# Creating Regular Expressions as mRNA Motifs with GP to Predict Human Exon Splitting

Technical Report TR-09-02, 19 March 2009

W. B. Langdon, J. Rowsell, A. P. Harrison

Crest, Department of Computer Science, King's College, London, Strand, London, WC2R 2LS, UK
Departments of Mathematical and Biological Sciences, University of Essex, UK

## ABSTRACT

Low correlation between mRNA concentrations measured at different locations for the same exon show many current Ensembl exon definitions are incomplete. Automatically created patterns (e.g. `TCTTT`) in genic DNA sequences identify potential new alternative transcripts.

Strongly typed grammar based genetic programming (GP) is used to evolve regular expressions (RE) to classify gene exons with potential alternative mRNA expression from those without. RNAnet gives us correlations between Affymetrix HG-U133 Plus 2 GeneChip probe measurements for the same exon across 2757 Homo Sapiens tissue samples from NCBI's GEO database. We identify many non-atomic Ensembl exons. I.e. exons with substructure. Biological patterns can be data mined by a Backus-Naur form (BNF) context-free grammar using a strongly typed GP written in `gawk` and using `egrep`. The automatically produced DNA motifs suggest that alternative polyadenylation is not responsible. (Short version in [19].)

The training data is available on the internet.

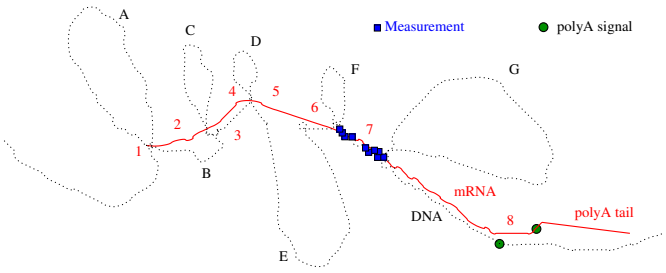



**Figure 1: DNA (dotted line) is transcribed into mRNA. Initially all the gene's DNA is copied, but loops (A to G) in the RNA are excised. I.e. they are introns. The remainder (1 to 8) are exons. Alternative splicing might remove exon 3, leading to an mRNA molecule containing exons 1245678. In the schematic, the polyA tail has been added after the $2^{nd}$ polyA signal. If the $1^{st}$ polyA site was active, exon 8 would be shorter and the mRNA might be translated into a different protein. Again for illustrative purposes, DNA GeneChip measurements are scattered along exon 7. If exon 7 is "atomic" then all measurements should be the same and be highly correlated.**

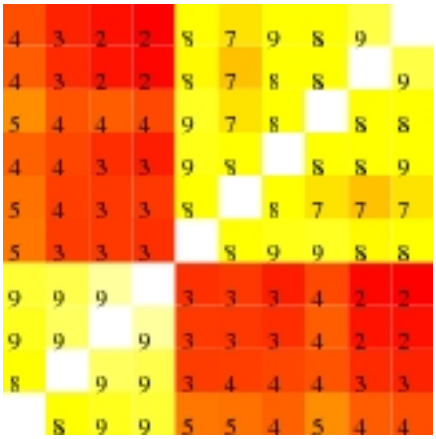


**Figure 2: Correlation ($\times 10$) between different locations in an Ensembl exon across 2757 tissue samples. The first four locations (lower left) clearly fall into a different block than the others (top right).**

## 1. INTRODUCTION

Following the Human Genome Project, a great deal is known about gene DNA sequences. A number of sites on the Internet provide free access to their databases. E.g. Ensembl provides definitions of all Human genes in terms of their locations, DNA sequences and current knowledge of intron and exon locations in the known gene transcripts.

RNAnet (`http://bioinformatics.essex.ac.uk/users/wlangdon/rnanet/`) contains measurements of genes activity from thousands of people from a wide variety of tissues and disease states. Typically there are about 11 measurements for each gene/tissue sample. These are designed to target the gene's exons. Typically there are either multiple data per exon or the exon is not targeted at all. Figure 1 shows the relationship between DNA, mRNA and measurements. In Figure 1 all the measurements are in exon 7.

The process whereby DNA is transcribed into mRNA is fairly well known. In particular it is known that the complete DNA sequence is transcribed into RNA but large loops (known as introns and shown in Figure 1) are rapidly removed. The remaining transcript between the introns are known as exons. Recently it has become clear, particularly in Man, that this is not the end of the story. Sometimes there are different transcripts for the same gene which have different sequences of the gene's exons. Alternative splicing may lead to the removal of exons or repeats of exons. Notice

that exons are treated as atomic. I.e. indivisible.

If an exon is present in mRNA, then all of it should be present. Therefore measurements from the same exon/tissue sample should be the same (ignoring noise, calibration error, etc.). We use correlation between measurements for the same exon to see if they really do tell the same story about their exon. Correlations have the advantage that they provide a robust statistic. By calculating the correlations across thousands of tissue samples, we alleviate the effects of noise. In most cases, using Ensembl's definitions of exons, we do indeed find high correlation between data for the same exon. We also find many cases where measurements across an exon are not well correlated. Instead sometimes, like that shown in Figure 2, we find blocks of neighbouring measurements which are well correlated but poorly correlated with other measurements. We suggest that perhaps such Ensembl exons have been mislabelled.

