3: The mutation operator used on a tree representation of a program.

3.2.2 Genetic operators

Individuals chosen for the next generation are not merely copied into the next generation. To have evolution, the individuals must be subjected to small changes, and these changes are achieved by using genetic operators. There are 3 basic genetic operators: mutation, cross-over, and transfer.

Mutation

Mutation happens when the genetic material is changed in a single individual. There are two types of mutations: point mutation and area mutation. In point mutation, a single genetic code is changed (i.e. an adenine is changed to a guanine in the genetic code), and in area mutation, an area of the genetic code is randomly changed (i.e. a DNA-subsequence is randomly shuffled). In genetic programming, point mutation is achieved by randomly changing a leaf node in the program tree, or by changing an internal node with another with the same number of children. Area mutation is achieved by randomly selecting an internal node, deleting its subtree, and creating a new subtree using the original tree-creation process (see Figure 3.3).

Cross-over

Cross-over is the sexual method for evolution: two parents are selected to procreate, create an offspring. The genetic code of the offspring consists of some of the genetic code from the father and some of the genetic code of the mother. In genetic programming this is accomplished by selecting two individuals for the operator. In each individual, a node is randomly selected as the cross-over point. The subtrees below these cross-over points are interchanged, and the new individuals created are placed in the new generation (see Figure 3.4).

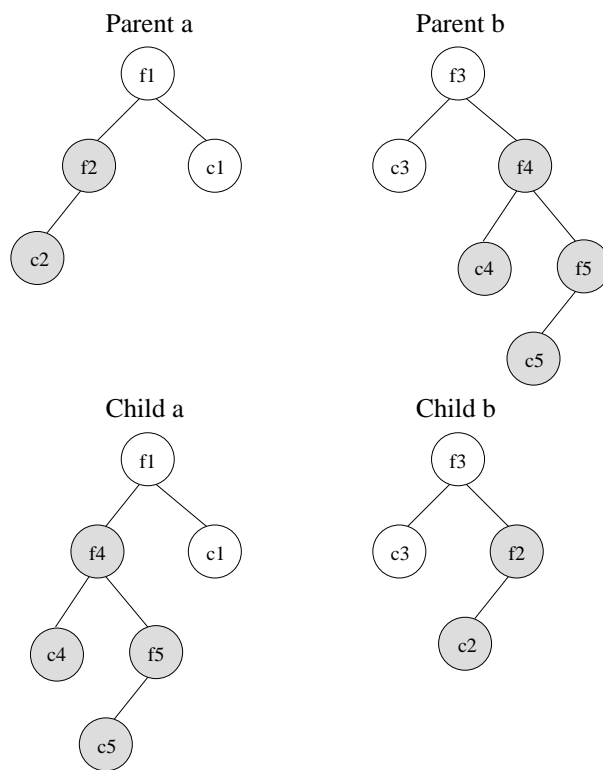


Figure 3.4: The cross-over operator used on two individuals in tree representation.

Transfer

The transfer operator is the simplest form of genetic operator. The operator takes an individual and copies it to the next generation as it is. It does not represent an actual evolutionary operator, but it is often used in genetic programming to help preserving “strong” individuals for later generations. This can keep the search more focused, as it prevents the search from randomly jumping through the search space.

3.2.3 Fitness and selection

In every generation, the fitness of each individual is evaluated using a fitness function. The fitness function is highly problem specific, and designing a good fitness function that discriminates well between good solutions and bad solutions is essential for making the genetic programming process work. In a normalized fitness function, the function returns a value between 0 and 1, with 1 being a perfect solution.

After the fitness of every individual have been evaluated, individuals for the next generation are selected using a given selection scheme. Several different selection schemes have been developed, where the different schemes have different selection pressure. High selection pressure means that individuals with high fitness have much higher probability of being selected than individuals with low probability.

Fitness-proportional selection

In fitness-proportional selection, the probability of a individual being selected is directly tied to its fitness value. The probability is given by Equation 3.6[14].

$$p_i = f_i / \sum_{j=0}^N f_j \quad (3.6)$$

Rank selection

In rank selection, the probability of a individual being selected is given by its fitness rank within a generation. The individuals are sorted based on their fitness value, so individuals with high fitness are ranked before individuals with low fitness. The selection probability is a function of the rank.

Tournament selection

In tournament selection, the selection competition is not between the whole generation, but between individuals in a randomly picked subset of the generation. Different strategies for selection based on the subset are used, with the most common being selecting the individual with highest fitness. The size of the subset is called the tournament size. The selection pressure can be varied by changing the tournament size. The larger the tournament size, the higher the selection pressure[14].

3.2.4 Introns

Almost every program developed using genetic programming contains code that have no effect on its ability to solve the given problem. These code sequences are called introns[14], and Equation 3.7 gives an example.

$$\text{constant} = \text{constant} * 1 \quad (3.7)$$

The existence of introns often cause the final programs to bloat and become much larger than necessary. In later generations, much of the code is in fact introns. Therefore will the genetic programming process almost always stagnate: beneficial changes in the code becomes more and more unlikely, and no evolution occurs between the generations. The large individuals also consume more computer resources, causing the evolution to progress slower and slower[14].

3.3 Boosting

The principle of boosting is to combine several classifiers into an ensemble that works better than each single classifier in the ensemble. Let h_1, h_2, \dots, h_N be a set of hypotheses. The final ensemble classifier is then defined by Equation 3.8,

$$f(x) = \sum_{n=1}^N \alpha_n h_n(x) \quad (3.8)$$

where α_n are coefficients with which the hypotheses are combined. Both α_n and h_n are outputs from the boosting process. Each of the hypotheses must only be slightly better than a random classifier for the boosting process to work[4].

The basic principle behind the AdaBoost-algorithm is to assign a weight to each sample in the training set. After a training run, the weights of samples that are

misclassified, are increased. By manipulating the weights, the training process will concentrate its efforts more on samples that are hard to classify later in the boosting process. The AdaBoost-algorithm is described in Algorithm 2 [4].

-
1. **Input:** $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, Number of iterations T
 2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \dots, N$
 3. **Do for** $t = 1, \dots, T$,
 - (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \mathbf{x} \rightarrow \{-1, +1\}$, i.e. $h_t = \mathcal{L}(S, \mathbf{d}^{(t)})$
 - (b) Calculate the weighted training error ϵ_t of h_t :

$$\epsilon_t = \sum_{n=1}^N d_n^{(t)} \mathbf{I}(y_n \neq h_t(x_n)),$$

- (c) Set:

$$\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

- (d) Update weights:

$$d_n^{(t+1)} = d_n^{(t)} e^{-\alpha_t y_n h_t(\mathbf{x}_n)} / Z_t,$$

where Z_t is a normalization constant, such that $\sum_{n=1}^N d_n^{(t+1)} = 1$.

4. **Break if** $\epsilon_t = 0$ or $\epsilon_t \geq 1/2$ and set $T = t - 1$.
5. **Output:** $f_T(\mathbf{x}) = \sum_{t=1}^T \frac{\alpha_t}{\sum_{r=1}^T \alpha_r} h_t(\mathbf{x})$

Algorithm 2: The AdaBoost algorithm [4]. At step t in the boosting process, a non-negative weighting $\mathbf{d}^{(t)} = (d_1^{(t)}, \dots, d_N^{(t)})$ is assigned to each sample. A hypothesis is created based on these weights, and based on the errors produced by the hypothesis, the weights are updated. $\mathbf{I}(E)$ is a function that returns 1 if the event E occurs, 0 otherwise.

3.4 Pattern Matching Chip

The Interagon Pattern Matching Chip (PMC) is an application specific integrated circuit used for pattern matching at constant speed in large datasets [19]. The queries used in the pattern matching range from simple word matching to complex regular expressions, and several queries can be screened in parallel because of its multiple instruction, single data (MISD) architecture. 16 PMCs have been placed on a single search card, and a single computer can have several cards installed. A computer with four such cards have a theoretical performance of 7×10^{12} character comparisons per second. Each PMC have 128 MB of local memory, which means large data sets have to be distributed on more PMCs.

The queries used on the PMC are expressed using a high level query language

IQL query	Interpretation
ACU*GGG	Find an A and a C followed by 0 or more consecutive Us followed by 3 Gs.
{ACGGGCUA:p>4}	Find the sequence “ACGGGCUA” with 3 or less mismatches.
{UACCC:d=5}{.:r=5}A(A U)CC	Find the sequence “UACCC” followed by the sequence “AACC” or “AUCC” with 5 to 10 wildcards between (a wildcard matches any character).

Table 3.1: IQL-query examples.

called the Interagon Query Language (IQL) [20]. IQL is a superset of regular expressions (see Table 3.1 for IQL-examples).

The PMC is configured using the PMC Specification Language (PSL). PSL is a low-level query/chip specification language that allows for detailed control over the PMC-configuration. When searching, an IQL-query is first translated into a PSL-query, which is then used to configure the PMC and perform the search.

Chapter 4

Previous work

A number of computational approaches for identifying miRNA target sites have been suggested. Most methods predict candidate sites by using sequence complementarity and/or free energy calculations on secondary structures. Since the complementarity between miRNA and mRNA is low, relying on matches in a single genome can be ineffective. If, however, the target site is conserved between species, the confidence in the prediction is increased since it signifies a potential functional constraint [21]. Some algorithms therefore use target site conservation as an additional filter [22, 23]. A drawback with this method is that the 3' UTR regions in some genomes have not been verified experimentally, introducing some uncertainty in the calculations performed. Also, even if a target site is not conserved across species, it might still be a valid target site in the species examined. Another filter frequently used is to require multiple target sites within the same gene for a single miRNA. Several binding sites can improve target regulation, but in the end, even a gene with a single miRNA target site can be regulated, so single sites should not be disregarded [21].

Rajewsky and Socci [10] propose an algorithm whose initial step is to search for a consecutive sequence of complementary base pairing between miRNA and mRNA, called binding nucleus. The score of the binding nucleus is the weighted sum of base pairs. Using a cutoff on this score, the free energy between the target candidate and miRNA is calculated and used as a second filter. The free energy is calculated using MFOLD [24].

TargetScan [22] also uses a seed step before calculating free energy score. TargetScan demands perfect Watson-Crick pairing in nucleotides 2-8 in the miRNA, and secondary structure is predicted by using RNAfold [25]. As a last step, TargetScan uses conservation between different species.

Robins et al. [26] propose another algorithm that uses the seed region as an initial filter. Instead of using the free energy across the complete miRNA-mRNA pairing

as a second step, the algorithm uses a scoring metric that treats Watson-Crick pairing positively, GU-wobbles neutrally, and mismatches and gaps negatively. It also incorporates 3' UTR structure of the target site, and scoring from multiple sites within targets to predict regulation.

The RNAhybrid-algorithm [11] uses free energy calculations to predict miRNA target site candidates. It finds the energetically most favorable hybridization between a long mRNA sequence and a short miRNA sequence. RNAhybrid can demand perfect Watson-Crick complementarity in sections of the miRNA-mRNA pairing, usually nucleotides 2-7.

The algorithms mentioned above only try to predict single miRNA target sites. The output is usually the binding site position within the dataset, and some scoring that measure the likelihood of the binding site being an actual target site. None of them, with the exception of TargetScan, try to incorporate into the scoring scheme the effect of multiple binding sites within the target gene, or of multiple miRNAs cooperating in the target regulation. TargetScan allows for multiple binding sites between miRNA and mRNA. However, the PicTar sequence scoring algorithm [23] is the first algorithm that incorporates both multiple binding sites and miRNA cooperation in the scoring. PicTar models the 3' UTR as a Hidden Markov Model (HMM) [16, 27]. The HMM states consist of the binding sites for a set of miRNAs and the "background". PicTar uses expectation maximization (EM) to calculate the parameters in the HMM, and the final score is the log ratio between the 3' UTR sequence being generated by this model and being generated by a background process (a Markov Model of order 0). The actual improvement of the PicTar algorithm lies in the modeling: the miRNA binding sites uses in the HMM is found by using a variant of the binding nucleus of Rajewsky and Socci [10] as an initial seeding step and free energy score calculated by RNAhybrid [11] as a second seeding step. The method has previously been used to model transcriptional regulation [28].

Chapter 5

Methods

TargetBoost searches for miRNA target sites in mRNA by using weighted sequence motifs. These motifs were created using a variant of the GPboost-algorithm [29, 30], a machine learning algorithm that combines genetic programming and boosting, using GP as the base learner, and AdaBoost as the boosting algorithm. The motifs are expressed using the Interagon Query Language, enabling the learning process to be performed using the Interagon Pattern Matching Chip.

5.1 Training the classifier

The miRNA target site is highly sequence specific, but as mentioned in Chapter 2.4, there seems to be a pattern in how the miRNA bind to the target mRNA. The goal of the machine learning process is to find this pattern.

5.1.1 Architecture

The main input into the GPboost algorithm is the architecture used to create the individuals in the GP-population. The architecture defines the atoms and functions that are to be used in the program-generation, and therefore also defines how complex the individual programs can become. Since GPboost uses Strongly Typed Genetic Programming (STGP) [31], a method that ensures that the generated expressions are of legal type, the architecture also defines the legal grammar the expressions must follow.

We created several architectures with different amount of domain knowledge incorporated into the architecture. The production rules and the semantic meaning

of the main architecture, “Displaced-Ordered”, can be seen in Table 5.1. The production column presents the grammar in Backus-Naur form, with non-terminals represented by uppercase-letters, and terminals by boldface letters. The second column describes the semantic meaning of the productions. This architecture tries to encode the knowledge that a miRNA binds almost perfectly to its target site at its 5’ end, and imperfectly on its 3’ end. According to this architecture, each individual consists of two parts: an unspecified pattern (R in Table 5.1) followed by a consecutive sequence of near perfect matches (O in Table 5.1). These two parts are separated by a variable amount of nucleotides. The separation-length is decided by the separation D : the lower bound is given by the number of wildcards in the W -production, the upper bound by the number of wildcards plus the displacement d in the D -production. The architectures create template expressions. The terminals, P_n , represent positions in the miRNA sequence counted from the 3’ end. Therefore, before screening a miRNA, the terminals must be exchanged with the appropriate nucleotides.

An expression matches a string if $R.hit$ is true. $match(\mathbf{a})$ returns 1 if the character in the position indicated by \mathbf{a} equals the character it is compared against. $linger(F.hit, N)$ is a function such that if $F.hit$ is true, 1 is returned for N clock-ticks.

A second architecture, “Simple-Ordered”, was created by removing the D from the S -production. This architecture is therefore much simpler: The individuals will be much smaller with less parameters to tune. A third architecture, “Displaced-Pnofm” was created by exchanging the O in the S -production with an R . This architecture is more relaxed than “Displaced-Ordered”, because the second part can be a more arbitrary expression in stead of a consecutive sequence of near perfect matches.

5.1.2 Fitness function

The fitness of an expression is calculated by using the PMC to screen the expression against a positive and a negative dataset. The positive dataset consists of 36 experimentally confirmed target sites for the miRNAs let-7, lin-4, miR-13a, and bantam [32, 10, 33]. The target sites are padded such that they are 30 nucleotides long. The negative set consists of 3000 random strings all 30 nucleotides long. The nucleotide frequencies used in the sequence generation are the same as used in [10] ($p_A = 0.34$, $p_C = 0.19$, $p_G = 0.18$, $p_U = 0.29$). Each string of 30 nucleotide is called a document.

