Applying GAs in Searching Motif Patterns in Gene Expression Data

Project in bioinformatics

Tal Kaminer

Nir Laor

Advisers
Dr. Zohar Yakhini
Doron Lipson

Introduction

Differential expression of gene groups implies existence of a common control pattern. Such a control pattern is executed by binding of transcription factors to short conserved sequences in the gene's upstream region, called **Motifs**. Therefore, the existence or absence of these motifs in specific combinations is a definition of a control pattern.

Classified gene expression data consists of samples coming from several biologically meaningful classes (e.g. sample taken from cancer patients vs. those taken from healthy individuals). One central step in the analysis of such data is the ranking and scoring of genes according to their measured or inferred inter-class differential expression [6].

In this work we address the following general question: given a set of genes and their classification ranking, find a control pattern by identifying a combination of motifs that is unique to the control sequence of genes from one class.

Additional information on recent work in the field of identifying control patterns using clustering methods can be found in and Pilpel Y, Sudarsanam P & Church G [1]; Bussemaker HJ, Li H, Siggia ED.[2].

Motivation

Gene expression changes during cell's life, in accordance with:

- The cell's main function: for example, different genes will be expressed in brain cells and liver cells.
- The cell's condition:
  - Physical conditions: heat causes expression of 'heat shock' genes like chaperones. Chemical environment conditions like pH and ion concentration cause higher expression level of several sets of genes, including ion 'pumps' that are used to maintain regular conditions in the cell cytoplasm.
  - Cell age and stage in cell cycle: during mitosis, the cell expresses genes that code for chromosomal duplication and spindle creation. These genes expression level during other stages of the cell cycle is much lower. More data regarding cell cycle regulated genes in available in P. T. Spellman et al[3].
  - Disease: for example, one of the properties of a cancer cell is high expression level of genes that are responsible for cell division and low expression level of genes that control cell cycle.

Regulating Gene Expression

Gene expression is regulated in many levels, starting from changes in chromatin structure, through different transcription and translation stages, to post translation modifications.

The transcription initiation is a single-entry point control path, therefore it is the most common level of regulation. In this type of control only required genes will be transcribed to mRNA and later on to proteins.

The control region in transcription initiation regulation is the upstream sequence – the non-coding sequence before the gene. This sequence contains many short, conserved, regions called **motifs** (also referred as *cis*-acting sequences). Transcription factors, proteins that recognize specific sequences in the DNA, bind these upstream motifs and control gene expression in both positive and negative ways. There are two types of transcription factors – enhancers and inhibitors: The enhancement factors form the transcription initiation complex and enable strong binding of RNA Polymerase. The inhibition factors lower the affinity of the initiation complex for binding RNA Polymerase, or inactivate the Polymerase.

Since proteins generate both positive and negative control, the presence or absence of a motif affects the inhibition and enhancement of gene transcription, as summarized in the following table:

	Motif Absent	Motif Present
Enhancer Protein	⬇	⬆
Inhibitor Protein	⬆	⬇

Legend:

⬆ Positive control ⬇ Negative control

In eukaryotic cells, genes that have related function, such as genes that code for components of a metabolic pathway, are spread around the genome (as opposed to related genes in prokaryotic cells which are clustered in operons). The main mechanism that enables their synchronizes expression is the presence of the same motifs in their upstream control regions. Different combinations of motifs in the upstream sequence define different regulatory pathways.

Finding a combination of motifs in the control region of genes that are highly expressed in a specific condition can help identifying a control pathway that is responsible for this condition.

Another categorization of upstream elements is by their location relative to the transcription start site: motifs close to the start site (roughly 300 base pairs upstream to this point) are called ‘promoter motifs’. These motifs location is critical to the function they perform. The other group consists of motifs that are located far away from the transcription start site (usually up to 1000 base pairs). These motifs are called ‘enhancers’ although they can be both positive or negative control elements. The location of the enhancers is not important to their function, and in fact they perform the same function even when are synthetically moved thousands of base pairs away [4].

Definition of Boolean motifs and tree representation

Boolean motif definition

A **motif** is composed of:

- Sequence - a short word over the alphabet {a, c, t, g and n}. The sequence length is variable in the range of 4 to 10 terminals. The terminal ‘n’ is used as a wild card to allow mismatches and sequence diversity. The number of allowed wild cards in a sequence is a function of the sequence length.
- The sequence location in the upstream region. The location is measured in base pairs from the transcription start site.
- Offset – used as a relaxation parameter for the location condition. A motif appears in the gene’s upstream region if the sequence (with possible mismatches) is found in the interval [location – offset, location + offset].
  The offset is a function of the location (currently, the offset is 20% of the sequence location). This is based on the division of motifs to promoters and enhancers.

This is a simplistic view and there is no real “binary” control pattern. We do take this view, however, to facilitate some of our calculations.