Having identified these DNA sequences, we try to find explanations looking initially only at the mRNA sequences. Since we hope to find biological meaningful reasons we look for regular expressions of the type biologists commonly use when describing sequence motifs. Our genetic programming system is designed to automatically generate DNA sequence motifs. To evaluate their fitness, we test the evolved motifs on the DNA sequence for the gap between highly correlated blocks within an exon. A motif gets high fitness if it matches many blocky exons (i.e. positive examples) but fails to match many exons with high correlations but without blocks (negative examples).

We evolve a new biological motif and show the existing polyA motif [28, Fig. 3] could only explain at most a fraction of the observations.

The next section summarises the use of GP and grammars, particularly for evolving bioinformatics solutions. Section 3 outlines the grammar based GP. Section 4 describes how the training data was prepared for the GP (which is described in detail in Section 5). The evolved motif and its performance on out of sample data is given in Section 6. This is followed by a discussion (Section 7) and our conclusion (Section 8) that the correlations in vast public data sets can give valuable hints of novel biology and that evolutionary computation, if couched in biologist friendly terms, may help them interpret it.

## 2. EVOLVING GRAMMARS AND PROTEIN MOTIFS

Existing research on using grammars to constrain the evolution of programs can be broadly divided in two: Grammatical Evolution [30, 24], a linear GP, and work by Whigham, Wong and McKay using typed tree representations. See, for example, [35, 36, 37] and [21].

Hoai *et al.* used Tree Adjunct Grammars (TAGs) [8]. TAGs have the advantage that every subtree of a TAG tree is both syntactically and semantically meaningful. This should ensure that there is much more flexibility for crossover and mutation to transform TAG trees.

In the spirit of PBIL [31] a number of people have allowed progress so far to update the grammar as evolution proceeds [27, 33, 2].

There is quite a body of work on using evolution to induce formal grammars. E.g. Nikolaev tackled the Tomita regular expression benchmarks [23]. GP has also evolved context free grammars [11]. Cetinkaya used grammatical evolution to create regular expressions for processing HTML [5].

Ross induced stochastic regular expressions from a number of grammars to classify proteins from their amino acid sequences [29]. Regular expressions have been evolved to search for similarities between proteins, again based on their amino acid sequences [6]. Whilst Brameier used amino acids sequences to predict the location of proteins by applying a multi-classifier [15] linear GP based approach [3] (although this can be done without a grammar [13]). A similar technique has also been applied to study microRNAs [4].

Non-stochastic machine learning techniques have also been applied to DNA motifs. E.g. [10] presents a method based on decision trees, specifically C4.5. However the Tomita *et al.* motif uses amino acid properties, not regular expressions of DNA sequences [34].

Both GPRM [9] and the more recent GeRNAMo [22] are designed to predict RNA secondary structure, i.e. hairpin folds. To do this GeRNAMo must be given some prior information. In particular it needs to be told how many double helixes (stems) the folded RNA contains. (This is also required by GPRM, however Yuh-Jyh Hu advocates running GPRM multiple times with different numbers of stems until the required number can be inferred from GPRM's answers [9, p3447].) GeRNAMo creates folding predictions directly using a strongly typed GP with a function set which describes RNA folds and does not use a grammar or evolve regular expressions. While folding of mRNA might be responsible for the sub-exon blocks this is not known. Consequently the number of stems in unknown. Therefore we need a less restricted system.

## 3. EVOLVING DNA MOTIFS WITH STRONGLY TYPED GP

As with [25] we distinguish between genotype (a strongly typed tree) and the phenotype. The phenotype is a regular expression suitable for use with `egrep`. The GP performs genetic operations (i.e. crossover and mutation) upon the genotype which is then treated as a grammar and expanded. The leafs of the grammar expansion form the phenotype. (Grammar GPs are described in [18].) We use the BNF grammar shown in Figure 6, and so we are guaranteed that all the evolved phenotypes will be legal regular expressions which can be executed by `egrep`. Creation of the initial population, genetic operations and fitness selection are all performed under Linux by shell scripts using `gawk`.

In [16] we applied the system to model Affymetrix GeneChip technology. This has the advantage that the target sequences all have the same length (25 bases). (And thus also two ends). Here we are dealing variable amounts of mRNA, from a few bases to more than a few thousand bases. It is also not obvious where the ends of affected transcript should be placed.

## 4. PREPARATION OF TRAINING DATA

RNAnet contains 7 Gbytes of data derived from forty thousand public Affymetrix microarrays [1, 20, 7]. In particular it holds data on 2757 HG-U133 Plus 2 Human GeneChips. We calculated the correlation heatmaps [20, 12] for every Ensembl exon for which there was HG-U133 Plus 2 data (excluding those which might refer to more than one Ensembl exon [32]).

Since previously we had found Mismatch MM control probes to be particularly susceptible to poor correlation with the rest of their probeset [12], they were ignored. Similarly we have previously [16] found Affymetrix PM probes matching `GGGG|CGCC|G(G|C){4}|CCC` also to be susceptible to error. Such probes may be producing valid signals, but to reduce ambiguity, they were also excluded. The correlations (http://bioinformatics.essex.ac.uk/users/wlangdon) were used to select suitable exons.

If an exon had at least six[1] non-overlapping pairs of probes whose correlations greater than 0.8, then the exon was passed to the next stage. This yielded 3373 exons with on average 12 probes per exon.