Since our expressions are template expressions (see Chapter 5.1.1), we must translate them into sequence specific expressions using the miRNA we want to screen, before performing the search. Since the expression should generalize to all miR-

	Production	Semantic Rule
(1)	$S \rightarrow (D)(O)$	$S.hit := D.hit \text{ AND } O.hit$
(2)	$D \rightarrow \{F : d = N\}$	$D.hit := linger(F.hit, N)$
(3)	$F \rightarrow (R)(W)$	$F.hit := R.hit \text{ AND } W.hit$
(4)	$R \rightarrow \{C : p \geq N\}$	$R.hit := C.count \geq N.cutoff$
(5)	$R \rightarrow C$	$R.hit := C.hit$
(6)	$C \rightarrow (C_1)(C_2)$	$C.count := C_1.count + C_2.count$ $C.hit := C_1.hit \text{ AND } C_2.hit$
(7)	$C \rightarrow A$	$C.count := A.count$ $C.hit := A.hit$
(8)	$A \rightarrow A_1 \mid A_2$	$A.count := A_1.count + A_2.count$ $A.hit := A_1.hit \text{ OR } A_2.hit$
(9)	$A \rightarrow L$	$A.count := L.count$ $A.hit := L.hit$
(10)	$L \rightarrow \mathbf{a}$, for some $\mathbf{a} \in \{P_1, \dots, P_{24}\}$	$L.count := match(\mathbf{a})$ $L.hit := match(\mathbf{a})$
(11)	$N \rightarrow \mathbf{n}$, for some $\mathbf{n} \in \{1, 2, \dots\}$	$N.cutoff := \mathbf{n}$
(12)	$W \rightarrow \{. : r = N\}$	$W.hit := 1$
(13)	$O \rightarrow \{LC : p \geq N\}$	$O.hit := LC.count \geq N.cutoff$
(14)	$LC \rightarrow (LC_1)(LC_2)$	$LC.count := LC_1.count + LC_2.count$
(15)	$LC \rightarrow L$	$LC.count := L.count$

Table 5.1: Grammar used in the genetic programming.

NAs, several miRNAs are used in the training process. Several searches are therefore performed before the expression's fitness is calculated. Each miRNA in the training set has a different set of positive target sites. This must be allowed for in the fitness calculation. We used Equation 5.1 for fitness calculation. A perfect classification will here give a score of 0, while a wrong classification give a low score. $poshit(i)$ returns 1 if document i in the positive dataset was hit by a miRNA which has document i as a part of its positive dataset. d_i is the weight of document i .

$$f(ind) = -\left(\sum_{i=1}^N -d_i \cdot poshit(i) - 1\right) + \sum_{j=1}^M d_j \cdot hit(j) \quad (5.1)$$

5.1.3 Training parameters

We perform 10 separate boosting runs when training the classifier. Since GP is a stochastic process, there will be variations in the output from separate training runs. By combining several such classifiers, a better classifier is created [34](see also Chapter 6.3). The final classifier is the average of the 10 independent boosting runs: each boosting run had 25 boosting iterations, and each GP-run had 500 individuals per generation, and ran 75 generations before terminating. This means the final classifier consisted of 250 individual IQL-queries.

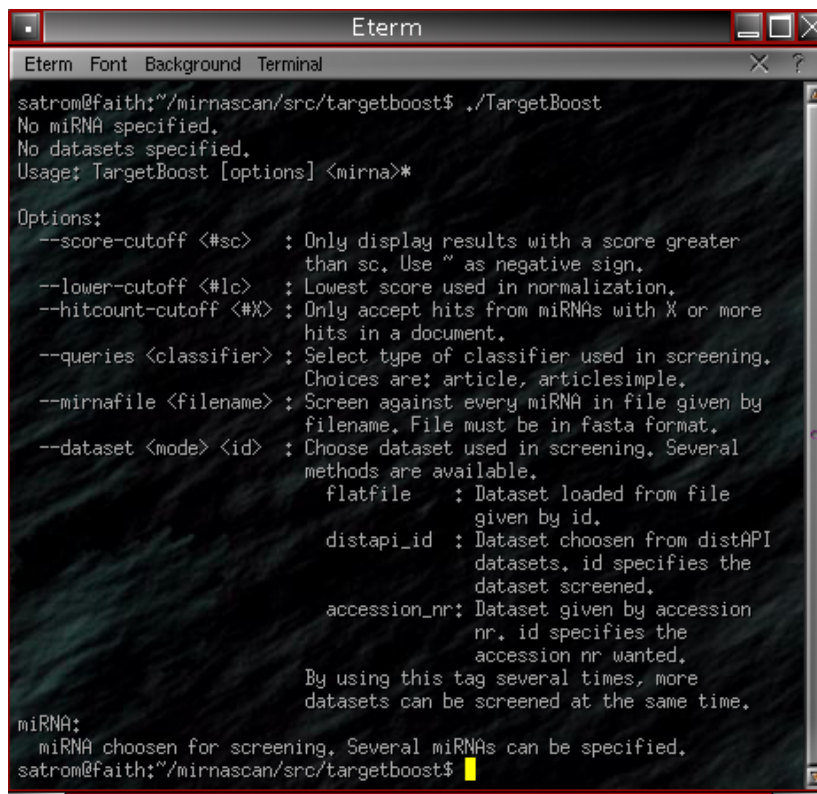
5.2 MicroRNA target-site screener

TargetBoost uses the classifier trained by the GPboost algorithm for finding miRNA target sites. Using TargetBoost, several miRNAs can be screened against several datasets, and different filters can be applied to refine the results.

5.2.1 Program interface

There are two user interfaces to TargetBoost: a command line interface, and a web-based interface. The command line interface can be seen in Figure 5.1, and the web interface can be seen in Figure 5.2.

Results are given in html-format from both interfaces. The main result page can be seen in Figure 5.3. Here, targets with the highest combined regulation score are presented first. Links to sites describing the target genes are generated if possible. A detailed alignment for each target site predicted can also be seen (see Figure 5.4).

The image shows a terminal window titled "Eterm" with a menu bar containing "Eterm", "Font", "Background", and "Terminal". The terminal content shows the execution of the "TargetBoost" command. The output indicates that no miRNA or datasets were specified and provides usage instructions and options. The options listed include: --score-cutoff, --lower-cutoff, --hitcount-cutoff, --queries, --mirnafile, and --dataset. The --dataset option has sub-options: flatfile, distapi_id, and accession_nr. The terminal ends with a prompt for miRNA input.

```
satrom@faith:~/mirnascan/src/targetboost$ ./TargetBoost
No miRNA specified.
No datasets specified.
Usage: TargetBoost [options] <mirna>*

Options:
  --score-cutoff <#sc>  : Only display results with a score greater
                        : than sc. Use ~ as negative sign.
  --lower-cutoff <#lc>  : Lowest score used in normalization.
  --hitcount-cutoff <#X> : Only accept hits from miRNAs with X or more
                        : hits in a document.
  --queries <classifier> : Select type of classifier used in screening.
                        : Choices are: article, articlesimple.
  --mirnafile <filename> : Screen against every miRNA in file given by
                        : filename. File must be in fasta format.
  --dataset <mode> <id> : Choose dataset used in screening. Several
                        : methods are available.
                        : flatfile      : Dataset loaded from file
                        :                : given by id.
                        : distapi_id   : Dataset choosen from distAPI
                        :                : datasets, id specifies the
                        :                : dataset screened.
                        : accession_nr : Dataset given by accession
                        :                : nr, id specifies the
                        :                : accession nr wanted.
                        : By using this tag several times, more
                        :                : datasets can be screened at the same time.

miRNA:
  miRNA choosen for screening. Several miRNAs can be specified.
satrom@faith:~/mirnascan/src/targetboost$
```

Figure 5.1: TargetBoost command line user interface.

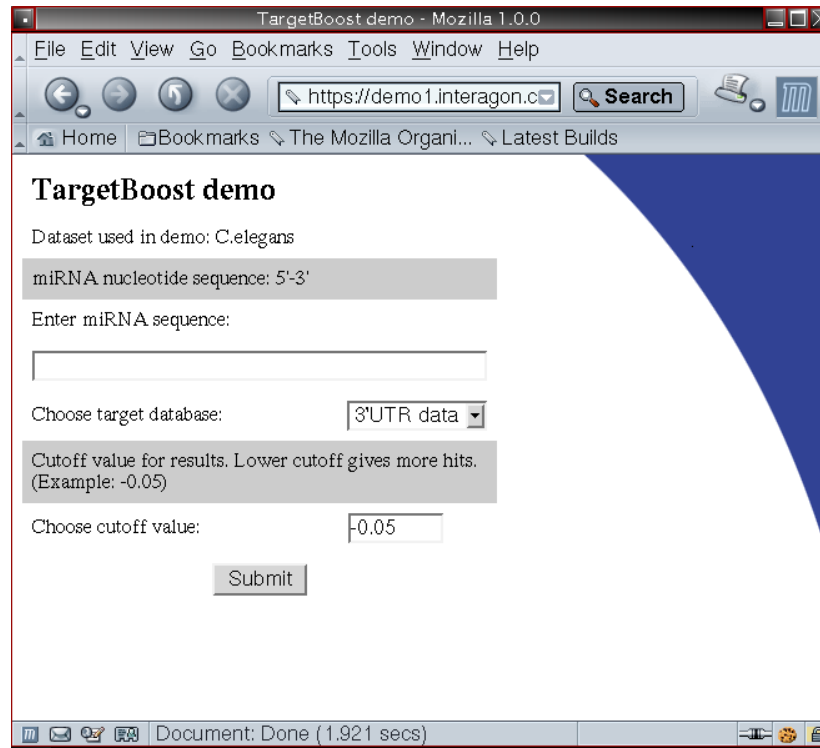


Figure 5.2: TargetBoost web user interface.

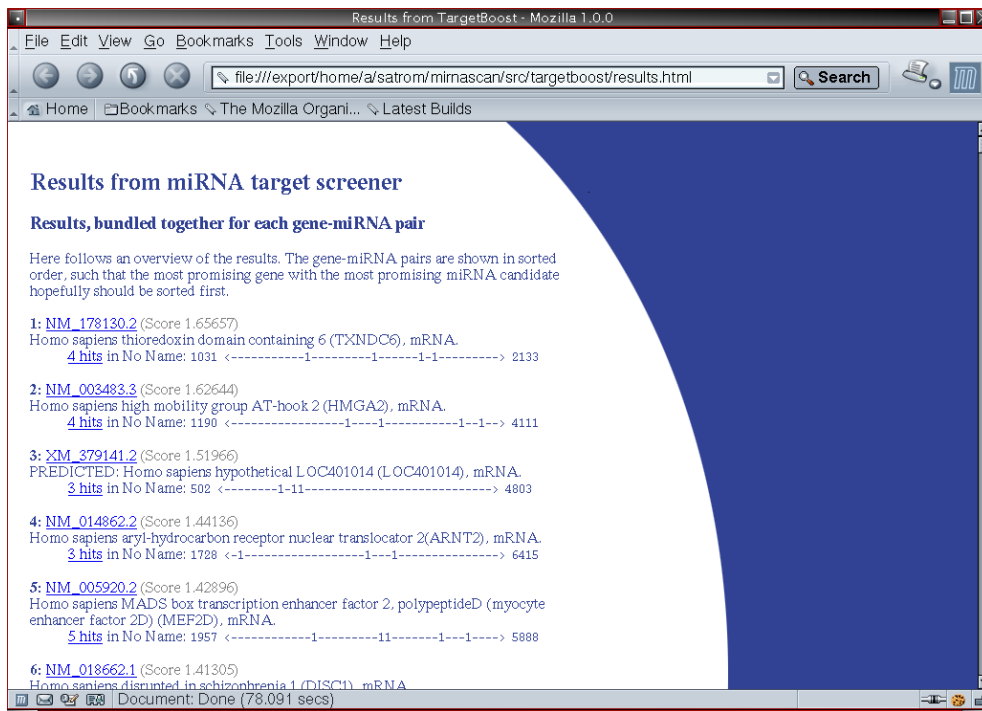


Figure 5.3: TargetBoost main result page.

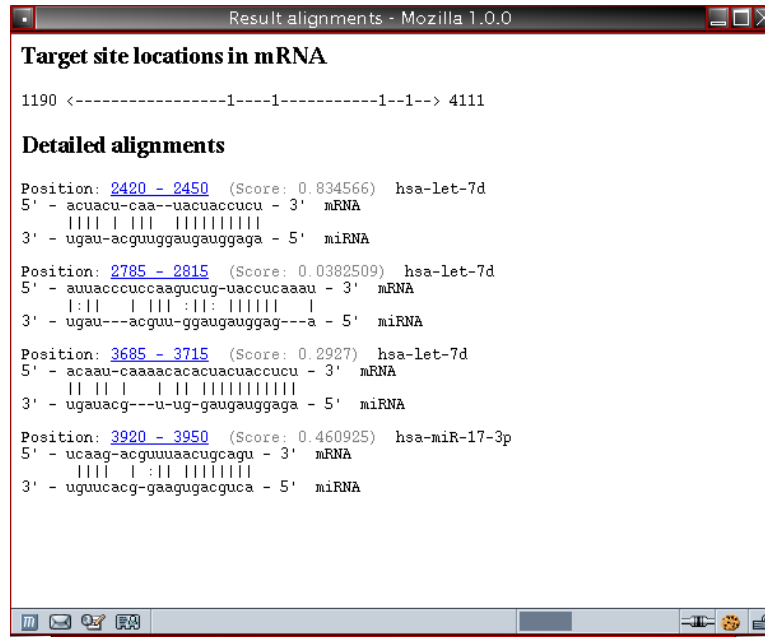


Figure 5.4: Detailed target site alignment.

5.2.2 The TargetBoost algorithm

The TargetBoost algorithm can be seen in Algorithm 3. The input to the algorithm consists of the IQL-queries and their calculated alpha values from the GPboost algorithm. Additionally, TargetBoost take as input the miRNAs and the datasets to be screened. A cutoff value for the target site fitness score can also be defined. Target sites with fitness score below this cutoff are not post-processed and presented to the user.

TargetBoost calculates the “likelihood” of a position in a dataset being a miRNA target-site. This likelihood is hereafter called the position or window fitness. When searching a dataset with a query f_i with a accompanying alpha α_i , a hit in window j should increase the window fitness with α_i , while a miss should decrease the fitness with α_i . After performing a screening, the system reports which dataset window that was hit by the query. Therefore, instead of initializing the result table to 0, and for each query traversing the complete result table and increase or decrease its score according to whether the window was hit or missed, the result table is initialized to its minimum possible value, $MinScore$. For each query, only the query hit list is traversed, and if a window was hit the fitness score of the window in question is increased by $2 \times \alpha_i$.

Before a query f_i can be used in a search, it must be made miRNA-specific. This is done by exchanging the position terminals in the original query with the nucleotide in the specified position in the miRNA to be screened (see Figure 5.5).

-
1. **Input:** Motifs $F = \{f_1, \dots, f_M\}$ with weights $\{\alpha_1, \dots, \alpha_M\}$, Datasets $D = \{d_1, \dots, d_N\}$, miRNAs $S = \{s_1, \dots, s_T\}$, $scoreCutoff$
 2. **Initialize:** $MinScore = -\sum_{i=1}^M \alpha_i$, Create windowed dataset d_i^w for all $i = 1, \dots, N$, Binding site list bsl
 3. **Do for** $n = 1, \dots, N$
 - **Do for** $t = 1, \dots, T$
 - (a) Initialize result table: $r_i = MinScore$ for all $i = 1, \dots, |d_n^w|$
 - (b) **Do for** $m = 1, \dots, M$
 - i. Make query miRNA specific:
 $f_m^{trans} = TranslateQuery(f_m, s_t)$
 - ii. Screen query: $Result = Screen(f_m^{trans}, d_i^w)$
 - iii. **Do for** $p = 1, \dots, |Result|$
 $r_{Result_p} = r_{Result_p} + 2 \times \alpha_m$
 - (c) Generate binding sites: **Do if** $r_i \geq scoreCutoff$: Add d_i^w to binding site list
 4. Print output
-

Algorithm 3: The TargetBoost algorithm

Example Expression

$$\{((P_{14}|P_{14})|((P_{18}|P_{18})|P_5))\{.:r = 10\}:d = 7\}\{P_{17}P_{18}P_{19}P_{20}P_{21}P_{22}P_{23}P_{24}:p \geq 6\}$$

let-7 in reverse-complemented form

a c u a u a c a a c c u a c u a c c u c a

Translated Expression using let-7

$$\{((c|c)|((u|u)|c))\{.:r = 10\}:d = 7\}\{c u a c c u c a :p \geq 6\}$$
Figure 5.5: Translation to a miRNA-specific query. P_i indicates miRNA nucleotide i counted from its 3' end.