Solution definition

The biological model for this work defines a control pattern as a logical expression over the motifs in the gene’s control region. Each literal in the logical expression is a motif, representing the Boolean question “is the sequence present at this location?”. The logical expression may consist of any number of motif literals and the AND, OR and NOT operators.

Each component of the logical expression has an **Appearance Profile.** The appearance profile is a binary vector in the length of the data set (number of input genes).
Each entry in Motif appearance profile represents whether the gene contains the motif in the specified location.

The appearance profile for each operator is calculated by applying the logical operator on the appearance profiles of its sub expressions. Each entry in the vector represents whether the gene satisfies the expression, i.e. contains a set of motifs in the specified locations, according to the relations defined by the logical expression.

The logical expression is represented by as a binary tree, where a leaf represents a motif literal and an inner node represents a logical operator. The tree structure impose a hierarchy in the logical expression: the appearance profile of the expression, by which the expression fitness is evaluated, is calculated from leafs level up to the tree root, applying the logical operators in the inner tree nodes.

The full logical expression of the solution is represented in the tree root node. Therefore, the solution appearance profile and fitness score are taken from the tree root.

The major feature of defining the solution as a regular expression is its flexibility. By defining a general scheme rather than a specific solution structure, the results are not restricted to the current biological knowledge.

Biological definition and meaning of logical operators

The “AND” operator forces a correlation between the appearance profiles of it’s two sub expressions. In other words, the appearance of both sub expressions is essential for the simultaneous expression of the genes. This interpretation assumes an obligatory composition of transcription factors that interact with each other in order to express the gene under a specific condition.

The “OR” operator doesn’t force a strict correlation between the appearance profiles of the sub expressions, although such a correlation may exist. The biological meaning of the OR operator is that the appearance of any one of the sub expressions is a sufficient criteria for co-expressing the gene along with other genes in response to a specific cell condition.

The “NOT” operator and its sub expression can be interpreted as either inhibitory control components or as exhibitory control components of the other genes group.
The biological meaning of interpreting the NOT expression as an inhibitory control components is the existence of inhibitory transcription factors that specifically recognize and bind the motifs in the NOT expression. This transcription factors interfere with the creation of an active transcription complex, thus preventing the expression of the gene.

The “NOT” operator applies a bit-wise logical “not” action on the appearance profile of an expression, therefore the expression can be interpreted as control pattern over the other group of genes. This result is highly important when using an a-symmetrical fitness function (such as the Accumulative Hyper Geometric probability function used in this work, see in the fitness function section below) to rank the solutions.

The computational problem

Given a set of genes control sequences and their classification ranking, the task is finding logical expressions that in turn define control patterns.

The motif literals are all the words in the length of 4 to 10 letters over an alphabet containing 5 letters.

The search space is defined as all the logical expressions over motif literals, using the logical “And” “Or” and “Not” operators.

Methods

The genetic algorithm

The search space, defined by the biological model, is too large for exhaustive search. Therefore, the genetic algorithm is used as a search heuristic.

The genetic algorithm is a heuristic search method that doesn’t always find the best solution, but always converges to a local maximum in the scores plane, without computing all possibilities. This algorithm imitates the natural selection by maintaining a population of solutions, which is changed each generation by recombination and mutation events. The first population and most of the ‘breading’ events made during the algorithm iterations are generated randomly.

The genetic algorithm in the current work is performed in an altruistic – generation mode, in which a certain amount of good solution are transferred “as is” from the previous population to the next one, while all other solutions in the population are created anew each iteration, using genetic operators on good solutions in the previous population.

Fitness function - mHG scoring

The appearance profile has a Fitness Score, defined as the accumulative hyper geometric probability to create such a binary vector.

According to the hyper geometric probability function, the probability of creating a binary vector, with all the 1’ in it clustered together close to one of the edges is low. An appearance profile that shows this property is created by a combination of motifs that appear only in one of the ‘original’ classes.

As the probability to create the profile at random decreases, it is more likely that the solution that created this appearance profile is related to biological phenomena.

Note: it may seem that the a-symmetrical nature of the accumulative hyper geometric function will lead to the identification of control patterns in one group only. This problem is solved by using the “NOT” operator.

Scoring function

Genetic algorithms rapidly push the population towards convergence. That is, all individuals in the population soon become nearly identical. Even when multiple solutions to a problem exist, a genetic algorithm based on solutions fitness locates only one of them.

In order to avoid this problem, the solution score is composed of two parameters [5]:
Fitness – describes how well the solution separates the two classes.
Diversity – a measure to the solution uniqueness in the population.

The solution score is calculated by the following formula:

$$score = \frac{(fitness)^{fit\_weight}}{(diversity)^{div\_weight}}$$

Where fit_weight and div_weight are configurable parameters, used to balance between the two separate objectives.

Diversity score

The solution diversity score is relative to the current population. The score is calculated by summing over the number of appearances of each node (sub solution) in the population

Genetic operators

The genetic algorithm uses mutation and recombination operators on solutions, in order to create a new population based on the previous iteration population.