In the hope of finding more examples of blocky exons the correlations between probes in each exon were recalculated for eight subsets of the data each with 625 tissue samples. (625 is more than sufficient to ensure differences between correlations cannot be due to chance fluctuations). For the first four subsets the tissue samples were divided into four non-overlapping subsets using the existing experimental ordering. In the second group of four, the tissue samples were randomly split into four non-overlapping subsets of 625. Correlations for every pair of probes in every exon were calculated for each of the eight subsets.

For each subset, each exon was checked to see if it contained two groups of highly correlated probes (i.e.median value $> 0.7$) which are not correlated with each other (i.e. median correlation less than 0.4). In both cases, for the median to be considered, there must be at least three pairs of non-overlapping probes contributing to it.

The mRNA transcript between two such blocks is treated as a positive training example. Several exons give rise to more than one training example. Often these overlap. However we only use one case where positive examples from one or more of the eight data subsets are identical. Only if there are no positive examples from any of the eight data subsets is the exon considered as a negative example. This gave 375 exons with blocks (1184 positive examples) and the 2292 exons without. See Figure 3.

The closest block of correlated probes are only four bases apart but the most widely separated are eight thousand bases apart. It may be that these are really two different exons and Ensembl should not have classified them as being part of the same exon. Also it was felt that such a wide range of DNA base sequence lengths would make looking for simple motifs difficult. Obviously the longer a random base sequence is, the more chance it has of matching a given motif. Therefore we limited the blocks used in training to being closer than 100 bases. This reduces the number of potential positive examples to 274 from 140 exons, of these 100 were selected for training the GP. This leaves 40 exons with poorly correlated blocks close together for verification. Similarly 1536 exons without blocks were selected for training, leaving 756 for verification. This gives 195 positive sequences, cf. Figure 4. These are unique but in many cases overlap.

Many biological sequence signals do not appear exactly at the same point as the action takes place. For example, the polyA motif occurs up to 38 bases from where the mRNA is cut [28, Fig. 3]. Therefore we start positive training examples 50 bases before the last probe in the first well correlated block. Similarly each positive sequence is extended 50 bases after the end of the first probe in the next well correlated block. See Figure 5.)

[1]Six pairs means there may be enough data for at least two blocks of three probes.

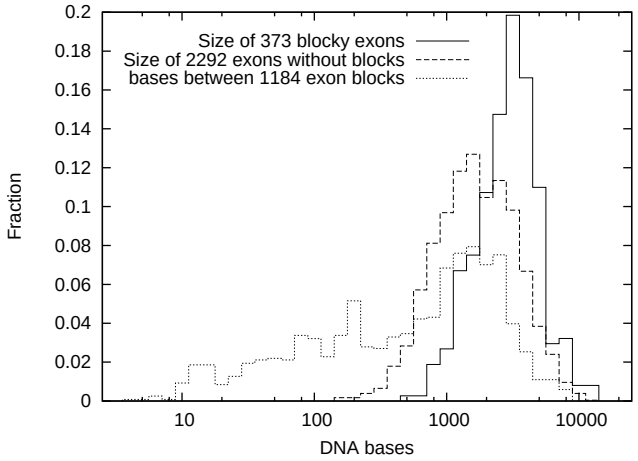



**Figure 3: Well expressed Ensembl exons (decibel bins). Notice exons with blocks (solid line) tend to be longer than those without (dashed) and the wide range of separation between blocks (dotted).**

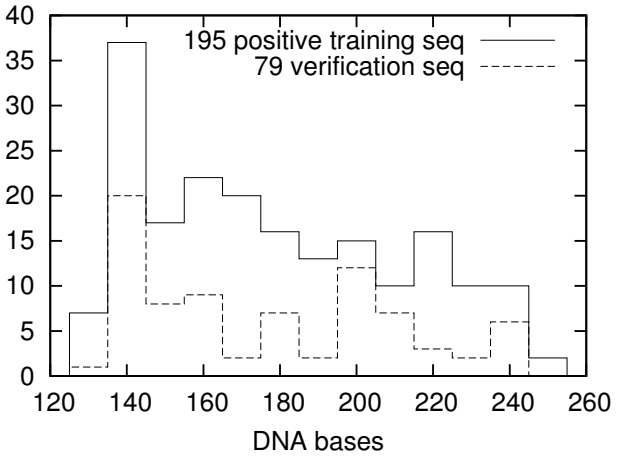



**Figure 4: Length of DNA sequences used for training. (Positive and negative examples are paired, thus negative examples are the same length as positive examples.)**

The 195 positive training examples are placed in a random order in a ring. Each generation the next 100 positive examples are drawn from the ring and 100 matching negative examples are constructed. If a positive example starts near the beginning of its exon (i.e. $\leq 300$ DNA bases from its start) then the negative example is simply cut from one of the suitable exons without blocks starting the same distance from the start and containing the same number of DNA bases as the positive example. For each positive example a randomised ring of suitable (i.e. having the same length within a factor of $\frac{1}{2}$ or 2) negative exons is used. However there are a few very long positive sequences for which there are few suitable exons without blocks to pair against. Therefore if the positive example starts more than 300 bases from the start of its exon, the negative example is

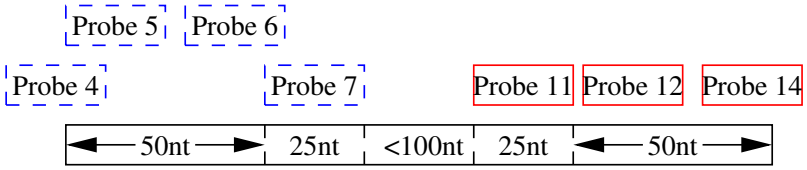



**Figure 5: Example positive training mRNA sequence (black). On average probes 4–7 (dashed) are well correlated (i.e. $> 0.7$) as are probes 11-14 (red). However the median correlation between left (dashed) and right (red) probes is less than 0.4. To reduce the chance of random matches the distance between blue and red probes must be less than 100 bases. Since the signal causing an exon to be cut may be some distance from the cut point, "Collars" of 50 nucleotide bases are added both upstream (left) and down stream (right) of the gap.**