After all searches for a miRNA have been performed, the score table is traversed. Windows with score above a threshold, *scoreCutoff*, are treated as predicted target sites. They are added to a list of binding sites, and reported back after all miRNAs and all datasets have been screened.

5.2.3 System overview

The system consists of three main parts: the dataset, the front end, and the screener. The dataset holds the information needed about the dataset, the front end displays the results to the end user, and the screener is responsible for performing the searches. The searching is performed using the PMC (see Chapter 3.4), using the API “distAPI”, an API that allows for distributed searches on several PMCs. An UML-diagram of the program can be found in Appendix A.

PMC-distAPI

TargetBoost uses the PMC-distAPI as an interface against the PMC. As mentioned above, the PMC-distAPI allows for queries on a dataset to be distributed onto several PMCs. The main reason for doing this, is because a single PMC only has 128 MB of internal memory. Datasets greater than 128 MB must therefore be split up, so that each part fits on a single PMC. Using the distAPI, datasets greater than 128 MB are split up, sent to PMCs and searched in parallel. Since results from queries are handled using callbacks, the search results do not need to be collected before being processed. A second reason for using several PMCs becomes apparent when the number of queries increases. The PMC can at maximum search with 64 different queries in parallel. This number drops with the complexity of the queries. By using several PMCs, more queries can be executed in parallel. A third advantage of using the PMC-distAPI is time-saving in search setup. The PMC-distAPI allows for preloading of frequently used datasets, which saves time as the datasets do not have to be loaded into the PMC SDRAM before the searches

Dataset

The dataset’s main responsibility is to load the datasets to be screened into the program. This can be done in several ways: the dataset can be loaded from datasets found preloaded in the PMC-distAPI, it can be downloaded from the NCBI nucleotide database, using the EFetch tool, or it can be loaded from a flat-file (More information about EFetch can be found at http://eutils.ncbi.nlm.nih.gov/entrez/query/static/efetchseq_help.html). Although a dataset

physically is a byte-array, it logically consists of several documents. The documents are separated by document separators.

The classifier was trained using samples 30 nucleotides long. Also, since the final result is calculated from the combined result from many separate IQL-queries, the results from the individual queries must be translated into document positions within the datasets in a consistent way. The logical documents are therefore transformed using a moving window with size 30 and an overlap of 25. This 6-fold enlarged dataset increases the search time, but speeds up and simplifies the post processing for each query.

The dataset module transforms the datasets using the moving window. The module saves the transformation data, so information from searches performed on the enlarged dataset can be translated into information about the original dataset. The enlarged dataset is sent to the PMC-distAPI, and can be preloaded there if this is wanted.

The module consists of the classes `TBDocument` and `TBDataset` in the UML diagram (see Figure A.1 in Appendix A).

Screeener

The screener implements the for-loops used in the TargetBoost algorithm (Algorithm 3). The module's main responsibility is to perform the searches; that is, to make each query miRNA-specific and to send the queries and dataset to the PMC-distAPI. The screener-module also implements the result processor, which implements the callbacks used when processing the results from the PMC.

The queries, which originally are IQL-queries, have been pre-translated into PSL-queries before being loaded into the screener-module. This saves time, because a query does not have to be transformed from an IQL-query to a PSL-query every time it is used in a search.

This module consists of the classes `TargetBoost`, `Configuration`, `WeightedTemplatedEnsembleSearcher`, `Mirna`, and `ResultProcessor` in the UML diagram (see Figure A.1 in Appendix A).

Front end

The front end displays the results from the searches to the end user. In the output, genes with the highest likelihood of being regulated by the miRNAs used in the screening, are presented first, with each miRNA aligned with their predicted target sites in the mRNA. The front end-module is responsible for performing this alignment, using a variant of global alignment. The complete miRNA must

be present in the alignment, but this is not necessary for the target mRNA. The Needleman-Wunsch algorithm for global alignment is used as a basis for the alignment calculations. This algorithm is described in Algorithm 4.

-
1. **Input:** $A = a_1, \dots, a_n, B = b_1, \dots, b_m$
 2. **Initialize:** $F_{i,0} = 0, i = 1, \dots, n, F_{0,j} = 0, j = 1, \dots, m$
 3. **Do for** $i = 2, \dots, n$
 - **Do for** $j = 2, \dots, m$
 - $F_{i,j} = \max((F_{i-1,j-1} + Sa_i, b_j), (F_{i-1,j} + d), (F_{i,j-1} + d))$
 4. Starting in $F_{n,m}$, traceback choices made and create alignment.
 5. **Output:** Aligned sequences.
-

Algorithm 4: Needleman-Wunsch algorithm [5].

The scoring-matrix S used is a standard match matrix that also allows for GU-wobbles in the alignment. Two different gap-penalties d are used: A start penalty and a continue penalty, with the start penalty being the highest.

This module consists of the classes TBBindingSite, BindingSite, DummyQueryFactory, MirnaQueryFactory, BindingSiteList, Document, DocumentIndex, FileDocumentIndex, and FlatFileDocumentIndex in the UML diagram (see Figure A.1 in Appendix A).

5.2.4 Scoring algorithm

The output from TargetBoost should report the likelihood of an mRNA being regulated by the miRNAs used in the search. The likelihood, or scoring, should therefore be uniform regardless of the dataset searched or the number and types of miRNAs used. During a search, each separate target site candidate gets an individual score, reflecting its likelihood of being a real target site. The scores of the individual target sites are then combined in a final scoring algorithm, whose output reflects the likelihood of an mRNA target being regulated by the miRNAs used in the search.

Scoring of target site candidates

Every target site candidate get a numerical score that is the weighted sum of the classifier queries that matches the site in question. This sum is dependent on the miRNA used in the screening, and is especially dependent on the miRNA size. The architecture used when creating the classifier allows for miRNAs with up to 24 nucleotides to be used. During training, miRNAs shorter than 24 were appended with wildcards on the 3' end so the complete length became 24. This,

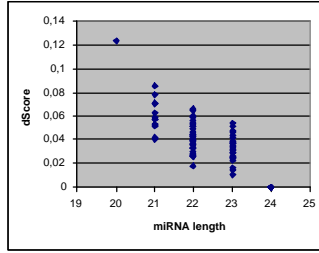


Figure 5.6: Correlation between miRNA length and target site score in the original procedure for making queries miRNA-specific.

however meant that target site candidates for shorter miRNAs got a higher score than longer miRNAs, since wildcards matches every nucleotide (see Figure 5.6).

The size of the human miRNAs range from 17 to 25 nucleotides long, with most being 21 to 22. Few have less than 21 nucleotides. In order to get a more uniform score from every miRNA, only the 21 first nucleotides counted from the 5' end are used in the screening. Three wildcards are appended on the 3' end to make the final length 24. This allows for a more uniform scoring of miRNAs of size 21 and longer, which means most miRNAs. Shorter miRNAs will still get a higher score, but this affects only 21 of 321 reported human miRNAs. If fewer than 21 nucleotides are used, it could also mean that some interaction between mRNA and the miRNA 3'end are overlooked.

Even though some of the variance in the scoring of each individual target site has been removed by only using the first 21 nucleotides, some variation is still present. A normalization of the scoring is therefore necessary. Additionally, a scoring between 0 and 1 makes the individual scores easier to use in later stages. The normalization function used is given in Equation 5.2.

$$Score = \frac{Score - LB}{UB - LB} \quad (5.2)$$

LB and UB is the lower bound and upper bound used in the normalization. Only target site candidates above the score cutoff are normalized, and the cutoff is therefore used as the lower bound. The upper bound is normally the maximum score reported for the miRNA. The problem with this approach appears when several miRNAs are used in a search. One miRNA might regulate the target mRNA, and therefore, rightfully get a high score both before and after the normalization. A second miRNA might not regulated the mRNA and therefore get a low score. If the maximum score for this miRNA is used as an upper bound in the normalization process, the miRNA's score will be artificially enlarged compared to the scores of the first miRNA. In other words, the miRNAs' scores will

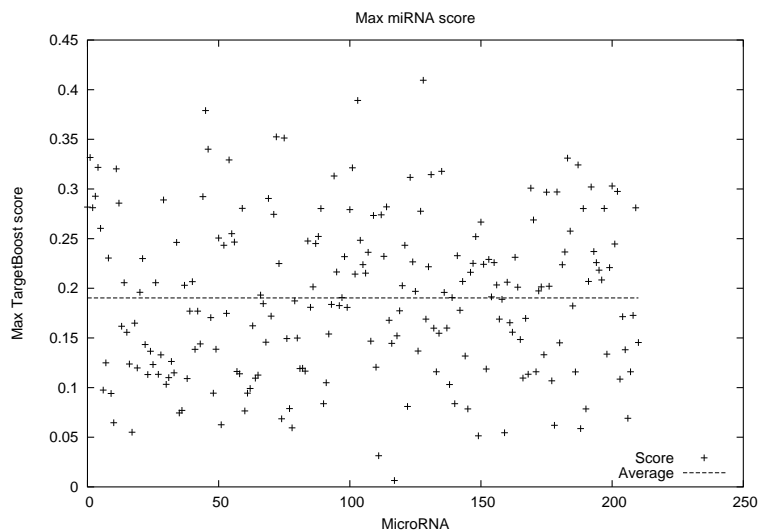


Figure 5.7: Maximum TargetBoost score for every human miRNA in the human genome.

be indistinguishable. A minimum upper bound is therefore used in the normalization. This was found by screening every human miRNA against the 3'UTR region of every gene in the human genome. The maximum score reported for every miRNA were found, and the average was used as the minimum upper bound (see Figure 5.7).

Combined scoring algorithm

For each document in the dataset searched, TargetBoost outputs the likelihood of the document being regulated by the miRNAs used in the search. The scoring algorithm should calculate this likelihood independent of the number of miRNAs used in the search, the number of target site candidates found in the document, and the document in question. These factors are very variable, because TargetBoost allows both for screening a single gene using a single miRNA, and for screening the complete human genome using every miRNA expressed in humans.

The simplest scoring scheme is to use the likelihood of the best target site candidate as the likelihood of regulating the mRNA (see Equation 5.3. \mathbf{B} is the set of binding sites within the document i). This method is overly simplistic since it completely disregards the existence of multiple target sites in the mRNA, or the cooperation of several miRNAs in the mRNA regulation.

$$F(d_i) = \max(S(\mathbf{B})) \quad (5.3)$$

A slightly more advanced scoring scheme is to use the sum of every target score reported as the final score (see Equation 5.4). Using this measure, the occurrence of multiple target sites will be taken into account. There several problems with this scoring scheme. The range of the final score is unbounded, and highly dependent on the number of binding sites $|\mathbf{B}|$. This means that the final score will be very different when comparing the the output from searches using few miRNAs, and searches using many miRNAs.

$$F(d_i) = \sum_{j=1}^{|\mathbf{B}|} S(b_j) \quad (5.4)$$

Since the method does not preprocess the binding sites in the list, binding sites from different miRNAs may overlap. Overlapping binding sites can not cooperate in regulating a target, especially if the seed region is overlapping [12]. Summing the scores of overlapping binding sites therefore makes no biological sense. Algorithm 5 removes binding sites overlapping other binding sites with a higher score, finding an optimal set of binding sites for regulating the mRNA. The sum of these non-overlapping sites are then used as the final score.

-
1. **Input:** A set of binding sites B , allowed overlap l .
 2. **Initialize:** Score list S
 3. Sort binding site by their binding site address in the document.
 4. **Do for** each address in binding site list, find binding site with highest score. Add this to the list B_m .
 5. **Do for** each binding site b_i in B_m ,
 - (a) **Do for** each s_j in S starting at the end,
 - **Do if** $Addr(s_j) + l \leq Addr(b_i)$, set $Score(b_i) = Score(b_i) + Score(s_j)$, $Point(b_i) = s_j$, break loop
 - (b) If none are found, set $Point(b_i) = \emptyset$
 - (c) **Do if** $Score(b_i) \geq Score(\text{last item in } S)$, append b_i to S
 6. Starting with s_N at the end of S , follow $Point(s_n)$ until \emptyset is reached, appending the s_n 's traversed to the result list R
 7. **Return** result list R .

Algorithm 5: Optimal binding site set finder

The output from the previous scoring scheme is still dependent on the number of miRNAs used in the search, since it is directly connected to the number of target site candidates predicted by the classifier. A simple solution is to use a cutoff-value on the output, saying scores above indicates target regulation (final score 1), and scores below does not (final score 0).

Comparing the score-algorithms

The efficacy of the different scoring algorithms are measured by computing the signal to noise ratio between searches using real miRNAs (the signal) and searches using shuffled miRNAs (the noise). The real miRNAs selected are the set of human miRNAs with a distinct seed region (nucleotides 2 to 8 counted from the 5'end). From the set of human miRNAs, we found 144 distinct seed regions. When several miRNAs had the same seed region, the miRNA representing that seed region were randomly selected and added to the miRNA set. The miRNAs selected are shuffled using dinucleotide shuffling [35], thereby preserving their dinucleotide distribution. The average of 10 different shuffled miRNAs is used as noise. The signal to noise ratio is calculated using Equation 5.5.

$$S/N = \frac{|Score(miRNA_{real}, d_i) \geq X|}{|Score(miRNA_{shuffled}, d_j) \geq X|}, d_{index} \in D \quad (5.5)$$

The equation calculates the ratio between the number of documents in the dataset with a score greater than a cutoff value X using the real miRNAs and using the shuffled miRNAs, respectively.

Chapter 6

Results

We compared the predictive power of TargetBoost with the efficiency of two other published algorithms for detecting miRNA-target site candidates, namely Nucleus [10] and RNAhybrid [11]. In the comparison, the different algorithms ability to discover experimentally verified target sites were measured. Here, the TargetBoost classifier was found to be as efficient and better as Nucleus and RNAhybrid [36]. The comparison, however, was based on single target sites, and the ability of a miRNA to regulate an mRNA based on single target sites. Additionally, several target sites from several miRNAs can cooperate to regulate mRNAs. A comparison of different scoring schemes for combining targets into a final score were therefore performed.

First we give a short summary of the results published in [36]. This includes the main results from the algorithm-comparisons. Additionally, some results not published in the article regarding the TargetBoost classifier are also mentioned. Last, the results from the comparison between the different scoring algorithms are presented. For the full article about TargetBoost, see Appendix B.

6.1 The TargetBoost classifier is a state of the art miRNA target site predictor

We compared the TargetBoost classifier against the algorithms Nucleus and RNAhybrid. The comparison was performed using cross-validation and receiver operating characteristics (ROC) analysis on the dataset used when training the TargetBoost classifier (see Chapter 5.1.2 for details on the dataset).

Figure 6.1 shows the overall ROC-curves for the algorithms. It can be seen that TargetBoost has the best performance at specificity levels above 0.9. Figure 6.2

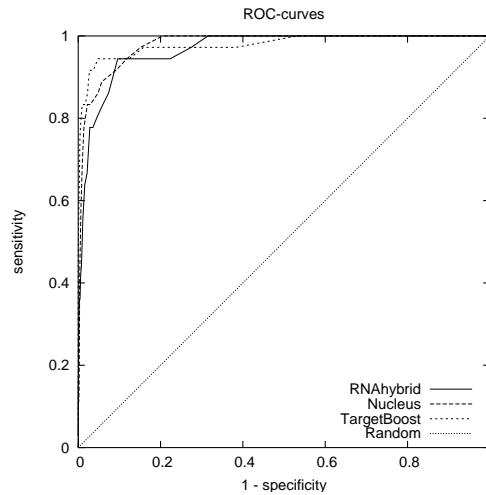


Figure 6.1: Overall ROC-curves for algorithms.

shows the algorithm-performance on each separate miRNA in the dataset. For each miRNA, TargetBoost either performed best, or was not significantly worse than the other algorithms. This was confirmed both when running true-positive frequency tests (a test that compares the sensitivity of two classifiers at a given specificity level), and when computing ROC_{50} -scores for the algorithms.