When applying a genetic operator on a solution, the operator is applied on one of the solution tree nodes only. This node is selected using a random selection method, taking into account the node depth.

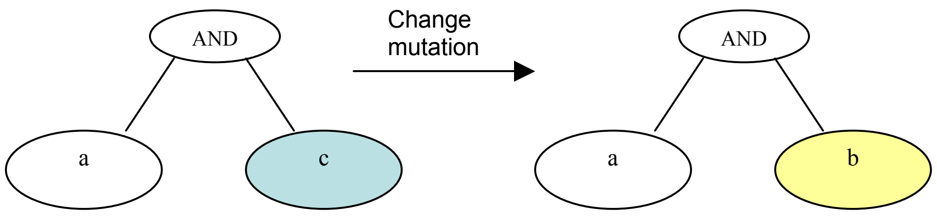
Mutation

The mutation operator is applied on a single solution each time, and changes the solution according to the selected mutation type:

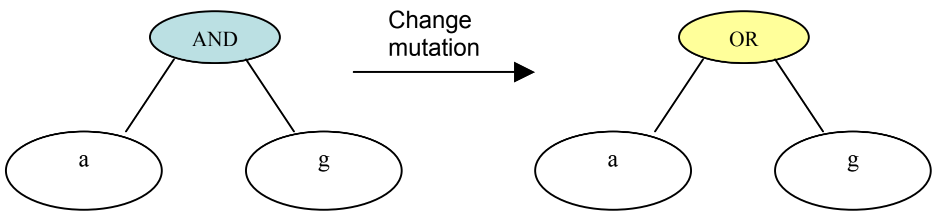
Change mutation – changing a tree node data. This mutation may change the sequence or location in a leaf (motif) node, and the logical operator in an inner tree node.

Figure 1: change mutation

I. Change mutation on a leaf



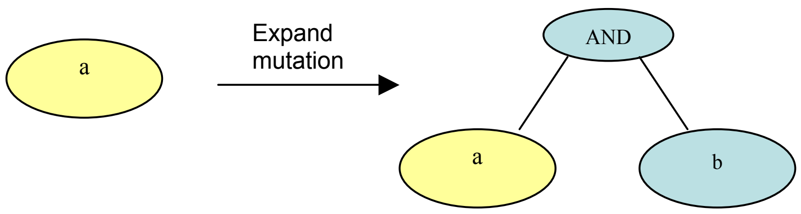
B. Change mutation on an operator node



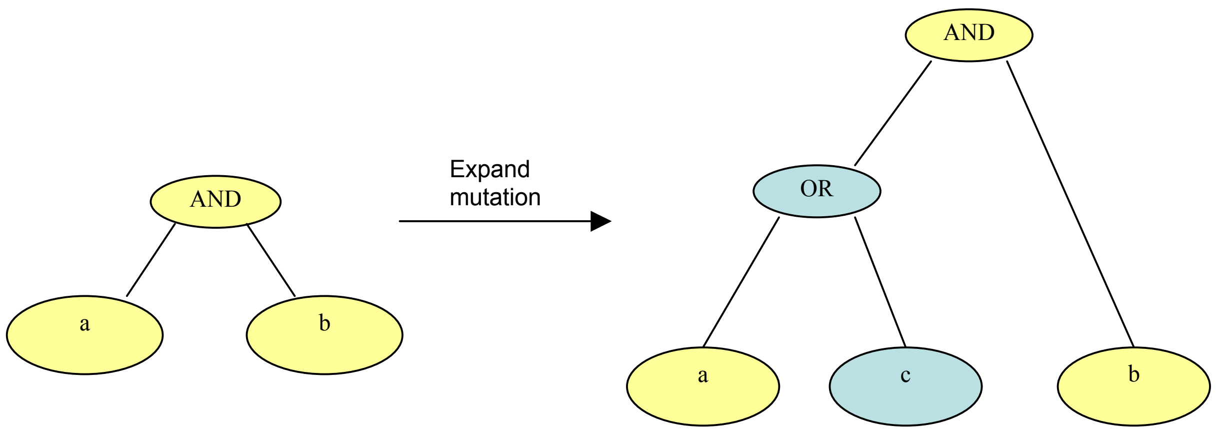
Expand mutation – expanding the node. In this mutation type, a new “father” node is randomly created (possibly with a new random “brother” node) for the mutated node.