**Table 1: Strongly Typed Grammar GP for Exon Splitting Prediction**

| | |
|---|---|
| Primitives: | The function and terminal sets are defined by the BNF grammar (cf. Figure 6). BNF rules with two options correspond to binary GP functions. The rest of the BNF grammar correspond to GP terminals. |
| Fitness: | true positives+true negatives. (I.e. proportional to the area under the ROC curve or Wilcox statistic. [14].) Less large penalty if `egrep` fails or it matches all probes or none. |
| Selection: | best 2000 become breeding population for next generation. Population size = 10000 |
| Initial pop: | Ramped half-and-half 3:7 [26, Sect. 2.2] |
| Parameters: | 90% subtree crossover. 10% mutation subtree 5% shrink 5%). Max tree depth 17 (no tree size limit) |
| Termination: | 50 generations |

still the same length as the positive example but is copied from a random starting point in the negative exon. Thus the 195 positive examples are reused about 26 times during the 50 generations of the GP run but the GP meets each negative example only once.

For validating the evolved regular expressions the remaining 40 exons with blocks were held back. Together they contain 79 unique sequences. For each a negative sequence starting the same distance from the start of the exon and of the same length was extracted from one of the exons without blocks. Except for one very long exon with two close block near its end, 78 unique negative exons were used. There are no negative examples long enough to cope with the special case, so a sequence of the right length was cut from a random position from a negative exon on similar size. All the validation examples are unique.

# 5. EVOLVING MOTIFS

## 5.1 Setting up Genetic Programming

The genetic programming system is a strongly typed tree GP system. The left hand side of the BNF grammar rules specify the nodes in the tree. The type of the node is the name on the left hand side. Thus, as we can see from Figure 6, we have many more types than is common in GP.

The right hand side (the grammar productions) say what must be attached to nodes of tree. The number of productions can be thought of as the node's arity. For simplicity, the right hand side will either have exactly two alternatives (indicated by a vertical bar, e.g. the right hand side of `<RE>`) or none. For example, the three items (`[`, `<set-items>` and `]`) must always be attached to node `<positive-set>` in the geneotype tree. `[` and `]` are leafs of the genotype tree and nothing further is attached to them but `<set-items>` is another rule which must be expanded.

Notice the grammar is recursive and so can be expanded to become arbitrarily big. In common with Koza's Lisp GP, we impose a depth limit both during the creation of the initial random population and during subsequent evolution. Unlike [24] genetic operations must respect the strong typing rules. E.g. when crossover cuts a subtree from one parent it first chooses a node in the genotype tree. The node's type is the name of the corresponding grammar rule in the BNF. It then finds all nodes of the same type in the other parent and with equal probability randomly uses one. It then replaces the node and all subtree in the first parent with the subtree from the second parent. This ensures the new genotype formed by crossover still obeys the BNF rules. Mutation and the random construction of the initial population operate similarly.

To calculate an individual's fitness its phenotype must be constructed. We start with an empty regular expression. The geneotype tree is processed in a single pass in depth first order starting at its root. (I.e. processing the genotype tree starts at node `<start>`.) Each time a terminal is encountered it is appended to the regular expression. Even though the grammar is fully recursive, depth (or size) limits ensure the genotype is finite and therefore so is the phenotype. When the whole genotype has been processed, the phenotype is complete and its fitness can be calculated. Note the genotype, not the phenotype, is inherited. For efficiency, the phenotype of every individual in the population is created and then the fitness of the whole population is calculated.

Tournament selection is used to choose parents for the next generation. 90% of children are created by subtree crossover, 5% by subtree mutation and 5% by shrink mutation [26, 11]. See also Table 1.

The grammar (Figure 6) is devised to ensure that GP evolves regular expressions which can be executed by `egrep`. Unlike the short GeneChip probes used in [16], these DNA sequences come from the middle of live organisms (Man) and do not have natural starts or end. Therefore the regular expression end symbols (`^` and `$`) have been removed from the grammar.

Notice that the extended Kleene closure (i.e. `{n}` and `{n,m}`) allow up to seven repeats. Unfortunately unlimited values of `n` or `m` cause `egrep` to hang the computer. In addi-