The 5' end of miRNAs tend to bind perfectly to its target. Most algorithms (including Nucleus and RNAhybrid in an indirect way), uses this property when predicting target sites. TargetBoost rediscovered this property during the classifier training: the consecutive sequence in most patterns generated in the first boosting iteration, consisted of position 17 to 24, that is, nucleotides 1 to 8 counted from the 5' end (see Chapter 5.1.1 for more information about the consecutive sequence).

6.2 Efficacy comparison of different architectures

We created three different architectures or grammars, and used them in the training of the TargetBoost-classifier. These were called “Displaced-Ordered”, “Simple-Ordered”, and “Displaced-Pnofm” (see Chapter 5.1.1 for detailed descriptions of the different architectures). The basic structure of the architectures are the same, but there are differences in the strictness of the grammars. A strict grammar can only create a subset of the expressions created by other grammars. This gives the genetic programming process more freedom when creating solution proposal, but also increases the search space. The order of strictness with the least

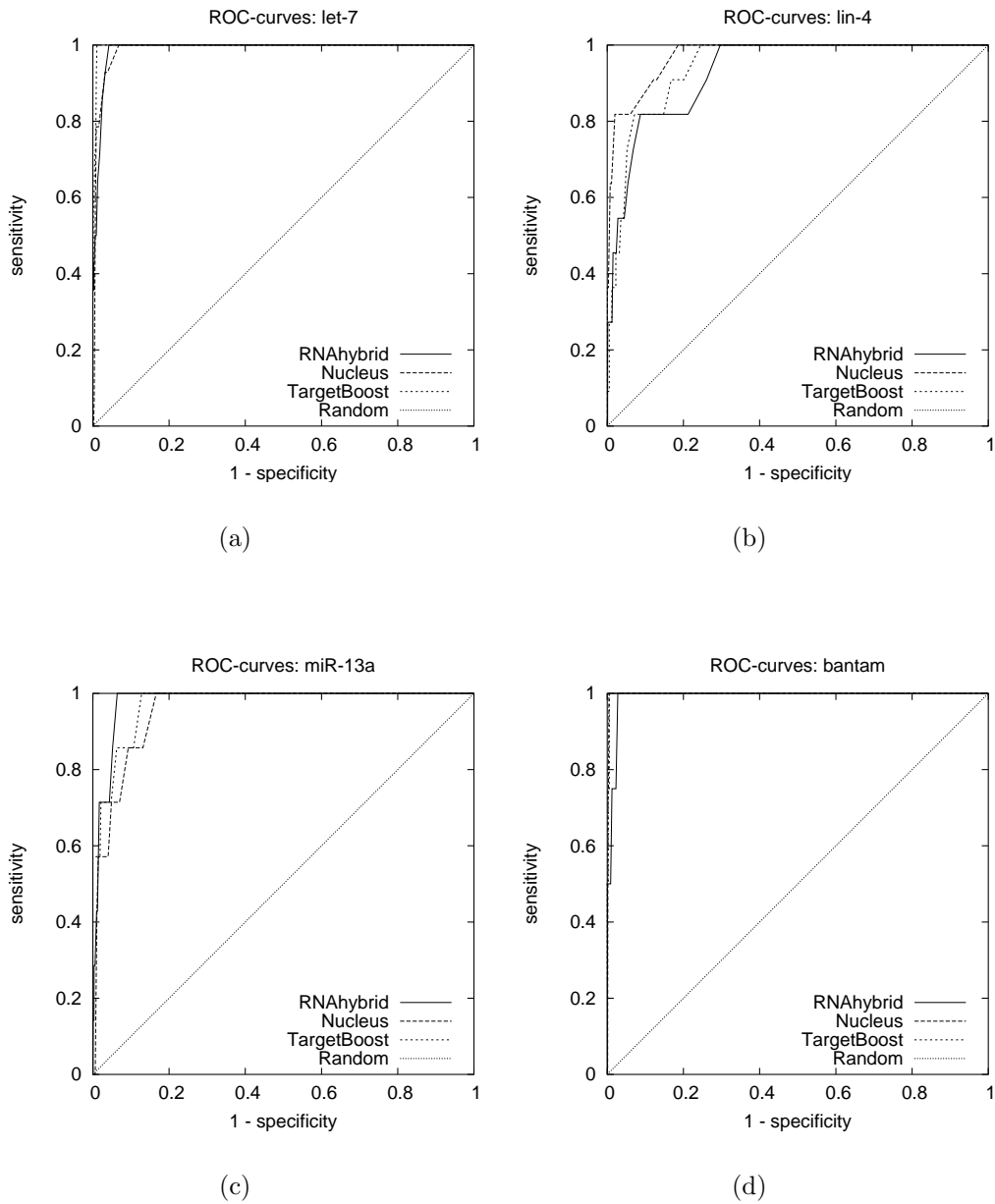


Figure 6.2: Performance comparisons on individual miRNAs.

strict grammar first, are as follow: “Displaced-Pnofm”, “Displaced-Ordered”, and “Simple-Ordered”. Figure 6.3 shows the ROC-curves for the different classifiers created using the architectures.

Too little strictness in the grammar increases the difficulty of finding the solution in the TargetBoost algorithm. “Displaced-Pnofm” has the lowest ROC-score for each miRNA, most often performing as bad as a random classifier (Figure 6.3(d), 6.3(b), and 6.3(c)). “Displaced-Ordered” and “Simple-Ordered” on the other hand, have comparable efficiencies: for miR-13a (Figure 6.3(c)) is “Displaced-Ordered” best, and for lin-4 (Figure 6.3(b)) “Simple-Ordered” is best. When comparing the performance on let-7 and bantam, their performance are almost equal, with “Displaced-Ordered” best in the high-specificity region of bantam, and “Simple-Ordered” best in the same region for let-7.

The comparable efficacies of “Displaced-Ordered” and “Simple-Ordered” indicate that TargetBoost discovers the binding pattern between miRNA and mRNA on the miRNA 3’end using the smaller expressions. This can be contributed to the boosting part of the TargetBoost classifier training: the complex pattern is generated by a weighted combination of many simple patterns.

6.3 The average of a set of TargetBoost classifiers creates a better final classifier

The Genetic Programming process is a stochastic process: individuals to be used in the evolutionary process and the genetic operator, are selected randomly from the the population. The exact same solution is therefore rarely found in two separate GP-runs. It can be shown that a set of diverse classifiers can be combined into a better classifier [34]. TargetBoost uses the average of 10 different classifiers as the final classifier.

The final classifier achieved higher specificity-levels for high sensitivity levels, especially in the high specificity region. This is wanted because it reduces the false positive count. Figure 6.4 shows the ROC-curves for each single classifier compared to the averaged classifier for the miRNA let-7. Since the classifiers were created using a stochastic process, there will be some differences in how they score target candidates, especially false target candidates. This randomness is removed by averaging across all the individual classifiers. Therefore, while the false-positive count increases for each of the single classifiers, it is held low in the averaged classifier.

6.3. THE AVERAGE OF A SET OF TARGETBOOST CLASSIFIERS CREATES A BETTER FINAL

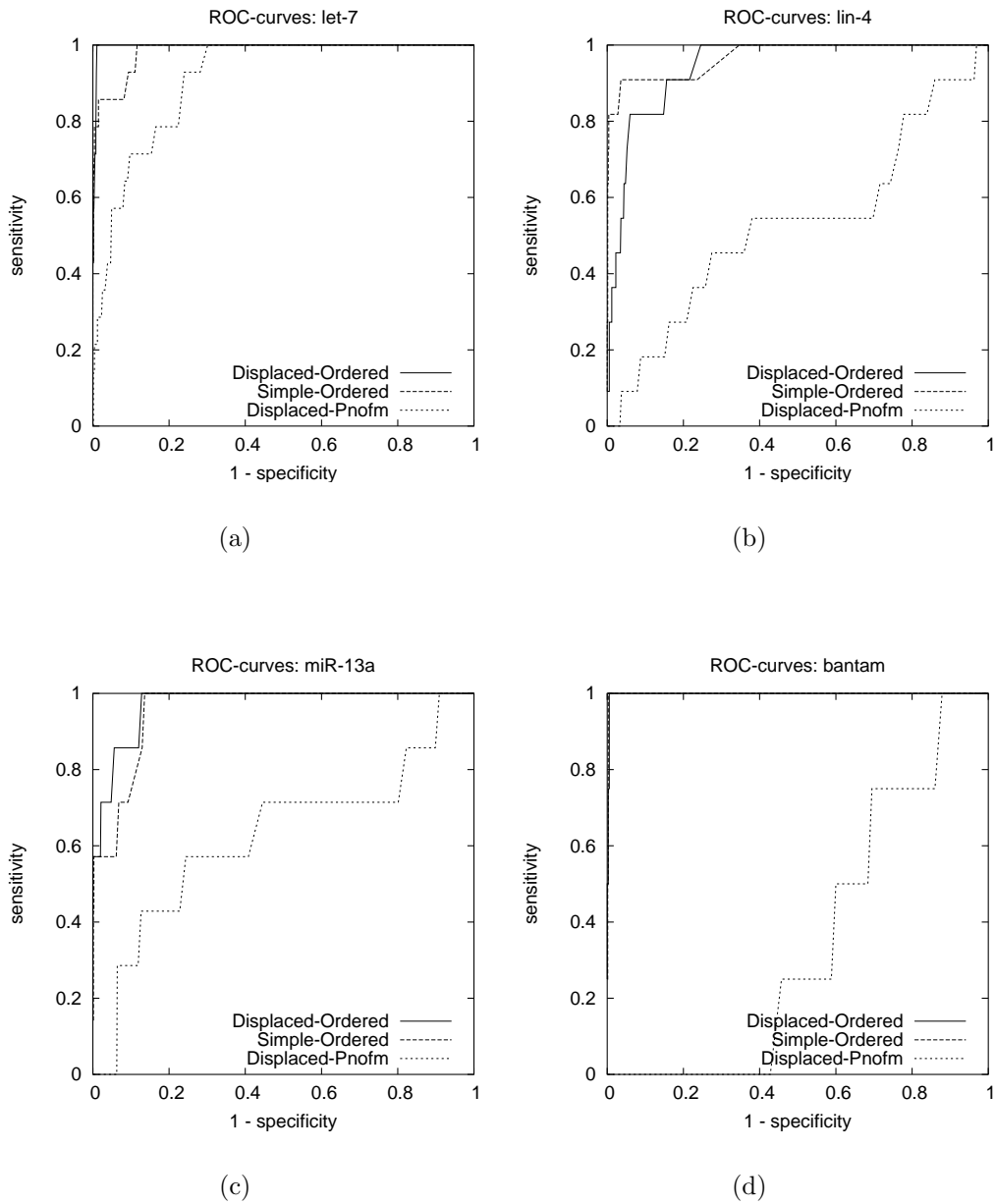


Figure 6.3: Performance comparison for different grammars.

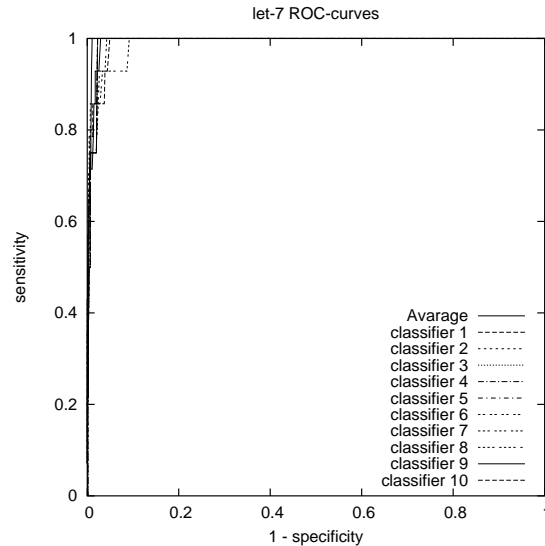


Figure 6.4: Efficacy comparison of individual classifiers vs average classifier for miRNA let-7.

6.4 The TargetBoost scoring is best for finding single mRNA-regulating target sites

Even if a miRNA can not regulate an mRNA by itself, it might still contribute to mRNA regulation through cooperative effects with other miRNAs. When using several miRNAs in a search, the final score should therefore reflect the likelihood of the target being regulated by the miRNAs used in the search. Based on the individual target sites predicted by the TargetBoost classifier, we developed several scoring functions for multiple target site regulation and compared their efficacies: “MaxScore”, “SumScore”, “SumScoreDivLength”, and “Sigmoid”. The scoring functions are described in Chapter 5.2.4. In the comparison, we used the scoring functions on both target sites predicted by TargetBoost, and on target sites predicted when using miRNA seed regions. In the latter method, target sites are predicted when the reverse complement of nucleotides 2-7 is found in the target mRNA. Such predicted target sites get a score of 0.8.

In the comparison, we screened the 3’UTR region of 2000 randomly selected human genes against the human miRNAs. To prevent seed region bias, each miRNA used in the screening had a distinct seed region, removing 66 of 211 miRNAs from the set. We also screened the dataset using 10 shuffled versions of the miRNAs. These were created using dinucleotide shuffling [35], preserving the dinucleotide distribution of the original miRNAs. As a scoring metric, we

6.4. THE TARGETBOOST SCORING IS BEST FOR FINDING SINGLE MRNA-REGULATING TA

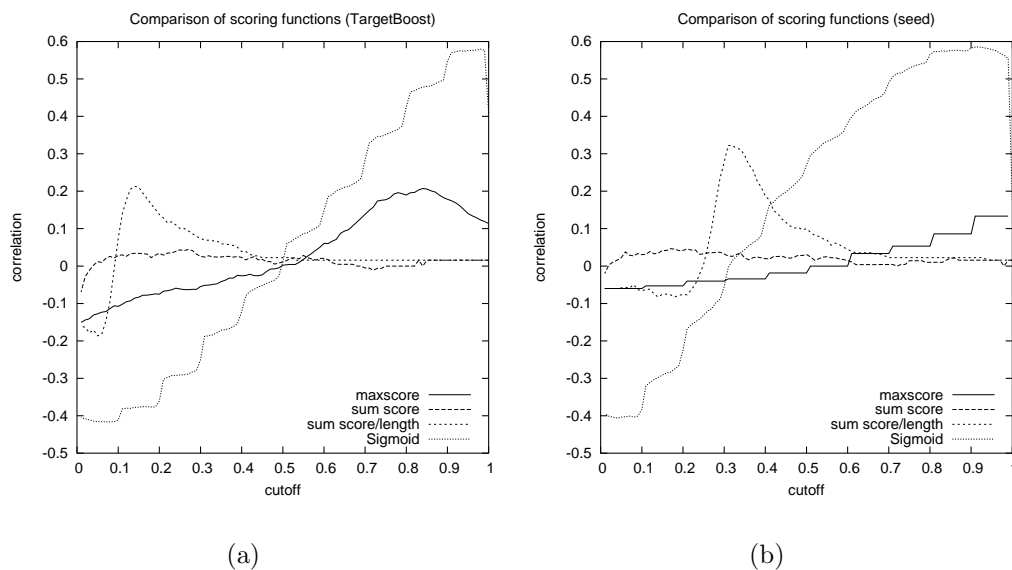


Figure 6.5: Correlation curves for scoring functions. (a) TargetBoost predicted target sites. (b) Seed predicted target sites.

counted the number of genes with a final score above a given scoring threshold c :

$$TP = |\text{Score}(\mathbf{miRNA}_{\text{real}}, d_i) \geq c| \quad (6.1)$$

$$FP = |\text{Score}(\overline{\mathbf{miRNA}_{\text{shuf}}}, d_i) \geq c| \quad (6.2)$$

$$FN = |\mathbf{d}| - TP \quad (6.3)$$

$$TN = |\mathbf{d}| - FP \quad (6.4)$$

Figure 6.5 shows the correlation curves for the different scoring functions, and Figure 6.6 shows the ROC-curves for the same scoring functions.