Figure 2: expand mutation

A. Expanding a leaf

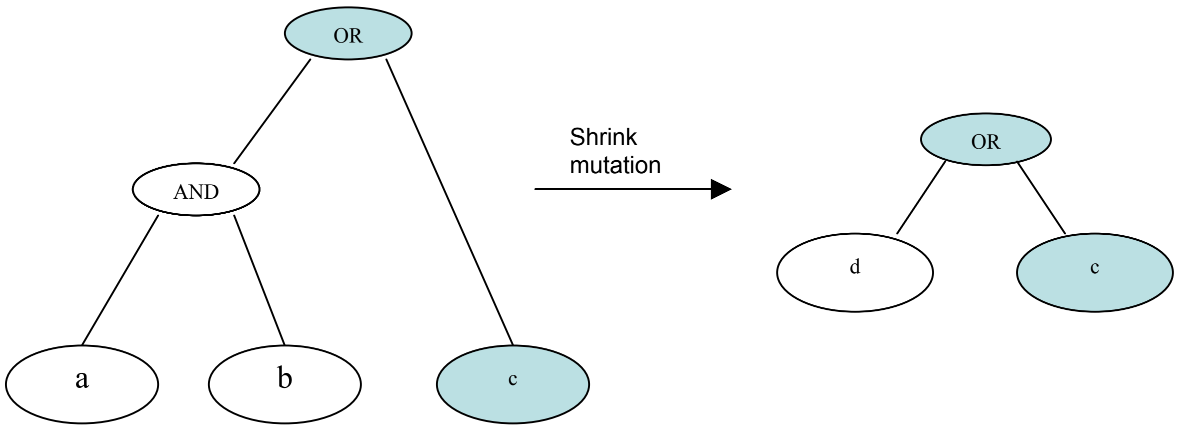


B. Expanding an inner node.



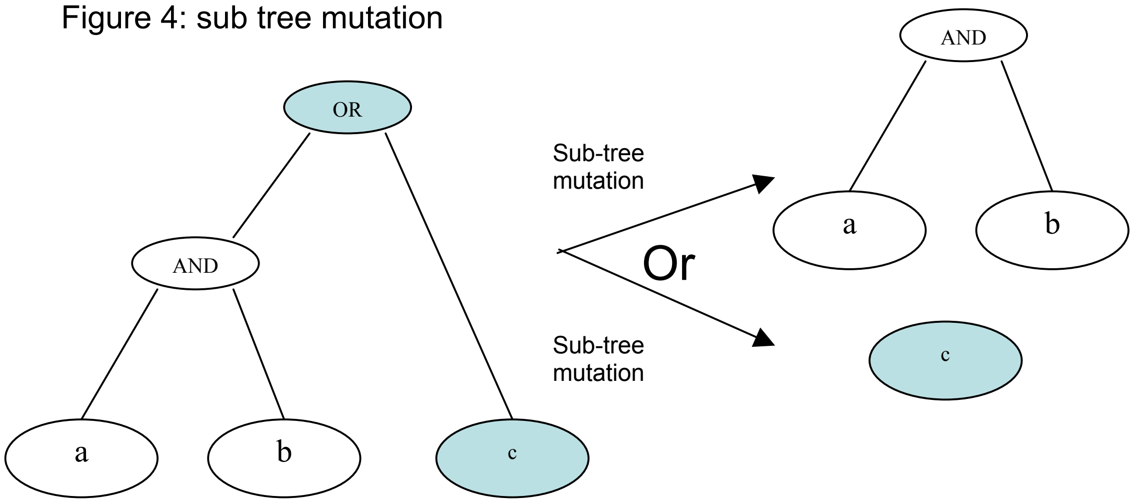
Shrink mutation – replacing a sub tree with a new random leaf. This mutation allows the solution to “loose” unwanted and un-valuable parts.

Figure 3: shrink mutation



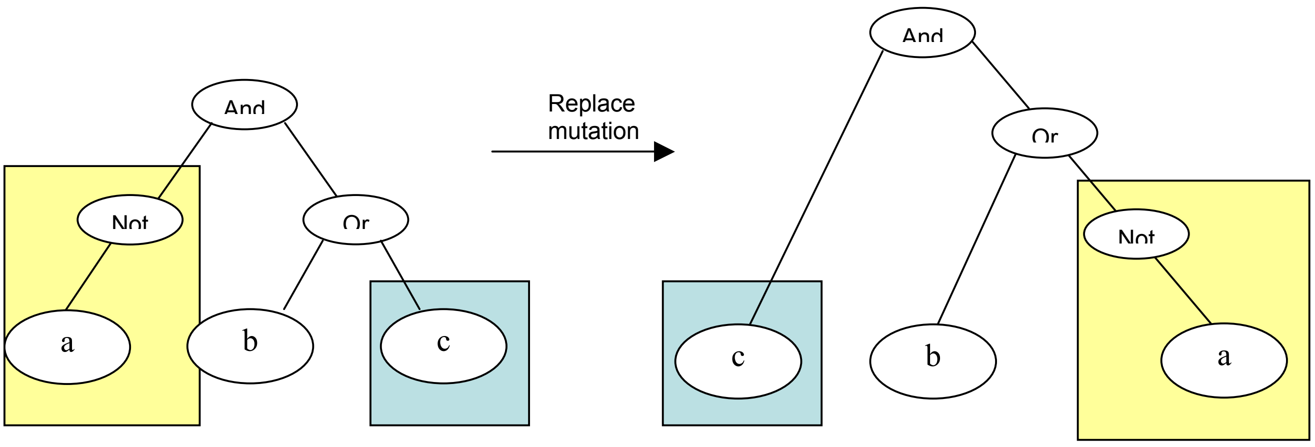
Sub tree mutation – replacing the solution with one of its sub solutions / replacing the tree with one of its sub trees. This mutation, as the shrink mutation, allows the solution to “loose” unwanted and un-valuable parts.