tional experiments `n` and `m` were allowed to be up to 31 but no improvement was seen and evolved solutions continued to favour `n`= 7.

```
<start>  ::=  <RE>
<RE>  ::=  <union> | <simple-RE>
<union>  ::= <RE> "|" <simple-RE>
<simple-RE>  ::=  <concatenation> | <basic-RE>
<concatenation>  ::= <simple-RE> <basic-RE>
<basic-RE>  ::= <RE-kleen> | <elementary-RE>
<RE-kleen>:::= <minmaxquantifier> | <kleen>
<kleen>::= <star> | <plus>
<star>  ::= <elementary-RE2> "*"
<plus>  ::= <elementary-RE2> "+"
<minmaxquantifier>   ::=
        <elementary-RE4> "{" <int> <optREint> "}"
<elementary-RE>  ::= <group> | <elementary-RE3>
<elementary-RE2>  ::= <any> | <elementary-RE3>
<elementary-RE3>::= <set> | <char>
<elementary-RE4>  ::= <group> | <elementary-RE2>
<group>  ::=  "(" <RE> ")"
<set>  ::=  <positive-set> | <negative-set>
<positive-set>  ::=  "[" <set-items> "]"
<negative-set>  ::=  "[^" <set-items> "]"
<set-items>  ::=  <set-item> | <set-items2>
<set-items2>  ::=  <set-item> <set-items>
<set-item>  ::=  <char>
<char>  ::=  <c00> | <c01>
<any>  ::=  "."
<c00>  ::=  T | C
<c01>  ::=  A | G

<optREint>  ::=  <2ndint> | $
<2ndint>  ::=  "," <int>
<int>  ::=  <d0>
#4 Bit Gray Code Encoder
<d0>  ::=  <d00> | <d01>
<d00>  ::=  <d000> | <d001>
<d01>  ::=  <d010> | <d011>
<d000>  ::=  1
<d001>  ::=  3 | 2
<d010>  ::=  7 | 6
<d011>  ::=  4 | 5
```

**Figure 6: Backus-Naur form grammar used to specify legal DNA sequence regular expressions, such as: `A(CT){3}`, which matches `ACTCTCT`.**

## 5.2 Evaluating Fitness of RE Motifs

In each generation, a command file is generated which contains an `egrep -c '`*RE*`'` command for each individual in the population. (*RE* is the individual's regular expression.) The command is run on a file holding the 100 sequences lying between two blocks. (Collars of 50 additional bases mean the last 50 bases of the first well correlated block and the first 50 bases of the trailing well correlated block are also included. See Figure 5.)

`egrep -c` counts the number of probes which match the evolved motif (*RE*). The same command is also run on a file holding the 100 sequences of exactly the same length taken from exons which do not contain well separated blocks. The fitness score of the regular expression is the difference between the number of lines in the two files which match *RE*. Since, for a given generation, there are equal numbers of positive and negative training examples, the difference turns out to be proportional to the area under the ROC curve (see Table 1). Expressions which either match all probes or fail to match any are penalised by subtracting 100 from their score.

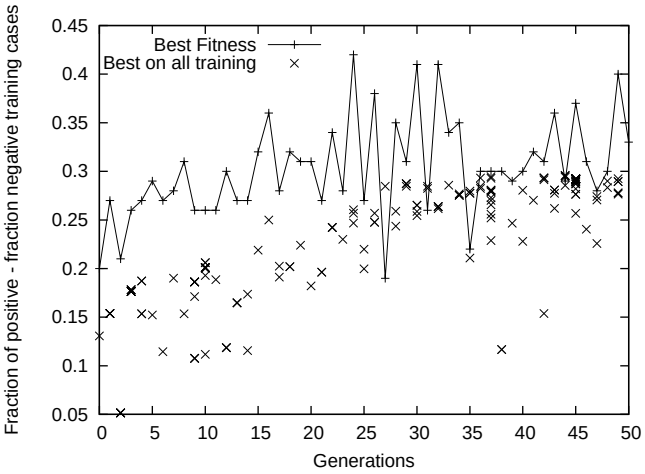



**Figure 7: Evolution of fitness for GP run predicting close sub-blocks in Human exons. Y-axis is the difference between fraction of positive and negative training cases matched by evolved regular expressions. While best performance on training data for the current generation (line) is subject to fluctuation, performance on the whole of the training data (×) improves from 0.15 to 0.3 by the end of the run.**

# 6. RESULTS

The best of generation 36 in the first run, with a population of 10 000 (cf. Table 1 and Figure 7), scored ≈0.3. This is almost as good as the best of later programs. Because of this and because its phenotype `TC(T){3}|TCGT|GG+(TCAA)|TTT(GCCA)` is shorter, it was chosen to represent the whole GP run. The evolved regular expression obtains approximately the same performance on all the training data as it did on the two hundred it was actually tested on in generation 36. However its performance falls on the 138 sequences used for validation, see Table 2. It is clear that most of the power of the evolved expression comes from its first term (`TCTTT`) and that it does well on the training data (first two columns of each group of four in Table 2). Table 3 suggests matches do not favour particular locations in the mRNA.