Comparing the performance of “MaxScore” on target sites predicted by TargetBoost and seed, we see that TargetBoost has a larger correlation than seed. The training set used when creating the TargetBoost classifier consisted of experimentally verified target sites that individually regulate the target mRNA. TargetBoost is therefore especially trained for finding these sites, which means highscoring target site candidates have a high likelihood of individually regulating

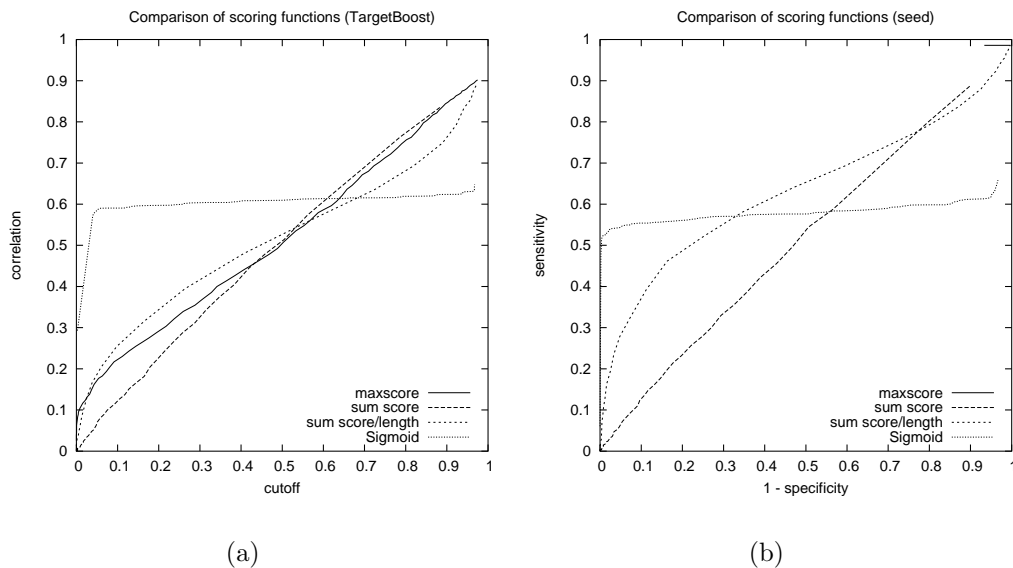


Figure 6.6: ROC curves for scoring functions. (a) TargetBoost predicted target sites. (b) Seed predicted target sites.

the target mRNA [36]. The probability of these sites occurring randomly is low, and this can be seen by the “MaxScore” correlation for TargetBoost. The probability of a sequence of 6 nucleotides occurring randomly in a sequence is much higher. Individually occurring seed sites is therefore a bad estimate for regulation compared to individually occurring TargetBoost sites. This can also be seen in the ROC-curve for “MaxScore” (Figure 6.6(b)): a specificity larger than 0.07 could not be found.

Summing the score of the predicted target sites is useless without regarding the target dataset length. Statistically one can expect a long mRNA to have more target sites than a short mRNA. This was confirmed when plotting the sum of the target site scores as a function of mRNA length (see Figure 6.7). When using “SumScoreDivLength”, we therefore see an increased efficacy compared to when using “SumScore”. This is because we now identify mRNAs with a density of target sites above average. For seed-predicted target sites, we see a much higher max-correlation for “SumScoreDivLength” compared to “MaxScore”. We do not find this increase for TargetBoost predicted target sites. While the correlation increases from 0.12 to 0.31 for seed target sites, it lies steadily on 0.22 for TargetBoost target sites.

The only difference between “Sigmoid” and “SumScoreDivLength” is the transform function between inputspace and outputspace. It therefore does not represent

6.5. THE TARGETBOOST CLASSIFIER SPECIALIZES IN FINDING 3' COMPENSATORY TARGET

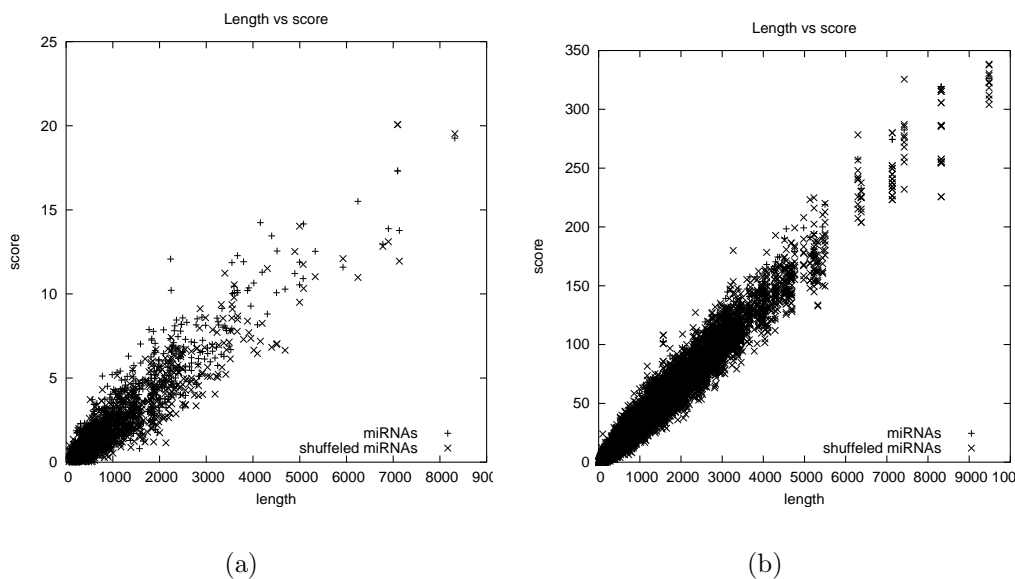


Figure 6.7: Score vs document size when summing target site scores using both (a) TargetBoost predicted target sites, and (b) seed predicted target sites.

a unique scoring method, but can be seen as a post-process for “SumScoreDivLength”. The effect is to smooth the differences found in the “SumScoreDivLength”.

TargetBoost is useful for finding target sites that regulated mRNA targets individually (see Figure 6.5, and [36]). But, compared to seed predicted target sites, we see no increase in the efficacy when combining target sites predicted by TargetBoost. This indicates that the scoring metric from the TargetBoost site is not suitable to predict combinatorial effects in mRNA regulation.

6.5 The TargetBoost classifier specializes in finding 3' compensatory target sites

Lim et al. [37] transfected the miRNAs miR-1 and miR-124 into human cells, and examined which genes that were regulated. They found 96 genes downregulated by miR-1 and 174 genes downregulated by miR-124. When checking the downregulated genes, they found miR-1 seed regions in 88% of the genes regulated by miR-1, and miR-124 seed regions in 76% of the genes regulated by miR-124. To compare the type of target sites reported by TargetBoost and the types reported by seed-regions, we screened the 3'UTR region of the human genome for miR-1

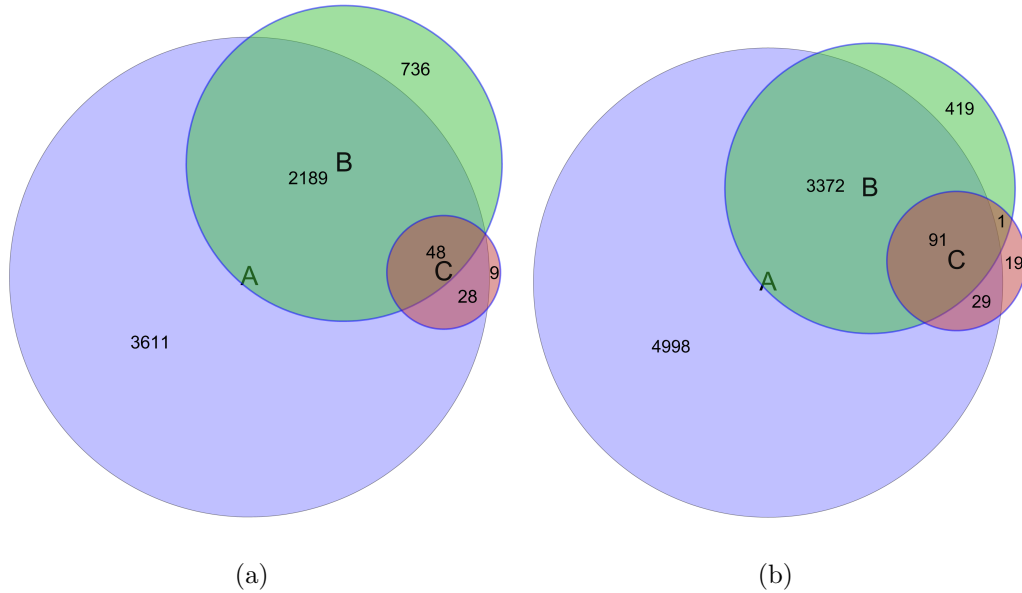


Figure 6.8: Venn-diagrams comparing the results when screening (a) miR-1 and (b) miR-124, using TargetBoost and seed-region checking. **A** is the set of genes predicted to be regulated using seed-region checking, **B** is the set of genes predicted using TargetBoost, and **C** is the genes found to be regulated by the miRNAs.

and miR-124 target sites using both TargetBoost and seed-region checking. Figure 6.8 shows the Venn diagrams comparing the target sites reported by the two methods.

We can see that the set of regulated mRNAs is small compared to the set of mRNAs predicted to be regulated. Failure to capture all regulated mRNAs in the experiment is one possible explanation. Lim et al. used stringent P -values when deciding which genes that were regulated, so they might miss some regulated targets [37]. A larger part of the predicted targets might therefore be regulated. A second explanation is regulation using cooperative effects: miR-1 and miR-124 is not able to regulate the targets by matches with their seed region. Only by cooperating with miRNAs naturally expressed in the cells are the genes regulated by miR-1 and miR-124. This would explain why TargetBoost only predicts regulation in 56% of the regulated genes for miR-1 and in 66% of the regulated genes for miR-124. TargetBoost specializes in finding target sites that individually can regulate a mRNA (see Figure 6.5(a) and [36]), and will therefore not predict regulation when weak target sites regulate mRNAs through cooperative effects with other miRNAs.

Perfect seed match, while it helps, it is not a requirement for mRNA target

6.5. THE TARGETBOOST CLASSIFIER SPECIALIZES IN FINDING 3' COMPENSATORY TARGET

regulation. The experimentally confirmed miRNA-mRNA pairs lin-4/lin-14 (see Figure 2.3) is one example of target regulation without a perfect seed. TargetBoost does not require a perfect match, and can therefore also find these target sites. This means TargetBoost is not a subset of seed-region checking, which can be seen in the venn-diagrams.

TargetBoost does not predict regulation in every gene with a seed-region. The TargetBoost classifier was trained on target sites with matches in the 3' end of the miRNA, that is, 3' compensatory sites (see Chapter 2.4). TargetBoost will therefore not predict regulation in target sites with perfect seed match and little or no match in the 3' end. This explains why TargetBoost only predicts regulation in a subset of the mRNAs with perfect seed matches.

Chapter 7

Discussion

7.1 Scoring function

We compared the performance of TargetBoost with simple seed-region checking on both synthetic and real data, and found that TargetBoost is most suitable for finding target sites that have a high likelihood of regulating a target mRNA. Combining the individual scores gave no improvement compared to when combining several seed regions (see Figure 6.5). The TargetBoost classifier is trained for finding sites with matches in the 3' end of the miRNA, and a higher degree of complementarity between the miRNA and mRNA means a higher chance of regulation [9]. Simple combination of these site-scores did not increase the information content.

Seed-region checking is more suitable when searching for cooperative effects in mRNA regulation. While a single seed-site within an mRNA gave little indication of regulation, the occurrence of several sites increased the probability of regulation (see Figure 6.5(b) and Figure 6.6(b)). This indicates that, while the occurrence of a single seed-target site is not enough to predict regulation, cooperative effects between several seed target sites is a good method for predicting regulation.

7.2 Dataset handling

As mentioned in Chapter 5.2.3, the datasets that are to be searched are first transformed using a moving window size of 30 with an overlap of 25. This increases the data that must be searched, which again increases the search time. The PMC is configured to report back the window number of each window that had a match during the search. This is not the only possible configuration of

the PMC: it can also be configured to report back the address where the the PMC found a match. In principle, if this configuration is used, the dataset to be searched does not need to be transformed first.

The main advantage of not transforming the dataset is the smaller size of the dataset to be searched. Smaller data size means shorter search times. It was found that the search time of the enlarged dataset was four times higher than the original dataset (the dataset size was increased by a factor of 6). In this comparison, the postprocessing of the results is not taken into account.

The disadvantage of using the original dataset in the search, is the increased complexity of the result postprocessing. The classifier is the weighted sum of 250 separate queries. The matches reported from the PMC from each query must therefore be mapped back to the dataset in a consistent way. This is easily done in a windowed dataset when the PMC reports back the window numbers where matches occur: a window number maps to a window number in the dataset. This is more complex when the PMC reports the address.

One possible way doing this mapping is to use the concept of a windowed dataset logically. This might initially seem like a good idea, because an address can (with some work) be mapped to a dataset window. This will require finding the correct document in the dataset, and then finding the correct window within this document. The document can be found by using a binary search in an array with start addresses for each document in the dataset. Having the correct document, an address can be directly translated into windows. However, there is one problem with this approach: depending on the physical length of a query, it might be mapped to one to six different windows. This is because a window is 30 nucleotides long, and there is an overlap of 25 nucleotides between neighboring windows. This means a query of length less than five can be mapped to six different windows, while a query of length 28 can only be mapped to a single window.

Another possible method is to “create” the logical window in an address position when a match is reported in an address. As more and more matches are reported, the list of already created windows are checked to see if the the match belongs to one or more of the windows. If not, a new window is created. If a new window is created which overlaps with an already created window, matches reported in the previous window must be checked to see if the also belong to the new window. This requires a data structure that is easy to traverse and modify. Query size is also here an important factor.

The main reason for not implementing any of the address-based methods, is their dependence on the query-size. For simple queries, this can easily be calculated, but for complex queries the size calculation becomes more difficult. Queries created with the production rules described in Table 5.1, have a variable size

because of the linger function (see Chapter 5.1.1 for more information). This means the number of windows these queries can be mapped to is unknown and sequence specific. For instance, the query

$$\{(a|u)uc\{. : r = 2\} : d = 20\}\{uug : p \geq 2\}$$

have a minimum size of 8 and a maximum size of 28, and can therefore be mapped to any number of windows between. The actual size is dependent on the distance between the $(a|u)uc\{. : r = \}$ -expression match and the $\{uug : p \geq 2\}$ -expression match, and is therefore different from target site to target site. No method for automatically finding the actual size exists. A possible solution is to train a classifier that uses a fixed displacement or to use the “Simple-Ordered” architecture, but this is not a relevant solution here. The grammar in Table 5.1 was published in [36], and TargetBoost must therefore implement this grammar.

Chapter 8

Further work

The main task in further work will be combining the TargetBoost-predicted target sites with target sites predicted using other, simpler methods, such as seed region checking. Since combining seed-region target sites gives a better estimate for predicting cooperative effects (see Figure 6.5(b) and [23]), and TargetBoost-predicted target sites are excellent for finding regulating target sites (see Figure 6.5(a) and [36]), a scoring algorithm that combines both types of target sites should give better regulatory predictions. This scoring algorithm could also incorporate distance between target sites as a parameter.

Other tasks would be check for p-value variance in the microarray data of Lim et al. [37]. Genes regulated by a single miRNA are probably less regulated than genes regulated by cooperative effects [38]. They will therefore probably get a lower p-value in Lim et al. An interesting study would be to vary the p-value in the microarray data and performing the study done in Chapter 6.5.

Appendix A

TargetBoost system

The UML-diagram of the TargetBoost is depicted by Figure A.1.