Figure 4: sub tree mutation



Replace mutation – replacing two nodes (either leafs or inner nodes) in the tree. This mutation rearranges the logical expression and thus causing a change in the appearance profiles of (at least) two nodes in the tree (note that the appearance profiles are calculated from the leafs level up to the root, therefore swapping leafs or sub trees may change the appearance profile). This mutation can be viewed as inner recombination.

Figure 5: replace mutation



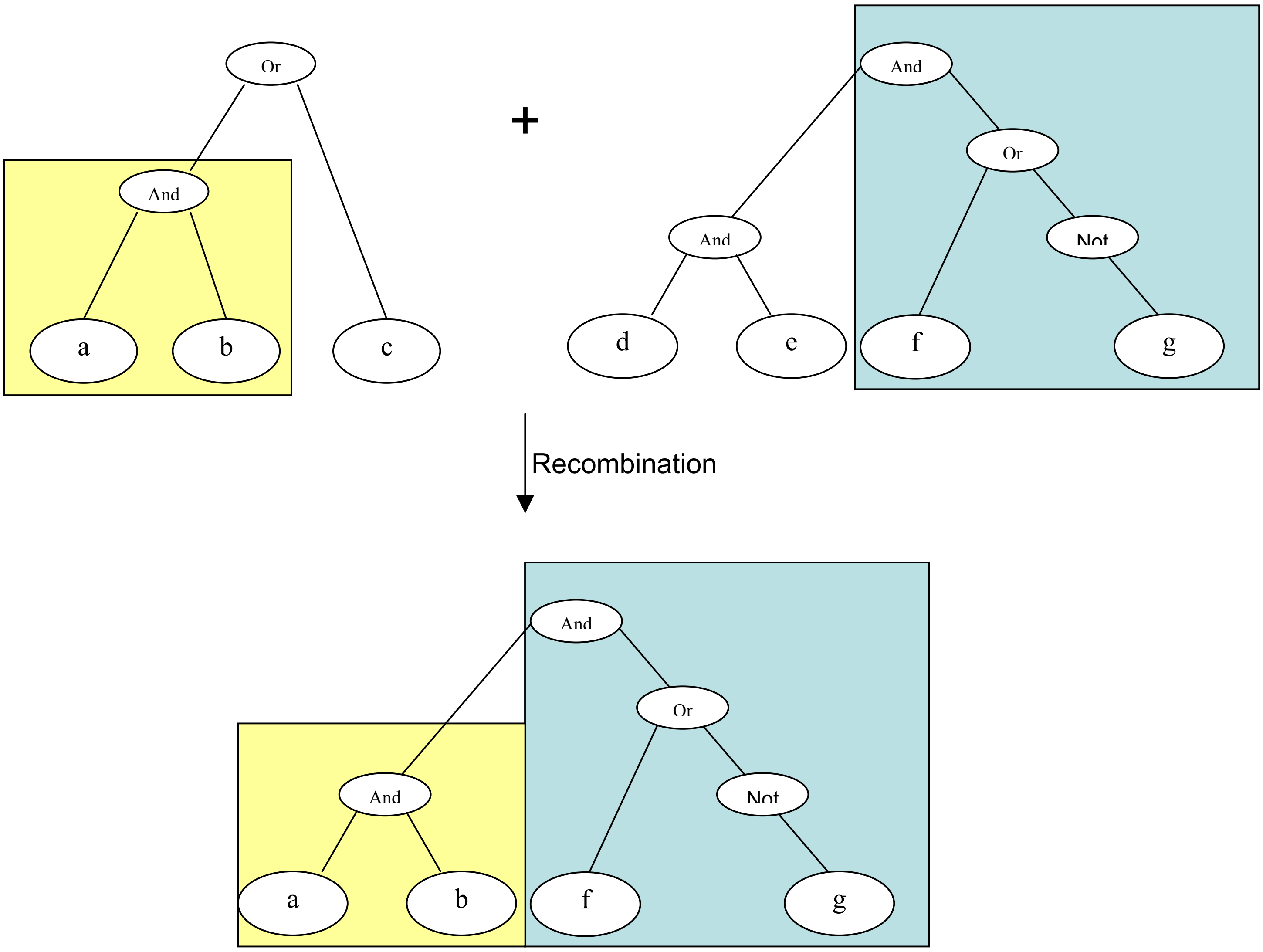
In order to preserve a reasonable and understandable solution size, the available mutation types and the mutation frequencies for a solution are a function of the solution size. For example, the “expand mutation” won’t be available when mutating a solution with size that passes a certain threshold. On the other hand, a “shrink mutation” won’t be available for too small solutions.