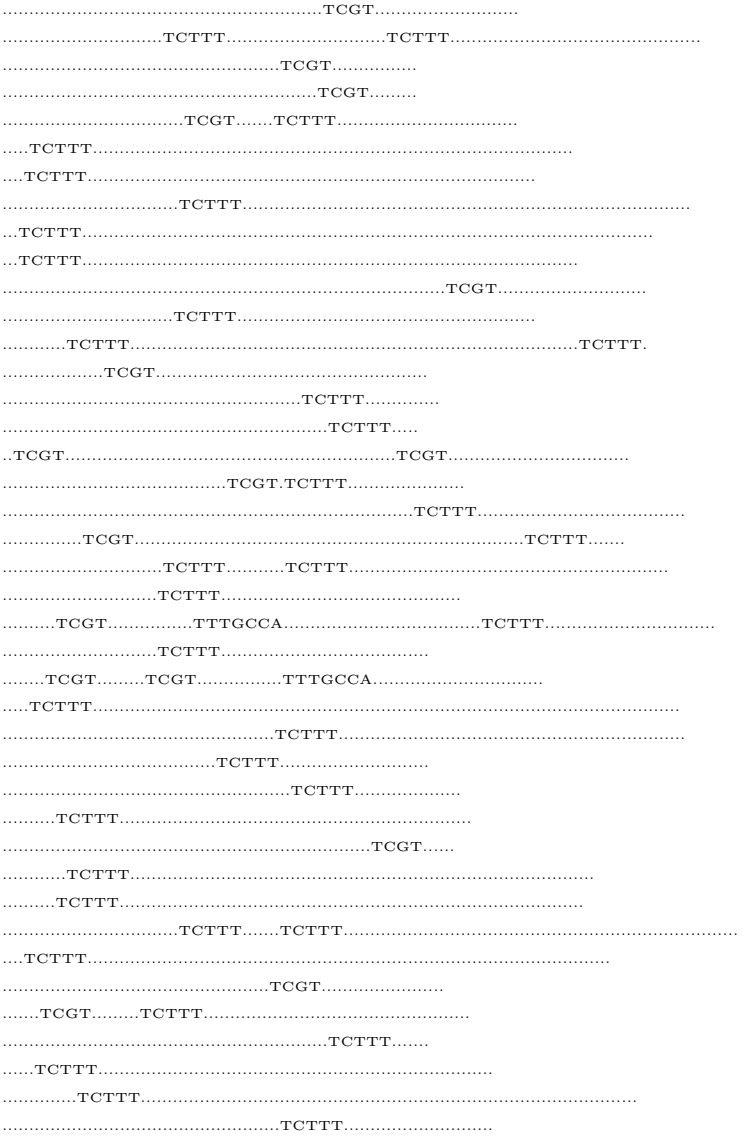
The middle of the last row of Table 2 shows the polyA regular expression does not differentiate between mRNA sequences between blocks and those in exons without sub-blocks. This suggests that alternative polyadenylation is not responsible for the observed blocks of correlations.

**Table 2: Number of DNA sequences matched by regular expression `TCTTT|TCGT|GG+TCAA|TTTGCCA` evolved in generation 36. The number of blocks identified by GP ($5^{th}$ line) falls from 78% to 52% on the holdout data but it still does significantly better than chance (second table). In the last line (PolyA) we recast Retelska *et al.*'s twin PAS DSE probabilistic motif's [28] as the deterministic regular expression `AATAAA.*T`$\not$`A(T|C)`$\not$`AT(G|T)`$\not$`A`. ($\not$`A` means any base except `A`.)**

| | Training | | Holdout | | Percent Training | | Holdout | |
|---|---|---|---|---|---|---|---|---|
| | pos | neg | pos | neg | pos | neg | pos | neg |
| `TCTTT` | 115 | 1982 | 32 | 18 | 59% | 39% | 41% | 23% |
| `TCGT` | 35 | 627 | 14 | 7 | 18% | 12% | 18% | 9% |
| `GG+TCAA` | 7 | 99 | 0 | 1 | 4% | 2% | 0% | 1% |
| `TTTGCCA` | 13 | 107 | 3 | 1 | 7% | 2% | 4% | 1% |
| GP | 152 | 2480 | 41 | 26 | 78% | 49% | 52% | 33% |
| PolyA | 26 | 406 | 5 | 7 | 13% | 8% | 6% | 9% |

| | Expected (i.e. random) Training | | Holdout | |
|---|---|---|---|---|
| | pos | neg | pos | neg |
| `TCTTT` | 31 | 794 | 13 | 13 |
| `TCGT` | 97 | 2525 | 38 | 38 |
| `GG+TCAA` | 8 | 209 | 3 | 3 |
| `TTTGCCA` | 2 | 53 | 1 | 1 |
| GP | 116 | 3025 | 46 | 46 |
| PolyA | 3 | 71 | 1 | 1 |

**Table 3: Location and type of matches of evolved regular expression on 79 positive holdout cases. (To save space one dot represents two nucleotides.) 37% of matches occur in the first 50 base collar and 22% in the trailing 50 base collar. 41% lie wholly within the gap between poorly correlated probes.**



# 7. DISCUSSION

It is known that the quality of GeneChip data tends to degrade further from the 3' end because one of the critical enzymes used in the preparation of complementary RNA strands works from the 3' end forward but may fall off leading to unexpectedly shorten molecules. However by insisting that both blocks either side of the poor correlation region be strong[2] we hoped to mitigate against this. One potential cause of the enzyme malfunction could be the RNA binding to itself and forming hairpin secondary structures. The evolved regular expression does not support this, nor did initial experiments using a grammar designed to support the evolution of motifs containing complementary patterns found in hairpins.

Support for the assumption that the blocky big Ensembl exons are in fact two exons, comes from the fact that there should then be an intron between the two exons. One signature of introns is a "polypyrimidine tract" which is a sequence rich in Cs and Ts. In earlier work we trained the GP using all the exons, i.e. including very long gaps between blocks. The evolved motifs contained runs of Cs and runs of Ts. This is reasonable if the gap is indeed due to a polypyrimidine tract associated with an undetected intron.

# 8. CONCLUSIONS

Hundreds of thousands of correlation coefficients for thousands of exons across thousands of people supports many of the current definitions of exons. However in about a thousand cases the calculation of large scale correlations of GeneChip data suggests something unexpected: sub-structure within what were previously thought of as "atomic" exons.

We have used a grammar based strongly typed GP to automatically design a motif of the type biologists are familiar with. If biologists are to try an interpret our results it is important to present them in a "biologist friendly" form, rather than as a decision tree or a set of support vectors. The evolved motif uses only DNA sequence data and yet it has some ability to predict sub block structures within existing exons.