Appendix B

Article

Weighted sequence motifs as an improved seeding step in microRNA target prediction algorithms

OLA SÆTROM,¹ OLA SNØVE JR.,² and PÅL SÆTROM²

¹Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway

²Interagon AS, Medisinsk teknisk senter, NO-7489 Trondheim, Norway

ABSTRACT

We present a new microRNA target prediction algorithm called TargetBoost, and show that the algorithm is stable and identifies more true targets than do existing algorithms. TargetBoost uses machine learning on a set of validated microRNA targets in lower organisms to create weighted sequence motifs that capture the binding characteristics between microRNAs and their targets. Existing algorithms require candidates to have (1) near-perfect complementarity between microRNAs' 5' end and their targets; (2) relatively high thermodynamic duplex stability; (3) multiple target sites in the target's 3' UTR; and (4) evolutionary conservation of the target between species. Most algorithms use one of the two first requirements in a seeding step, and use the three others as filters to improve the method's specificity. The initial seeding step determines an algorithm's sensitivity and also influences its specificity. As all algorithms may add filters to increase the specificity, we propose that methods should be compared before such filtering. We show that TargetBoost's weighted sequence motif approach is favorable to using both the duplex stability and the sequence complementarity steps. (TargetBoost is available as a Web tool from <http://www.interagon.com/demo/>.)

Keywords: miRNA target prediction; genetic programming; boosting; machine learning

INTRODUCTION

MicroRNAs (miRNAs) belong to an abundant class of short noncoding RNAs (Lagos-Quintana et al. 2001; Lau et al. 2001; Lee and Ambros 2001) shown to mediate suppression of protein translation (Moss et al. 1997; Olsen and Ambros 1999; Reinhart et al. 2000) and cleavage of mRNA (Zeng et al. 2002; Yekta et al. 2004). Homologs exist across many species (Pasquinelli et al. 2000), which shows that miRNAs' function as gene regulators has been conserved through evolution. A total of 1340 miRNA genes from 11 species are listed in the 5.0 release of the miRNA registry (Griffiths-Jones 2004). Computational approaches have estimated that about 1% of all predicted genes in the human (Lim et al. 2003a), fruitfly (Lai et al. 2003), and worm (Lim et al. 2003b) genomes are miRNA genes.

It seems that miRNAs function as siRNAs and silence genes by mRNA cleavage when targets with near-perfect complementarity exist (Zeng et al. 2002; Yekta et al. 2004),

whereas inhibition of translation occurs when miRNAs are only partially complementary to their targets (Lee et al. 1993; Wightman et al. 1993). MicroRNAs known to induce translational suppression predominantly target 3' UTRs (Bartel 2004) with neighboring binding sites (Olsen and Ambros 1999; Reinhart et al. 2000), but it has been demonstrated that a siRNA targeting a single coding site with partial complementarity can induce translational suppression as well (Saxena et al. 2003). Regardless, the inhibition of protein synthesis is more effective when targeting multiple sites (Doench et al. 2003).

Several miRNA target prediction algorithms have appeared recently, and results for fruitfly (Enright et al. 2003; Stark et al. 2003; Rajewsky and Socci 2004) and mammals (Lewis et al. 2003; John et al. 2004; Kiriakidou et al. 2004) suggest that about 10% of protein-coding genes are regulated by miRNAs (John et al. 2004). Computational approaches for identifying miRNA targets generally use sequence complementarity, thermodynamic stability calculations, and evolutionary conservation among species to determine whether a miRNA:mRNA duplex is a likely target interaction (Bartel 2004; Lai 2004).

The RNAhybrid algorithm by Rehmsmeier et al. (2004) computes minimum-free energy hybridization sites for miRNAs, while forcing perfect complementarity in nucleotides

Reprint requests to: Pål Sætrom, Interagon AS, Medisinsk teknisk senter, NO-7489 Trondheim, Norway; e-mail: paal.saetrom@interagon.com.

Article published online ahead of print. Article and publication date are at <http://www.rnajournal.org/cgi/doi/10.1261/rna.7290705>.

(nt) 2–7. Potential sites are normalized by the product of a miRNA and its potential target to avoid high-scoring, but unlikely hybridizations to long target sequences. Extreme value statistics similar to that used in sequence-similarity searching is used to determine the likelihood of a candidate site being due to random hits in a large database. The DIANA-microT algorithm also minimizes the duplex-binding energy in its initial step (Kiriakidou et al. 2004). Most of the existing miRNA target prediction algorithms use similar thermodynamic calculations in post-processing steps following a requirement of near-perfect complementarity with the targets in the miRNAs' 5' ends.

Rajewsky and Socci (2004) define a binding nucleus of consecutive base pairs, and calculate a weighted sum typically consisting of six to eight addends favoring more hydrogen bonds. The algorithm is referred to as Nucleus throughout this article, and its position-specific weights differ only slightly from the weights of a similar algorithm called miRanda (Enright et al. 2003). In a subsequent post-processing step, Nucleus uses folding-free energy as determined by mfold (Zuker 2003) to make the final predictions. Simpler algorithms that use a seed of perfect complementarity in the miRNA's 5' region include TargetScan (Lewis et al. 2003) and an algorithm from EMBL (Stark et al. 2003), but these run the risk of losing targets that do not exactly meet their seed criteria.

We have developed a machine-learning algorithm called TargetBoost that creates classifiers for predicting miRNA target sites, and this is a novel approach to miRNA target site prediction. The algorithm, which is an adaptation of the boosted genetic programming algorithm of Sætrom (2004), creates weighted sequence motifs that characterize the probable binding characteristics between miRNAs and target sites. That is, given a miRNA and a potential target site, this classifier returns a score that represents the likelihood of the site being targeted by the miRNA. We used our classifiers to predict target sites in a set of genes important for fly body patterning in *Drosophila melanogaster*.

TargetBoost compares favorably to the algorithms of Rajewsky and Socci (2004) and Rehmsmeier et al. (2004) that were described previously. First, it rediscovers that miRNAs' 5' ends bind well to targets. Second, it proves to be a classifier with a high and stable performance across several targets. Third, and most importantly, it discovers more true targets than the aforementioned algorithms. As other known algorithms use variants of the Nucleus and RNAhybrid approaches, the performance of these two algorithms should be representative of the other algorithms' performance as well.

We have not included additional filters, such as requiring conservation of the target sites or the presence of multiple target sites in the 3' UTRs, in our algorithm comparisons. The reason is that these filters can be used independently of the initial method used to predict the target sites. Thus, improving the quality of the initial candidates will also improve the final predictions.

In summary, our main contributions are a new algorithm for predicting miRNA target sites, and an objective comparison of its performance to that of existing algorithms.

RESULTS

A machine learning algorithm that predicts miRNA target sites

GPboost is a machine-learning algorithm that, from a training set of positive and negative sequences, creates a sequence-based classifier that recognizes the positive sequences (Sætrom 2004). The classifier is the sum of several differentially weighted sequence patterns, where each pattern answers either yes (1) or no (−1) as to whether the pattern matches a given sequence or not. We have previously used variants of GPboost to predict the efficacy of short interfering RNAs (Sætrom 2004; Sætrom and Snøve Jr. 2004) and noncoding RNA genes in *Escherichia coli* (P. Sætrom, R. Sneve, K.I. Kristiansen, O. Snøve Jr., T. Grünfeld, T. Rognes, and E. Seeberg, in prep.).

To create the classifier, GPboost combines genetic programming (GP) (Koza 1992) and boosting (Meir and Rätsch 2003). More specifically, GP evolves the individual sequence patterns from a population of candidate patterns, and the boosting algorithm guides GP's search by adjusting the importance of each sequence in the training set. Then, the boosting algorithm assigns weights to the sequence patterns based on the patterns' performance in the corresponding training set. The final classifier is the average of several such boosted GP classifiers. Sætrom (2004) gives a more thorough description of the algorithm.

To train the miRNA target site predictors, we use a variant of the GPboost program, called TargetBoost, with two main differences. First, in Sætrom (2004) the patterns were simple queries, but the patterns we use here are template queries. That is, the sequence patterns are general expressions that describe the common properties of miRNA target sites. When using the patterns to search for target sites, we translate the general expressions into queries that are specific for each miRNA. Second, we use a different language to define what patterns are legal solutions. In the Materials and Methods, we give a formal definition of this pattern language along with additional details on how TargetBoost translates the patterns into miRNA-specific queries.

TargetBoost finds a good, stable miRNA target site predictor

To train and test the TargetBoost classifiers, we used a set of 36 experimentally verified target sites as positive data and a larger set of random sequences as negative data (see Materials and Methods for details). We compared TargetBoost's performance with the performance of Nucleus (Rajewsky and Socci 2004) and RNAhybrid (Rehmsmeier et al. 2004)—two recently published methods for identifying miRNA targets.

To test the algorithms, we used 10-fold and leave-one-miRNA-out cross-validation, and used receiver operating characteristics (ROC) analysis to compare the algorithms' performance; see Materials and Methods for further descriptions.

Figure 1 shows the 10-fold cross-validation ROC-curves for TargetBoost, RNAhybrid, and Nucleus. When comparing the curves for the different algorithms, we see that TargetBoost and RNAhybrid are better than Nucleus on high-specificity levels, with TargetBoost slightly better than RNAhybrid on specificity levels above 0.9.

Figure 2 shows the leave-one-miRNA-out cross-validation results as it displays the ROC-curves for TargetBoost, RNAhybrid, and Nucleus for each miRNA in the training set individually. We see that RNAhybrid and TargetBoost have approximately the same ROC-curves for every miRNA, with TargetBoost being slightly better for every miRNA except *miR-13a* and the high-specificity regions of *lin-4*. Nucleus has the highest performance for *lin-4*.

To compare the overall performance of the three algorithms, we computed the ROC-score for each algorithm on each miRNA. Then, on each individual miRNA, we tested whether the best algorithm was significantly better than the other algorithms. As Table 1 shows, TargetBoost not only had the best overall ROC-score, it was also the most stable of the three target site predictors, as for each individual miRNA, TargetBoost was either the best algorithm (*let-7* and *bantam*) or as good as the best algorithm (Nucleus for *lin-4* and RNAhybrid for *miR-13a*). Both RNAhybrid and Nucleus, however, were significantly worse than the best algorithm on at least one miRNA.

Although the overall performance of the classifiers is important, when using a classifier to predict miRNA target sites in genes, the most important characteristics of the classifier is that the top predictions made by the classifier

have a high probability of being true target sites. That is, the best classifier has higher sensitivity than the other classifiers when approaching maximal specificity.

The true-positive frequency (TPF) test determines whether there is a significant difference in the sensitivity of two classifiers at a given significance level (see Materials and Methods). For each miRNA, we tested whether the best classifier was significantly more sensitive than the other classifiers (99% confidence level) on specificities 0.995, 0.99, 0.98, 0.97, 0.96, and 0.95. On all specificities, TargetBoost was either the best or as good as the highest-scoring algorithm on all genes. RNAhybrid performed well on all specificities for all miRNAs except *lin-4*, where the algorithm was significantly less sensitive than Nucleus on all specificities. Nucleus, however, suffered from lower sensitivity in the high-specificity area; TargetBoost was significantly more sensitive than Nucleus for *let-7* (specificity 0.995—*P*-value 0.006) and *miR-13a* (specificities 0.995 and 0.99—*P*-values 0.005 and 0.006). Thus, as for the overall ROC-score, TargetBoost was the most stable of the three algorithms.

A possible explanation for RNAhybrid and Nucleus being less stable than TargetBoost is that the different miRNAs have slightly different binding characteristics. For example, *lin-4* and its target sites have a lower binding energy compared with the three other miRNAs, but may have other characteristics that the sequence-based methods, TargetBoost and Nucleus, have used to identify the target sites. This can explain RNAhybrid's poorer performance on this miRNA. The motif-based classifiers of TargetBoost, however, seem to be robust and capture both the thermodynamic and sequence characteristics of the miRNA target sites in our database.

TargetBoost finds more true target sites than do RNAhybrid and Nucleus

When we search for target sites, there will be far more negative than positive target sites. We are therefore interested in a classifier that finds as many positive target sites as possible, before the number of negative target sites in the result set becomes too large. The ROC₅₀-score, which is the area under the ROC curve until 50 false positives are found, reflects this interest, as the score takes into account that a user is seldom concerned with true positives that occur after the first page (about 50) of false positives (Gribnikov and Robinson 1996). We ran a ROC₅₀ test on the different algorithms to compare their performance on low frequencies of false positives; Table 2 lists the scores.

We found that TargetBoost performs better than RNAhybrid and Nucleus. Both RNAhybrid and Nucleus can be given extra constraints, such as forcing miRNA 5' helices in RNAhybrid and increasing the free-energy cutoff in Nucleus, to improve their predictive power. Both perform much better when they are given extra constraints, and

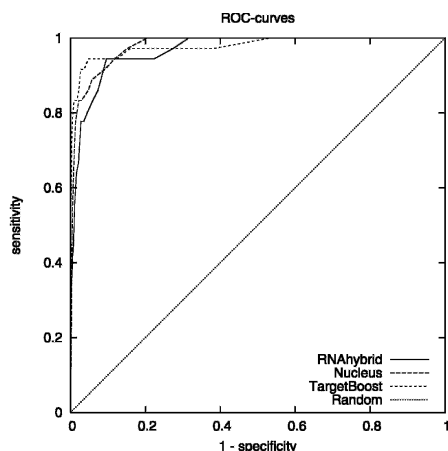


FIGURE 1. Overall ROC-curves for each algorithm.

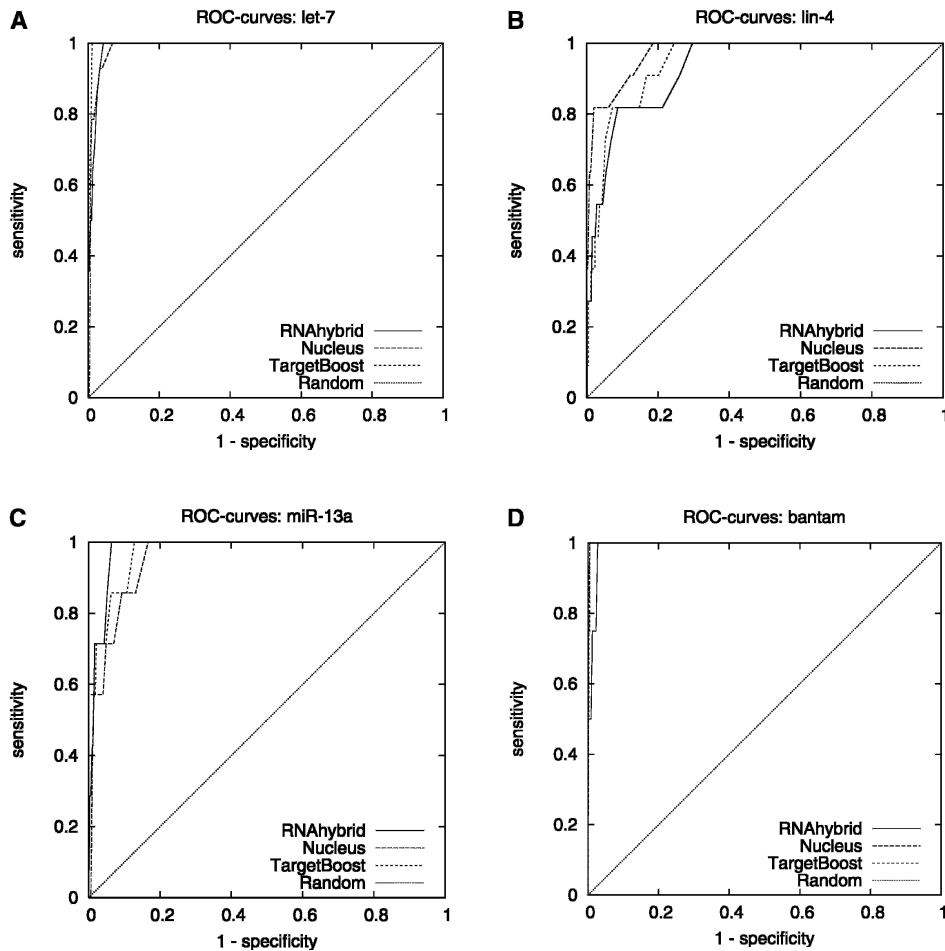


FIGURE 2. ROC-curves. A–D compare the performance of RNAhybrid, Nucleus, and TargetBoost at predicting the true target sites of *let-7*, *lin-4*, *miR-13a*, and *bantam*.

especially RNAhybrid get a much higher sensitivity for high levels of specificity (see Fig. 3A). The drawback is that the algorithms will miss several miRNA target sites when they are using these constraints. Figure 3 gives the complete ROC-curves for the different versions of RNAhybrid and Nucleus. TargetBoost does not have this problem, as each target site will get a score by TargetBoost and no target site will automatically be discarded. What is more, as Table 2

shows, TargetBoost finds more true target sites, even when the constraints are introduced in RNAhybrid and Nucleus.