Recombination

The recombination operator creates a new solution by replacing a sub solution taken from one solution with a sub solution taken from a different solution. The new solution is a combination of its “parents” and is different from both of them.

In the recombination operator, as in the mutation operator, the solution size is maintained in order to prevent the creation of large-scale solutions. This is achieved by allowing only a small difference in the replaced sub solutions size.

Figure 6: recombination



Implementation

Data base preprocessing

The source file, a tab delimited file with gene's names, scores and upstream sequences, sorted by the scores, is uploaded to the program on startup and is processed before the Genetic algorithm is started.

Each upstream sequence is represented as an array, indexed by subsequences from "aaaa" to "tttt" where each entry in the array is a list of all appearances of the short sequence.
The index of each appearance of a subsequence is stored in the lists array while the upstream sequence is read one time only. Subsequences with mismatches are generated by replacing known nucleotides with "n" signs and their index is also stored in the database.

Indexing the appearances of all short subsequences allows the fast calculation of motif appearance profile. The appearance profile of a longer than the stored subsequence length is easily calculated by disassembling the motif sequence to short (possible overlapping) sequences and building the appearance profile for each such sequence.

Each node has a unique hash code, generated using the node data (motifs, locations etc.). This hash code is used for three population manipulation and control objectives:

1.0 Solution hash – storing the hash codes of the solutions (tree roots). This data structure is used in order to prevent the appearance of identical solutions in the population.
2.0 Profile DB – this data structure stores the appearance profile of all the nodes in the population (tree roots and inner nodes, down to the leafs level). Since the genetic algorithm reuses nodes, storing and reusing the appearance profile of a node and its fitness score dramatically reduces the number of appearance profile and fitness calculations which is a time-consuming action.
3.0 Nodes reuse counter – in order to calculate the solutions diversity (as a part of scoring the solution, see Scoring Function section above), the number of reuses of each node is stored in the data structure. This data structure allows fast calculation of the diversity score of a solution, by summing over the number of appearances of its nodes in other solutions.

<u>Software scheme:</u>

1. Read the data file and preprocess data
2. Create an initial population of random solutions (may be either motifs-only population or random trees population).
3. Run the genetic algorithm:

3.1. Calculate scores for all solutions in the population, based on the solution fitness and diversity.

3.2. Create a new population of solutions, using genetic operators like mutation and recombination on highly scored solutions from the previous population.
Each iteration, a whole new population is created, except for a (configurable) number of “altruistic” solutions, transferred “as is” from the previous population.

Synthetic data model

A synthetic data with planted known solutions was generated in order to test the genetic algorithm and score parameters. The data was generated according to a probability model assuming a high appearance level of control-pattern motifs in the highly scored genes, decreasing in the control regions of genes with lower expression scores, down to random appearance pattern in the control regions of the other group of genes.

The generating function is $plant\_prob = \frac{1}{1+e^{a(b-x)}}$ where x is the sequence index (in the data file).

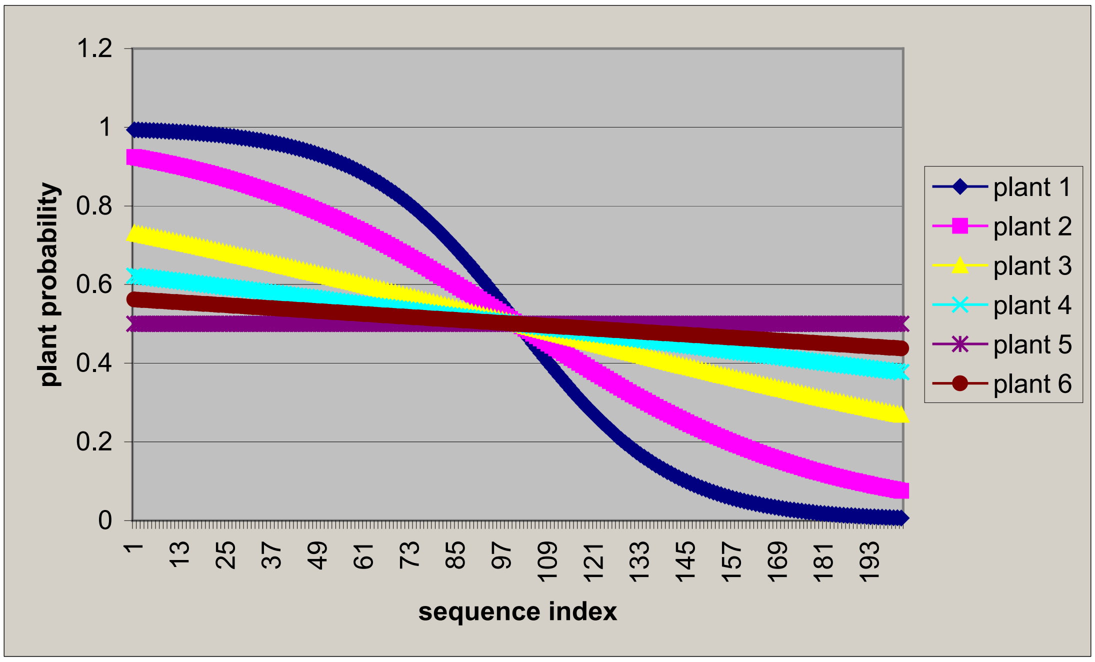
- The “a” parameter controls the graph slope, thus changing the appearance ratio between the two groups.
- The “b” parameter shifts the graph middle point, thus controls the separation point between the groups.