While both grammar based and strongly typed GP have been available for some time, the combination has been little used in biology and certainly not for similar applications[3].

As described in Section 5.1, the BNF grammar is the standard one used for generic regular expression and has only be modified slightly to deal with the limited alphabet of four DNA bases. While a detailed mathematical analysis of the combinatorics of the actual Backus-Naur grammar we have used is beyond the present discussion, the number of possible grammar productions can be upper bounded by the number of ways of labeling each of the possible grammar expansions. Fortunately the maximum expansion is limited by the limit on the depth of active choices (17). This limits the maximum number of expansions to $2^{17}$ and therefore the search space to $2^{2^{17}}$ or about $4\,10^{39\,456}$.

None of the four components of the evolved regular expression `TCTTT|TCGT|GG+TCAA|TTTGCCA` (and hence none of the strings it matches) themselves match either the initial indicator of polyadenylation (known as PAS) nor the "U rich" or "GU rich" signal after the cleavage site (known as DSE) [28]. If either PAS or DSE were responsible for the block structure we are seeing in Ensembl exons, we would expect both GP and our hand created regular expression should detect them. Table 2 shows our sequences contain more polyA sequences than would be expected by chance but the polyA regular expression does not differentiate between the sequences between correlated blocks and the negative examples. I.e. the PAS/DES motifs do not explain the observed correlations.

The fact that sometimes strong blocks of mRNA measurements for the same Ensembl exon are both not well correlated and are separated by thousands of bases along the transcript suggests that they are indeed separate exons and so Ensembl should not have grouped them together as one.

Alternative splicing and alternative polyadenylation are relatively new discoveries. The regulation of both is not well understood. It is reasonable to suggest that current bioinformatic databases may not be complete and that discoveries remain to be made. As increasingly large quantities of data from multiple disparate sources are available algorithmic tools like correlation will be more widely used. Evolutionary computation is increasingly being used in bioinformatics to aid our understanding of new aspects of biology.

Training data is available via `http://bioinformatics.essex.ac.uk/users/wlangdon/tr-09-02.tar.gz`

[2]By a "strong" block we mean a length of the gene's transcript where all the applicable measurements are highly correlated with each other. This can occur even if the measurements are low on average.

[3]Untyped non-grammar linear GP, applied to other bioinformatics sequences (which was described in Section 2) is perhaps the closest to our application.

# 9. REFERENCES


[1] T. Barrett *et al.* NCBI GEO: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Research 35*, Database issue (January 2007), D760–D765.

[2] Bosman, P. A. N., and de Jong, E. D. Grammar transformations in an EDA for genetic programming. In *GECCO 2004 Workshop Proceedings* (Seattle, 26-30 June 2004), R. Poli *et al.*, Eds.

[3] Brameier, M., Krings, A., and MacCallum, R. M. NucPred predicting nuclear localization of proteins. *Bioinformatics 23*, 9 (2007), 1159–1160.

[4] Brameier, M., and Wiuf, C. Ab initio identification of human microRNAs based on structure motifs. *BMC Bioinformatics 8* (18 Dec. 2007), 478.

[5] Cetinkaya, A. Regular expression generation through grammatical evolution. In *GECCO2007 workshop program* (London, 7-11 July 2007), T. Yu, Ed., ACM Press, pp. 2643–2646.

[6] Handstad, T., Hestnes, A. J. H., and Saetrom, P. Motif kernel generated by genetic programming improves remote homology and fold detection. *BMC Bioinformatics 8*, 23 (Jan. 25 2007).

[7] Harrison, A. P., Rowsell, J., da Silva Camargo, R., Langdon, W. B., Stalteri, M., Upton, G. J., and Arteaga-Salas, J. M. The use of Affymetrix GeneChips as a tool for studying alternative forms of RNA. *Biochemical Society Transactions 36* (2008), 511–513.

[8] Hoai, Nguyen Xuan, McKay, R. I. B., and Essam, D. Representation and structural difficulty in genetic programming. *IEEE Transactions on Evolutionary Computation 10*, 2 (Apr. 2006), 157–166.

[9] Hu, Y.-J. GPRM: a genetic programming approach to finding common RNA secondary structure elements. *Nucleic Acids Research 31*, 13 (1 July 2003), 3446–3449.

[10] Hu, Y.-J., Sandmeyer, S., McLaughlin, C., and Kibler, D. Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics 16*, 3 (2000), 222–232.

[11] Langdon, W. B. *Genetic Programming and Data Structures*. Kluwer, 1998.

[12] Langdon, W. B. Evolving GeneChip correlation predictors on parallel graphics hardware. In *2008 IEEE World Congress on Computational Intelligence* (Hong Kong, 1-6 June 2008), J. Wang, Ed., IEEE Computational Intelligence Society, IEEE Press, pp. 4152–4157.

[13] Langdon, W. B., and Banzhaf, W. Repeated sequences in linear genetic programming genomes. *Complex Systems 15*, 4 (2005), 285–306.

[14] Langdon, W. B., and Barrett, S. J. Genetic programming in data mining for drug discovery. In *Evolutionary Computing in Data Mining*, A. Ghosh and L. C. Jain, Eds., vol. 163 of *Studies in Fuzziness and Soft Computing*. Springer, 2004, ch. 10, pp. 211–235.