TargetBoost rediscovers that 5' ends bind with near-perfect complementarity

Earlier methods that identify miRNA target sites have used the property that the miRNA tends to bind perfectly to the target site on the 5' end of the miRNA. Enright et al. (2003), Kiriakidou et al. (2004), Lewis et al. (2003), and Stark et al. (2003) use this property directly by demanding perfect binding at the 5' end as a seed. Nucleus (Rajewsky and Socci 2004) uses the property indirectly by demanding a long GC-rich sequence of matches. This sequence will most often appear at the 5' end of the miRNA. RNAhybrid (Rehmsmeier et al. 2004) can also incorporate this property by demanding that parts of the miRNA have to form a perfect helix. By demanding a perfect helix on nt 2–7 on

TABLE 1. TargetBoost is the most stable algorithm

Algorithm	<i>let-7</i>	<i>lin-4</i>	<i>miR-13a</i>	<i>bantam</i>	All
TargetBoost	0.997	0.944	0.972	0.998	0.979
RNAhybrid	0.989	0.931	0.979	0.991	0.967
Nucleus	0.988	0.962	0.928	0.998	0.973

ROC-scores that are not significantly different from the highest score on a particular miRNA are in boldface (90% confidence level; see Materials and Methods for details on each algorithm).

TABLE 2. ROC₅₀ scores for the algorithms on the complete data set

Algorithm	ROC ₅₀ -score
TargetBoost	0.0025
RNAhybrid ₁	0.0012
RNAhybrid ₂	0.0017
Nucleus ₁	0.0006
Nucleus ₂	0.0011
Nucleus ₃	0.0014

See Materials and Methods for descriptions of the different algorithms.

the 5' end of the miRNA, better results were observed (see Rehmsmeier et al. 2004; Fig. 3A; Table 2).

TargetBoost confirmed the tendency of perfect matching in the 5' end. The production rules used to create the classifiers demand a segment of near-perfect pairing between miRNA and target site, but the position and length of this pairing is not encoded in the production. This is decided entirely by the training process. Almost every individual trained at the first boosting iteration resembled the expression in Figure 4. That is, in most expressions, the consecutive sequence part of the expressions (the rightmost {...} subexpression in Fig. 4) used positions 17–24 in the miRNA counted from the 3' end. These positions correspond to the first eight bases on the 5' end. As explained in the Materials and Methods, the $P \geq 6$ means that six of the eight bases have to match at the 5' core, and this indicates that almost every target site in the training set demands a near-perfect match in the 5' end of the miRNA. This corresponds to experimental evidence in the literature (Doench and Sharp 2004; Kiriakidou et al. 2004).

Target candidates in *Drosophila melanogaster*

We searched a set of genes important for fly body patterning in *D. melanogaster* for candidate target sites. This set is the same as was used in Rajewsky and Socci (2004) and Rehmsmeier et al. (2004). In the search, we used a set of 78 *D. melanogaster* miRNAs downloaded from the miRNA Registry version 5.0 (Griffiths-Jones 2004). We compared the target sites found in our search with the target sites predicted by Rehmsmeier et al. (2004) and Rajewsky and Socci (2004).

Figure 5 displays target sites predicted by either TargetBoost, RNAhybrid, or both. When comparing our results to the top five hits predicted by Rehmsmeier et al. (2004), we found that TargetBoost did not predict the potential *miR-92a* site in *tailless* and the potential *miR-210* site in *hairy* reported by RNAhybrid. This is because of the number of G:U wobbles in the target sites reported by RNAhybrid; for example, the *miR-92a* target in *tailless* has three G:U wobbles, two of them residing in the 5' core (see Fig. 5). The *miR-210* site in *hairy* has five G:U wobbles, with three wobbles in the first eight bases of the 5' core. As TargetBoost treats G:U wobbles as normal mismatches, we would not find potential target sites with a high number of G:U wobbles; especially if the sites resided in the 5' core. This may, however, be a strength of our method, as recent experimental results suggest that G:U wobbles may be detrimental to translational repression (Doench and Sharp 2004).

Although we did not find the same *miR-210* site in *hairy* as did RNAhybrid, TargetBoost did predict that *miR-7* has a potential target site in *hairy*. The target site is the same as the ones predicted by RNAhybrid and Nucleus, and it has only one G:U wobble. Stark et al. (2003) has shown that *hairy* is a target for *miR-7*.

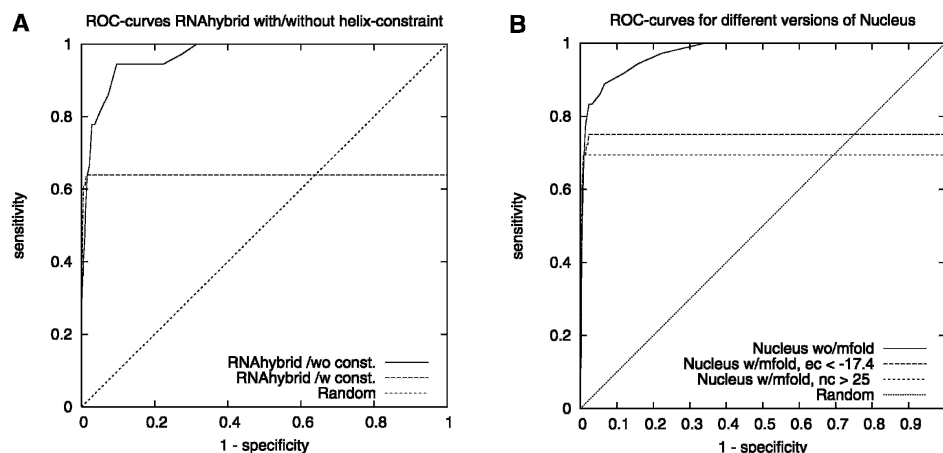


FIGURE 3. ROC-curves comparing different parameter settings on RNAhybrid (A) and Nucleus (B). We can see increased sensitivity for high-specificity values for RNAhybrid in A.

D. melanogaster and *Caenorhabditis elegans*. Specifically, the stretches of perfect complementarity may be longer, targets in the protein-coding region may be present, and the bias toward perfect complementarity in the miRNA's 5' region may be weaker. If this is true, current miRNA target prediction algorithms may have limited value when used to predict targets in mammals.

In summary, we have presented a new algorithm for predicting miRNA target sites. The algorithm uses machine learning to train a sequence-based target site predictor, and this is a novel approach to miRNA target site prediction. Our algorithm compares favorably to other algorithms, both in terms of overall performance and when making highly specific predictions. We believe that our algorithm will be an important tool, not only for finding the target sites of known miRNAs, but also for predicting potential miRNA off-target effects in RNAi experiments (Saxena et al. 2003; Scacheri et al. 2004).

MATERIALS AND METHODS

Algorithm and implementation

TargetBoost ensures that all patterns evolved in the genetic programming process are valid expressions in a pattern language (Sætrum 2004). Figure 6 shows the grammar and semantics of the pattern language used to create the miRNA target predictors. The grammar is in Backus-Naur form (Knuth 1964) and shows the legal production rules in the language, with nonterminals represented by uppercase letters and terminals represented by boldface letters. Syntactical elements in the language, such as parentheses and operators, are in normal typeface, alternatives are represented as separate productions, adjacent symbols are concatenated, and P_i represents position i in the miRNA-sequence, counted from the 3' end.

Production	Semantic Rule
(1) $S \rightarrow (D)(O)$	$S.hit := D.hit \text{ AND } O.hit$
(2) $D \rightarrow \{F : d = N\}$	$D.hit := \text{linger}(F.hit, N)$
(3) $F \rightarrow (R)(W)$	$F.hit := R.hit \text{ AND } W.hit$
(4) $R \rightarrow \{C : p \geq N\}$	$R.hit := C.count \geq N.cutoff$
(5) $R \rightarrow C$	$R.hit := C.hit$
(6) $C \rightarrow (C_1)(C_2)$	$C.count := C_1.count + C_2.count$ $C.hit := C_1.hit \text{ AND } C_2.hit$
(7) $C \rightarrow A$	$C.count := A.count$ $C.hit := A.hit$
(8) $A \rightarrow A_1 A_2$	$A.count := A_1.count + A_2.count$ $A.hit := A_1.hit \text{ OR } A_2.hit$
(9) $A \rightarrow L$	$A.count := L.count$ $A.hit := L.hit$
(10) $L \rightarrow \mathbf{a}$, for some $\mathbf{a} \in \{P_1, \dots, P_{24}\}$	$L.count := \text{match}(\mathbf{a})$ $L.hit := \text{match}(\mathbf{a})$
(11) $N \rightarrow \mathbf{n}$, for some $\mathbf{n} \in \{1, 2, \dots\}$	$N.cutoff := \mathbf{n}$
(12) $W \rightarrow \{. : r = N\}$	$W.hit := 1$
(13) $O \rightarrow \{LC : p \geq N\}$	$O.hit := LC.count \geq N.cutoff$
(14) $LC \rightarrow (LC_1)(LC_2)$	$LC.count := LC_1.count + LC_2.count$
(15) $LC \rightarrow L$	$LC.count := L.count$

FIGURE 6. The grammar (A) and semantics (B) of the pattern language used by TargetBoost. The grammar and semantics are explained in the main text.

Unknown pattern	Variable distance	Consecutive pattern
$Q_1: \{ \{P_1P_2(P_3 P_3)P_4 : p \geq 3\}$	$\{. : r = 8\} : d = 7$	$\{P_6P_7P_8P_9P_{10}P_{11} : p \geq 2\}$
$Q_2: \{ P_5P_{20}((P_3 P_{24}) P_{16})$	$\{. : r = 4\} : d = 10$	$\{P_{13}P_{14}P_{15}P_{16}P_{17} : p \geq 5\}$

FIGURE 7. Two example patterns from our pattern language. P_i denotes nucleotide i in the miRNA counted from the 3' end.

Figure 6B shows the language's semantics. A pattern matches a sequence if $S.hit$ is true. $match(\mathbf{a})$ returns 1 if the character in the position indicated by \mathbf{a} is identical to the character it is compared with. $linger(F.hit, N)$ is a function that if $F.hit$ is true, $F.hit$ will be returned for N clock-ticks (see Halaas et al. (2004) for details on the $linger$ -function). The production for W creates a sequence of N wild cards. This production will return a hit for any sequence of N characters it is compared with.

Each individual generated by these production rules consists of two parts as follows: an unknown pattern R , and a consecutive sequence O of near perfect matches. The two parts are separated by a variable amount of nucleotides, decided by the displacement D . The number of wild cards in the W -production gives the lower bound of the number of nucleotides, and the number of wild cards, plus the displacement d in the D -production gives the upper bound of the number of nucleotides.

Figure 7 shows two example patterns from our pattern language. In the first query, the unknown pattern and the consecutive sequence are separated by 8–15 nt, and in the second query, by 4–14 nt. As in Sætrum (2004), we use the pattern n -of- m operator ($P \geq N$ in productions 4 and 13 in Fig. 6) to introduce fuzzy matching. That is, the numeral N in productions 4 and 13 indicates the minimum number of terminals in the C and LC productions that must match. For example, in Q_1 , only two of six nucleotides must match, but in Q_2 , all five nucleotides must match. This is also the case for the unknown pattern; the complete expression must match in Q_2 , as it does not use the pattern n -of- m operator, but only three of four nucleotides must match in Q_1 .

The terminals in the expressions represent positions in the miRNA-sequence; the expressions are therefore translated before searching. During translation, the terminals that represent positions are replaced with the corresponding complemented nucleotide in the miRNA sequence. The positions in the miRNA are numbered from P_1 to P_{24} , with P_{24} corresponding to the 5' end of the miRNA. Our current implementation translates the miRNAs from 5' to 3', but only uses the 21 first nucleotides— P_1 to P_3 defaults to wild cards that match any nucleotide. TargetBoost evaluates a candidate pattern by using the translated queries to search the training set of positive and negative sequences. It then scores the pattern based on the number of true and false positive/negative hits and the relative weights the boosting algorithm has assigned to the sequences.

Reference algorithms for comparison

We compared the performance of TargetBoost with the performance of Nucleus (Rajewsky and Socci 2004) and RNAhybrid (Rehmsmeier et al. 2004) (these algorithms are described in the Introduction). Nucleus has two cut-off parameters that can be tuned—the weighted sum cut-off and the free energy cut-off—and when comparing the performance of this algorithm with the performance of our algorithm, we made certain modifications.

Nucleus₁ does not use mfold, and therefore, has only one cut-off parameter to tune. Nucleus₂ has a free-energy cut-off of -17.4 , while the weighted sum cut-off is tunable. This was the cut-off recommended in Rajewsky and Socci (2004). Nucleus₃ has a weighted sum cut-off of 25 , while the free-energy cut-off is tunable. Again, this cut-off was recommended in Rajewsky and Socci (2004).

We ran RNAhybrid in two modes; RNAhybrid₁ ran without forcing miRNA 5' helices, and RNAhybrid₂ forced miRNA 5' helices from position two to seven, as suggested by Rehmsmeier et al. (2004). Throughout this work, RNAhybrid and Nucleus are short for Nucleus₁ and RNAhybrid₁.

Positive data set

The positive data set consisted of 36 experimentally confirmed target sites for the miRNAs *let-7*, *lin-4*, *miR-13a*, and *bantam* in *C. elegans* and *D. melanogaster* (Boutla et al. 2003; Brennecke et al. 2003; Rajewsky and Socci 2004). Each target site was padded with their respective sequences, such that the length of the sequences was 30 nt. Target sites longer than 30 nt were discarded from the data set.

Negative data set

The negative data set consisted of 3000 random strings, all 30 nt long. The frequencies used in the generation of the random strings were the same as the frequencies used in Rajewsky and Socci (2004), ($P_A=0.34$, $P_C=0.19$, $P_G=0.18$, $P_U=0.29$), and correspond to the nucleotide composition of *D. melanogaster* 3' UTRs.

Cross-validation

Cross-validation is a common method to evaluate the performance of a classifier on data not used to train the classifier. Here, we used 10-fold cross-validation (Breiman et al. 1984) and an approach we call "leave-one-miRNA-out" cross-validation. A 10-fold cross-validation usually gives a good estimate of a classifier's predictive accuracy (Kohavi 1995). In this case, however, the number of verified target sites for each miRNA varied greatly, so that the miRNA having the most target sites (*let-7*) had a high chance of being present in both the training and test sets in many of the 10-folds. As this may cause a bias in the classifier performance estimated by the 10-fold cross-validation method, we tried a second cross-validation approach that did not have this bias. In the "leave-one-miRNA-out" cross-validation approach, we used all of the target sites from all of the miRNAs, but one, as training set; we then used the remaining miRNA's target sites as test set. This gave four training and test sets.

Comparing algorithms

We compared the algorithms by analyzing their receiver operating characteristics (ROC) curves. A ROC-curve describes the relationship between the specificity $Sp = TN/(FP + TN)$ and the sensitivity $Se = TP/(TP + FN)$ of a classifier. Here, TP , FP , TN , and FN are the number of true positives, false positives, true negatives, and false negatives.