Six data files, each containing 1000 random sequences in the length of 500 bp, were generated using the probability function. The appearance ratio between the groups was changed by changing the “a” parameter (see table 1 for “a” values), while the “b” parameter remained constant (having the value 100), thus creating a synthetic file of 100 genes from one class and 900 genes from the other class. See graph 1 for plant probability curves.

Table 1: plant parameters

Plant number	A
1	-0.05
2	-0.025
3	-0.01
4	-0.005
5	-0.0025
6	0

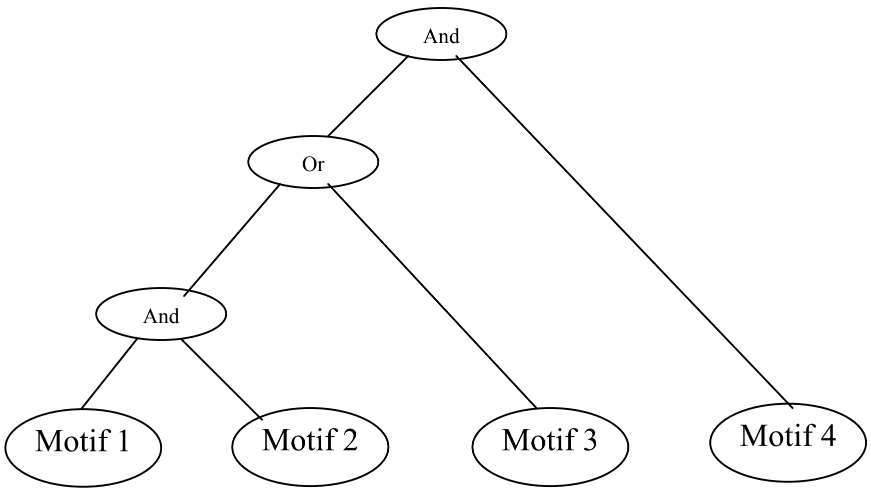
Graph 1: synthetic data plant parameters



The planted solutions are

- “Or” tree: composed of two motifs.
- “And” tree: composed of two motifs.
- Complex tree: composed of 4 motifs in the following structure:
  (((motif1 And motif2) Or motif3) And motif4).

Figure 7: The planted solution.



The data creation algorithm is as follows:

```
For each sequence i:
            If node.operator == OR
                  If (plant probability (i) < random number)
                        Plant left sub expression.
                  If (plant probability (i) < random number)
                        Plant right sub expression.
            If node.operator == AND
                  If (plant probability (i) < random number)
                        Plant both sub expressions.
```

Results

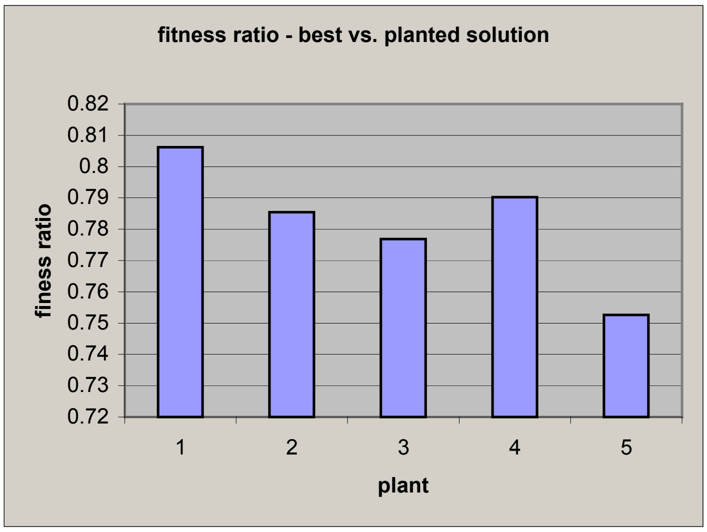
On synthetic data, range of parameters

The software sensitivity can be defined as the minimal appearance level of the planted solution above the background level, in which the planted solution can still be found. This threshold is a function of the solution complexity and structure, and therefore it is not possible to define its absolute value.