[15] Langdon, W. B., and Buxton, B. F. Evolving receiver operating characteristics for data fusion. In *Genetic Programming, Proceedings of EuroGP'2001* (Lake Como, Italy, 18-20 Apr. 2001), J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, and W. B. Langdon, Eds., vol. 2038 of *LNCS*, Springer-Verlag, pp. 87–96.

[16] Langdon, W. B., and Harrison, A. P. Evolving DNA motifs to predict GeneChip probe performance. *Algorithms in Molecular Biology*. In press.

[17] Langdon, W. B., and Harrison, A. P. Evolving regular expressions for GeneChip probe performance prediction. Tech. Rep. CES-483, Computing and Electronic Systems, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK, 27 Apr. 2008.

[18] Langdon, W. B., McKay, R. I., and Spector, L. Genetic programming. In *Handbook of Metaheuristics*. Springer.

[19] Langdon, W. B., Rowsell, J., and Harrison, A. P. Creating regular expressions as mRNA motifs with GP to predict human exon splitting. In *GECCO '09: Proceedings of the 11th annual conference on Genetic and evolutionary computation* (Montreal, 8-12 July 2009), F. Rothlauf *et al.*, Eds., ACM. Forthcoming.

[20] Langdon, W. B., Upton, G. J. G., da Silva Camargo, R., and Harrison, A. P. A survey of spatial defects in Homo Sapiens Affymetrix GeneChips. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2009). In press.

[21] McKay, R. I., Hoang, Tuan Hao, Essam, D. L., and Nguyen, Xuan Hoai Developmental evaluation in genetic programming: the preliminary results. In *Proceedings of the 9th European Conference on Genetic Programming* (Budapest, Hungary, 10 - 12 Apr. 2006), P. Collet *et al.* Eds., vol. 3905 of *Lecture Notes in Computer Science*, Springer, pp. 280–289.

[22] Michal, S., Ivry, T., Schalit-Cohen, O., Sipper, M., and Barash, D. Finding a common motif of RNA sequences using genetic programming: The geRNAMo system. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 4*, 4 (Oct.-Dec. 2007), 596–610.

[23] Nikolaev, N. I., and Slavov, V. Concepts of inductive genetic programming. In *Proceedings of the First European Workshop on Genetic Programming* (Paris, 14-15 Apr. 1998), W. Banzhaf *et al*, Eds., vol. 1391 of *LNCS*, Springer-Verlag, pp. 49–60.

[24] O'Neill, M., and Ryan, C. Grammatical evolution. *IEEE Transactions on Evolutionary Computation 5*, 4 (Aug. 2001), 349–358.

[25] Paterson, N. R., and Livesey, M. Distinguishing genotype and phenotype in genetic programming. In *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996* (28–31 July 1996), J. R. Koza, Ed., Stanford Bookstore, pp. 141–150.

[26] Poli, R., Langdon, W. B., and McPhee, N. F. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[27] Ratle, A., and Sebag, M. Avoiding the bloat with probabilistic grammar-guided genetic programming. In *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001* (Creusot, France, Oct. 29-31 2001), P. Collet, *et al.*, Eds., vol. 2310 of *LNCS*, Springer Verlag, pp. 255–266.

[28] Retelska, D., *et al.* Similarities and differences of polyadenylation signals in human and fly. *BMC Genomics 7*, 1 (2006), 176.

[29] Ross, B. J. The evaluation of a stochastic regular motif language for protein sequences. In *GECCO-2001* (San Francisco, 7-11 July 2001), L. Spector *et al.*, Eds., Morgan Kaufmann, pp. 120–128.

[30] Ryan, C., Collins, J. J., and O'Neill, M. Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the First European Workshop on Genetic Programming* (Paris, 14-15 Apr. 1998), W. Banzhaf, *et al.*, Eds., vol. 1391 of *LNCS*, Springer-Verlag, pp. 83–95.

[31] Salustowicz, R. P., and Schmidhuber, J. Probabilistic incremental program evolution: Stochastic search through program space. In *Machine Learning: ECML-97* (1997), M. van Someren and G. Widmer, Eds., vol. 1224 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 213–220.

[32] Sanchez-Graillet, O., Rowsell, J., Langdon, W. B., Stalteri, M. A., Arteaga Salas, J. M., Upton, G. J., and Harrison, A. P. Widespread existence of uncorrelated probe intensities from within the same probeset on Affymetrix GeneChips. *Journal of Integrative Bioinformatics 5*, 2 (2008), 98.

[33] Shan, Yin *et al.* Grammar model-based program evolution. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation* (Portland, Oregon, 20-23 June 2004), IEEE Press, pp. 478–485.

[34] Tomita, Y. *et al.* A motif detection and classification method for peptide sequences using genetic programming. *Journal of Bioscience and Bioengineering 106*, 2 (2008), 154–161.

[35] Whigham, P. A. Search bias, language bias, and genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference* (Stanford University, 28–31 July 1996), J. R. Koza *et al.* Eds., MIT Press, pp. 230–237.

[36] Whigham, P. A., and Crapper, P. F. Time series modelling using genetic programming: An application to rainfall-runoff models. In *Advances in Genetic Programming 3*, L. Spector *et al.*, Eds. MIT Press, June 1999, ch. 5, pp. 89–104.

[37] Man Leung Wong and Kwong Sak Leung Evolving recursive functions for the even-parity problem using genetic programming. In *Advances in Genetic Programming 2*, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, 1996, ch. 11, pp. 221–240.