We did three analyses on the ROC-curves, i.e., area tests, TPF tests, and ROC₅₀ tests. In the area tests, we calculate the area under the ROC-curve—the ROC-score. An area of 1 indicates a perfect classification, and an area of 0.5 indicates a random classification. In the TPF tests, we calculate the true-positive frequency (TPF = S_p) for a classifier for a given false-positive frequency (FPF = $1 - S_p$), or the amount of correctly classified positive samples given a specified amount of false-positive samples. In the ROC₅₀ tests, we calculate the ROC₅₀ score, which is the area under the ROC-curve plotted until 50 true negative samples are found (Gribskov and Robinson 1996).

We used ROCKIT (Metz et al. 1998) for statistical comparisons of ROC area and TPF values.

Availability

TargetBoost is available as a Web tool from <http://www.interagon.com/demo/>. Currently, the Web tool searches the 3' UTRs of *C. elegans*; other data sets are available for both commercial and strategic academic collaborations.

ACKNOWLEDGMENTS

We thank O.R. Birkeland for valuable comments on the manuscript and N. Rajewsky for sharing his data set of miRNA target sites. The work was supported by the Norwegian Research Council, grant 151899/150, and the bioinformatics platform at the Norwegian University of Science and Technology, Trondheim, Norway.

Received December 29, 2004; accepted April 7, 2005.

REFERENCES

- Bartel, D.P. 2004. MicroRNAs: Genomics, biogenesis, mechanism, and function. *Cell* **116**: 281–297.
- Boutla, A., Delidakis, C., and Tabler, M. 2003. Developmental defects by antisense-mediated inactivation of micro-RNAs 2 and 13 in *Drosophila* and the identification of putative target genes. *Nucleic Acids Res.* **31**: 4973–4980.
- Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. 1984. *Classification and regression trees*. Wadsworth, Belmont, CA.
- Brennecke, J., Hipfner, D.R., Stark, A., Russell, R.B., and Cohen, S.M. 2003. *bantam* Encodes a developmentally regulated miRNA that controls cell proliferation and regulates the proapoptotic gene *hid* in *Drosophila*. *Cell* **113**: 25–36.
- Doench, J.G. and Sharp, P.A. 2004. Specificity of microRNA target selection in translational repression. *Genes & Dev.* **18**: 504–511.
- Doench, J., Petersen, C., and Sharp, P. 2003. siRNAs can function as miRNAs. *Genes & Dev.* **17**: 438–442.
- Enright, A.J., John, B., Gaul, U., Tuschl, T., Sander, C., and Marks, D.S. 2003. MicroRNA targets in *Drosophila*. *Genome Biol.* **5**: R1.
- Gribskov, M. and Robinson, N.L. 1996. The use of receiver operator characteristic (ROC) analysis to evaluate sequence matching. *Comput. Chem.* **20**: 25–34.
- Griffiths-Jones, S. 2004. The microRNA registry. *Nucleic Acids Res.* **32**: D109–D111.
- Halaas, A., Svingen, B., Nedland, M., Sætrom, P., Snøve Jr., O., and Birkeland, O.R. 2004. A recursive MISD architecture for pattern matching. *IEEE Trans. on VLSI Syst.* **12**: 727–734.
- John, B., Enright, A.J., Aravin, A., Tuschl, T., Sander, C., and Marks, D.S. 2004. Human microRNA targets. *PLoS Biol.* **2**: e363.

- Kiriakidou, M., Nelson, P.T., Kouranov, A., Fitziev, P., Bouyioukos, C., Mourelatos, Z., and Hatzigeorgiou, A. 2004. A combined computational-experimental approach predicts human microRNA targets. *Genes & Dev.* **18**: 1165–1178.
- Kloosterman, W.P., Wienholds, E., Ketting, R.F., and Plasterk, R.H. 2004. Substrate requirements for *let-7* function in the developing zebrafish embryo. *Nucleic Acids Res.* **32**: 6284–6291.
- Knuth, D.E. 1964. Backus normal form vs. Backus Naur form. *Commun. ACM* **7**: 735–736.
- Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1137–1143. Morgan Kaufmann Publishers, Montreal Canada.
- Koza, J.R. 1992. *Genetic programming: On the programming of computers by natural selection*. MIT Press, Cambridge, MA.
- Lagos-Quintana, M., Rauhut, R., Lendeckel, W., and Tuschl, T. 2001. Identification of novel genes coding for small expressed RNAs. *Science* **294**: 853–858.
- Lai, E.C. 2004. Predicting and validating microRNA targets. *Genome Biol.* **5**: 115.
- Lai, E.C., Tomancak, P., Williams, R.W., and Rubin, G.M. 2003. Computational identification of *Drosophila* microRNA genes. *Genome Biol.* **4**: R42.
- Lau, N.C., Lim, L.P., Weinstein, E.G., and Bartel, D.P. 2001. An abundant class of tiny RNAs with probable regulatory roles in *Caenorhabditis elegans*. *Science* **294**: 858–862.
- Lee, R.C. and Ambros, V. 2001. An extensive class of small RNAs in *Caenorhabditis elegans*. *Science* **294**: 862–864.
- Lee, R.C., Feinbaum, R., and Ambros, V. 1993. The *C. elegans* heterochronic gene *lin-4* encodes small RNAs with antisense complementarity to *lin-14*. *Cell* **75**: 843–854.
- Lewis, B.P., Hung Shih, I., Jones-Rhoades, M.W., Bartel, D.P., and Burge, C.B. 2003. Prediction of mammalian microRNA targets. *Cell* **115**: 787–798.
- Lim, L.P., Glasner, M.E., Yekta, S., Burge, C.B., and Bartel, D.P. 2003a. Vertebrate microRNA genes. *Science* **299**: 1540.
- Lim, L.P., Lau, N.C., Weinstein, E.G., Abdelhakim, A., Yekta, S., Rhoades, M.W., Burge, C.B., and Bartel, D.P. 2003b. The microRNAs of *Caenorhabditis elegans*. *Genes & Dev.* **17**: 991–1008.
- Meir, R. and Rätsch, G. 2003. An introduction to boosting and leveraging. In *Advanced lectures on machine learning* (eds. S. Mendelson and A. Smola), Vol. 2600, pp. 118–183. Springer-Verlag, GmbH.
- Metz, C.E., Herman, B.A., and Roe, C.A. 1998. Statistical comparison of two ROC-curve estimates obtained from partially-paired datasets. *Med. Decis. Making* **18**: 110–121.
- Moss, E.G., Lee, R.C., and Ambros, V. 1997. The cold shock domain protein LIN-28 controls developmental timing in *C. elegans* and is regulated by the *lin-4* RNA. *Cell* **88**: 637–646.
- Olsen, P.H. and Ambros, V. 1999. The *lin-4* regulatory RNA controls developmental timing in *Caenorhabditis elegans* by blocking LIN-14 protein synthesis after the initiation of translation. *Dev. Biol.* **216**: 671–680.
- Pasquinelli, A.E., Reinhart, B.J., Slack, F., Martindale, M.Q., Kuroda, M.I., Maller, B., Hayward, D.C., Ball, E.W., Degnan, B., Müller, P., et al. 2000. Conservation of the sequence and temporal expression of *let-7* heterochronic regulatory RNA. *Nature* **408**: 86–89.
- Rajewsky, N. and Socci, N.D. 2004. Computational identification of microRNA targets. *Dev. Biol.* **267**: 529–535.
- Rehmsmeier, M., Steffen, P., Höchsmann, M., and Giegerich, R. 2004. Fast and effective prediction of microRNA/target duplexes. *RNA* **10**: 1507–1517.
- Reinhart, B., Slack, F., Basson, M., Pasquinelli, A., Bettinger, J., Rougvie, A., Horvitz, H., and Ruvkun, G. 2000. The 21-nucleotide *let-7* RNA regulates developmental timing in *Caenorhabditis elegans*. *Nature* **403**: 901–906.
- Sætrom, P. 2004. Predicting the efficacy of short oligonucleotides in antisense and RNAi experiments with boosted genetic programming. *Bioinformatics* **20**: 3055–3063.
- Sætrom, P. and Snøve Jr., O. 2004. A comparison of siRNA efficacy predictors. *Biochem. Biophys. Res. Commun.* **321**: 247–253.
- Saxena, S., Jonsson, Z., and Dutta, A. 2003. Implications for off-target activity of small inhibitory RNA in mammalian cells. *J. Biol. Chem.* **278**: 44312–44319.
- Scacheri, P.C., Rozenblatt-Rosen, O., Caplen, N.J., Wolfsberg, T.G., Umayam, L., Lee, J.C., Hughes, C.M., Selvi Shanmugam, K., Bhattacharjee, A., Meyerson, M., et al. 2004. Short interfering RNAs can induce unexpected and divergent changes in the levels of untargeted proteins in mammalian cells. *Proc. Natl. Acad. Sci.* **101**: 1892–1897.
- Smalheiser, N.R. and Torvik, V.I. 2004. A population-based statistical approach identifies parameters characteristic of human microRNA-mRNA interactions. *BMC Bioinformatics* **5**: 139.
- Stark, A., Brennecke, J., Russell, R.B., and Cohen, S.M. 2003. Identification of *Drosophila* microRNA targets. *PLoS Biol.* **1**: E60.
- Wightman, B., Ha, I., and Ruvkun, G. 1993. Posttranscriptional regulation of the heterochronic gene *lin-14* by *lin-4* mediates temporal pattern formation in *C. elegans*. *Cell* **75**: 855–862.
- Yekta, S., Shih, I., and Bartel, D.P. 2004. MicroRNA-directed cleavage of *HOXB8* mRNA. *Science* **304**: 594–596.
- Zeng, Y., Wagner, E., and Cullen, B. 2002. Both natural and designed micro RNAs can inhibit the expression of cognate mRNA when expressed in human cells. *Mol. Cell.* **9**: 1327–1333.
- Zuker, M. 2003. Mfold Web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.* **31**: 3406–3415.

Bibliography

- [1] National Human Genome Research Institute. Talking glossary of genetic terms: Illustration, 2005. http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration/base_pair.shtml.
- [2] Michael S. Waterman. *Introduction to computational biology*. Chapman & Hall/CRC, 1995.
- [3] Rosalind C. Lee, R.L. Feinbaum, and Victor Ambros. The *C. elegans* heterochronic gene *lin-4* encodes small RNAs with antisense complementarity to *lin-14*. *Cell*, 75(5):843–854, 1993.
- [4] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, volume 2600, pages 118–183. Springer-Verlag, 2003.
- [5] David W. Mount. *Bioinformatics: Sequence and genome analysis*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 2001.
- [6] Benjamin Lewin. *Genes VII*. Oxford University Press, Oxford, UK, 2000.
- [7] David P. Bartel. MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell*, 116(2):281–297, 2004.
- [8] Sam Griffiths-Jones. The microRNA registry. *Nucleic Acids Res.*, 32(90001):D109–111, 2004.
- [9] Julius Brennecke, Alexander Stark, Robert B. Russell, and Stephen M. Cohen. Principles of microRNA-target recognition. *PLoS Biology*, 3(3):e85, 2005.
- [10] Nikolaus Rajewsky and Nicholas D. Socci. Computational identification of microRNA targets. *Dev. Biol.*, 267(2):529–535, 2004.
- [11] Marc Rehmsmeier, Peter Steffen, Matthias Höchsmann, and Robert Giegerich. Fast and effective prediction of microRNA/target duplexes. *RNA*, 10(10):1507–1517, 2004.

- [12] John G. Doench and Phillip A. Sharp. Specificity of microRNA target selection in translational repression. *Genes Dev.*, 18(5):504–511, 2004.
- [13] David P. Bartel and Chang-Zheng Chen. Micromanagers of gene expression: the potentially widespread influence of metazoan microRNAs. *Nat. Rev. Genet.*, 5(5):396–400, 2004.
- [14] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers, San Francisco, CA, 1997.
- [15] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [16] Pierre Baldi and Søren Brunak. *Bioinformatics: the machine learning approach*. The MIT Press, Cambridge, Massachusetts, 2nd edition, 2001.
- [17] Michael Gribskov and Nina L. Robinson. The use of receiver operator characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–34, 1996.
- [18] John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, Cambridge Massachusetts, Dec 1992.
- [19] Arne Halaas, Børge Svingen, Magnar Nedland, Pål Sætrum, Ola Snøve Jr., and Olaf Renè Birkeland. A recursive MISD architecture for pattern matching. *IEEE Trans. on VLSI Syst.*, 12(7):727–734, 2004.
- [20] Interagon AS. The Interagon query language : a reference guide. <http://www.interagon.com/pub/whitepapers/IQL.reference-latest.pdf>, sep 2002.
- [21] Eric C Lai. Predicting and validating microRNA targets. *Genome Biol.*, 5(9):115, 2004.
- [22] Benjamin P. Lewis, I hung Shih, Matthew W. Jones-Rhoades, David P. Bartel, and Christopher B. Burge. Prediction of mammalian microRNA targets. *Cell*, 115(7):787–798, 2003.
- [23] Azra Krek, Dominic Grün, Matthew N. Poy, Rachel Wolf, Lauren Rosenberg, Eric J. Epstein, Philip MacMenamin, Isabella da Piedade, Kristin C. Gunsalus, Markus Stoffel, and Nikolaus Rajewsky. Combinatorial microRNA target predictions. *Nat. Genet.*, 37(5):495–500, 2005.
- [24] Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, 31(13):3406–3415, 2003.
- [25] Ivo L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Res.*, 31(13):3429–3431, 2003.

- [26] Harlan Robins, Ying Li, and Richard W. Padgett. Incorporating structure to predict microRNA targets. *Proc. Natl. Acad. Sci. U.S.A.*, 102(11):4006–4009, 2005.
- [27] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, New York, NY, USA, 2nd edition, 2001.
- [28] Nikolaus Rajewsky, Massimo Vergassola, Ulrike Gaul, and Eric D. Siggia. Computational detection of genomic *cis*-regulatory modules applied to body patterning in the early *Drosophila* embryo. *BMC Bioinformatics*, 3(30), 2002.
- [29] Gregory Paris, Denis Robilliard, and Cyril Fonlupt. Applying boosting techniques to genetic programming. In P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, editors, *Proc. of Artificial Evolution: 5th Int. Conf.*, pages 267–280. Springer-Verlag, Oct 2001.
- [30] Pål Sætrom. Predicting the efficacy of short oligonucleotides in antisense and RNAi experiments with boosted genetic programming. *Bioinformatics*, 20(17):3055–3063, 2004.
- [31] David J. Montana. Strongly typed genetic programming. *Evol. Comput.*, 3(2):199–230, 1995.
- [32] Julius Brennecke, David R. Hipfner, Alexander Stark, Robert B. Russell, and Stephen M. Cohen. *bantam* encodes a developmentally regulated miRNA that controls cell proliferation and regulates the proapoptotic gene *hid* in *Drosophila*. *Cell*, 113(1):25–36, 2003.
- [33] Alexandra Boutla, Christos Delidakis, and Martin Tabler. Developmental defects by antisense-mediated inactivation of micro-RNAs 2 and 13 in *Drosophila* and the identification of putative target genes. *Nucleic Acids Res.*, 31(17):4973–4980, 2003.
- [34] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(10):993–1001, 1990.
- [35] Christopher Workman and Anders Krogh. No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. *Nucleic Acids Res.*, 27(24):4816–4822, 1999.
- [36] Ola Sætrom, Ola Snøve Jr., and Pål Sætrom. Weighted sequence motifs as an improved seeding step in microRNA target prediction algorithms. *RNA*, page rna.7290705, 2005. In press.
- [37] Lee P. Lim, Nelson C. Lau, Philip Garrett-Engele, Andrew Grimson, Janell M. Schelter, John Castle, David P. Bartel, Peter S. Linsley, and Ja-

son M. Johnson. Microarray analysis shows that some microRNAs downregulate large numbers of target mrnas. *Nature*, 2005. Epub ahead of print.

- [38] John G. Doench, Christian P. Petersen, and Phillip A. Sharp. siRNAs can function as miRNAs. *Genes Dev.*, 17(4):438–442, 2003.