In order to identify the sensitivity threshold, the results of processing the synthetic data were evaluated using two criteria: the fitness ratio of the best solution found and the planted solution fitness, and the number of planted motifs found.

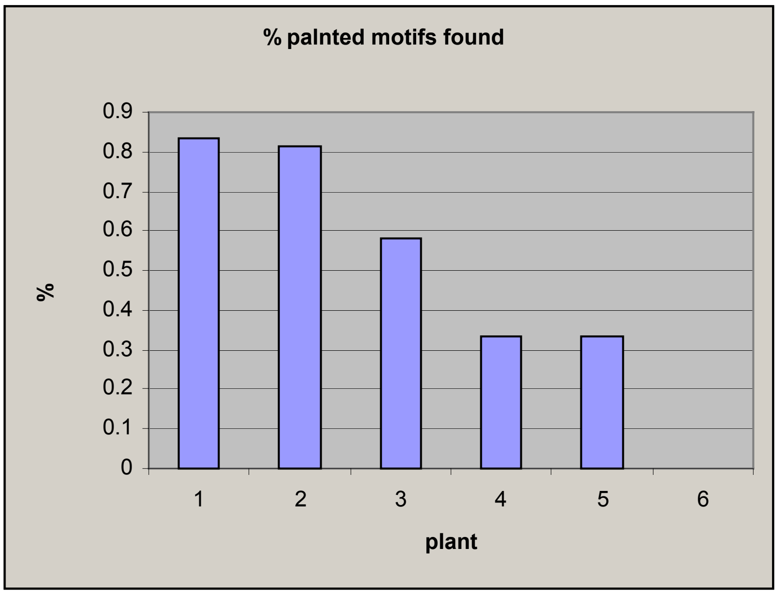
The results of processing the “complex tree” data files are summarized in graph 2 and 3:

Graph 2: average fitness ratio between the planted solution and the best solution found in the complex tree data files.



Note that although there are differences between the planted solution score and the best solution found the best solution was in most cases the planted solution, with small variations (inserting a mismatch terminal in the motif sequence, a small change in the motif location, etc.).

Graph 3: percentage of planted motifs found while processing the complex tree data files.



The software sensitivity for the smaller “And” and “Or” trees was identified as the probability parameters of plant 5 (a = -0.0025). The sensitivity for the complex tree is much lower – it is defined by the probability parameters of plant 3 (a = -0.01).

On yeast data

The expression level of almost every gene in the yeast *Sacchromyces cerevisiae* was measured over time in response to heat shock in an experiment conducted at the Stanford laboratories (http://genome-www.stanford.edu/). The differential expression score was calculated by subtracting the expression level of the gene measured in two time marks. The following results are based on the expression scores calculated using the expression level measured in the experiment first minute (0 minutes) and 60 minutes after the hit shock.

Using the upstream sequences of yeast genes, downloaded from SGD FTP site (http://www.yeastgenome.org/), the yeast expression data was processed and the results were compared with results of processing random data generated with the same statistical parameters (nucleotides ratios, scores, etc.). There is a clear statistical difference between the solutions fitness achieved when analyzing the yeast data and the random data – the average fitness of the best yeast solutions is more than 2.5 times higher than the average fitness of the best random solution.

The Heat Shock transcription Factor HSF1 was found in 3/4 of the program runs. Another, unknown, motif that re-appeared in the results is “cccct” found in the interval [220 - 320] base pairs upstream to the transcription start site. This motif and HSF1 motif were found in opposite classes.

Discussion and future work

In this work, we have shown that it is possible to identify regulatory motifs and logical combination of motifs in genes upstream data, using genetic algorithm.

The great strength of this work is the flexible structure of the solution. This general structure gives it the potential to identify control patterns defined by different compositions of upstream motifs, without being tight to the current biological data. However, the results of the software must have a post processing stage in order to remove false positives that are inserted to the results set as a part of a fitted solution. Such a post processing stage should re-evaluate the contribution of each motif to the solution and remove unfitted motifs.

References

[1] Pilpel Y, Sudarsanam P & Church G, Identifying regulatory networks by combinatorial analysis of promoter elements. Nature genetics vol. 29 pp153-157 (2001).
[2] Bussemaker HJ, Li H, Siggia ED, Regulatory element detection using correlation with expression, Nat Genet. 2001 Feb;27(2):167-71.
[3] P. T. Spellman et al**,** Comprehensive identification of cell cycleregulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization," Mol. Biol. Cell, vol. 9, pp. 3273--3297, 1998.
[4] Lewin, Genes 7th edition (2000).
[5] Samir W. Mahfod, Niching methods for GA's, 1995
[6] Amir Ben Dor, Lauraukay Bruhn, Nir Friedman, Iftach Nachman, Michel Shummer & Zohar Yakini , Tissue classification with gene expression profiles (2000).