Automated DNA Motif Discovery

W. B. Langdon, Olivia Sanchez Graillet, A. P. Harrison

Department of Computer Science, King's College London

Departments of Mathematical Sciences and Biological Sciences
University of Essex, UK

Abstract

Ensembl's human non-coding and protein coding genes are used to automatically find DNA pattern motifs. The Backus-Naur form (BNF) grammar for regular expressions is used by genetic programming to ensure the generated strings are legal. The evolved motif suggests the presence of Thymine followed by one or more Adenines etc. early in transcripts indicate a non-protein coding gene.

Keywords: pseudogene, short and microRNAs, non-coding transcripts, systems biology, machine learning, strongly typed genetic programming

1 Introduction

We present a new method for finding DNA motifs. First we will describe the existing work which uses grammars to constrain the artificial evolution of programs and its application to finding patterns, particularly finding protein motifs. The Methods section (2) describes how Ensembl [Hubbard *et al.*, 2009] DNA sequences are prepared and used. The new grammar based genetic programming (2.2) is demonstrated (Section 3) by its ability to automatically find patterns early in human genes which distinguish non-protein coding genes genes from protein coding genes.

1.1 Evolving Grammars and Protein Motifs

Existing research on using grammars to constrain the artificial evolution of programs can be broadly divided in two: Grammatical Evolution [O'Neill and Ryan, 2001] which uses BNF grammars and is based largely in Ireland and work in the far east using context-free grammars, tree adjoining grammars and inductive logic by Whigham, McKay and Wong. See, for example, [Whigham, 1996; Whigham and Crapper, 1999], [Hoang *et al.*, 2008] and [Wong and Leung, 1996]. Grammars are also used in many Bioinformatics applications, particularly dealing with sequences.

Ross induced stochastic regular expressions from a number of grammars to classify proteins from their amino acid sequence [Ross, 2001]. Regular expressions have been evolved to search for similarities between proteins, again based on their amino acid sequences [Handstad *et al.*, 2007 Whilst Brameier used amino acids sequences to predict the location of proteins by applying a multi-classifier [Langdon and Buxton, 2001] linear genetic programming (GP) based approach [Brameier *et al.*, 2007] (although this can be done without a grammar [Langdon and Banzhaf, 20 A similar technique has also been applied to study microRNAs [Brameier and Wiuf, 2007]. An interesting departure is Pappa's work which uses a grammar based GP to create application domain specific algorithms. E.g. [Pappa and Freitas, 2009], which considers prediction of

protein function. While Dyrka and Nebel have used a genetic algorithm and a more powerful but also more complicated context free grammar. For example, they used a CFG when finding a meta-pattern describing protein sequences associated with zinc finger RNA binding sites [Dyrka and Nebel, 2009]. Zinc finger was amongst the protein superfamilies sequence prediction tasks used by [Dobson *et al.*, 2009]. Although Support Vector Machines can achieve high accuracy (they obtained 66.3%) SVM models can be difficult for non-specialists to understand.

Non-stochastic machine learning techniques have also been applied to DNA motifs. E.g. [Hu *et al.*, 2000], present a method based on decision trees, specifically C4.5. Note we are deliberately seeking intelligible motifs and so rule out approaches, such as [Won *et al.*, 2007], which evolved high performance but non-intuitive models for protein secondary structure prediction. [George and Tenenbaum, 2009] concisely list current computational techniques used with RNA motifs.

We must be wary of over claiming. As [Baird *et al.*, 2006] point out computational prediction is hard. Indeed they say for one problem (identification of new internal ribosome entry sites (IRES) in viral RNA) it is still not possible. Nevertheless, by concentrating on a generic tool which generates human readable motifs, of a type which are well known to Biologists, computers may still be of assistance.

2 Methods

2.1 Preparation of Training Data

The DNA sequences for all human genes were taken from Ensembl (version 48). There are 46 319 protein coding and 9 836 non-coding transcripts. (Many genes have more than one transcript. There are 22 740 coding and 9 821 non-coding human genes.) As Table 1 shows most non-protein coding human genes are either pseudogenes of some sort or lead to short or micro-RNAs.

We need to be able to check later that the automatically generated motif is general. I.e. it has not over fitted the examples it has seen and does not fail on new unseen examples. Therefore the protein coding and non-coding genes were randomly split in half. (Transcripts for the same gene were kept together). One half is available for training the GP and the second is never seen by GP and is reserved for demonstrating the performance of the evolved motif. The training data were then processed for use by the GP.

2.1.1 Training Data Sets for Generating DNA Motifs

Where a gene has multiple transcripts one was randomly chosen to be included in the training data. The other transcripts for the same Ensembl gene were not used for training.

Figure 1 makes it clear that transcripts from non-coding genes tend to be shorter than those produced by protein coding genes. If the length of the transcript is known, this would be a very easy way to distinguish protein coding genes. However a classifier which simply said "if the transcript exceeds 500 bases, the gene encodes a protein" would tell us nothing new (even though it might be quite good at predicting). So we insist the GP seek out predictive DNA sequences. Therefore the GP is not told how long the transcript is. Instead all training

Table 1: Number and type of each non-protein coding Ensembl human gene

pseudogene	1516
snRNA	1337
misc_RNA	1041
miRNA	968
scRNA_pseudogene	843
snoRNA	716
Mt_tRNA_pseudogene	603
retrotransposed	565
snRNA_pseudogene	501
snoRNA_pseudogene	486
rRNA_pseudogene	341
rRNA	334
V_segment	236
tRNA_pseudogene	129
J_segment	99
C_segment	36
D_segment	32
Mt_tRNA	22
miRNA_pseudogene	21
misc_RNA_pseudogene	7
Mt_rRNA	2
scRNA	1
total	9836

data have exactly 60 bases taken from the start of the Ensembl transcript. (Transcripts less than 60 bases were not used for training). Finally duplicate sequences were removed. This gave 4639 unique non-protein coding and 11 191 unique protein coding sequences for use as training examples.

2.1.2 Genetic Programming Training Set

To avoid unbalanced training sets, every generation all 4639 non-protein coding examples were used and 4639 coding examples were randomly chosen from the 11 191 protein coding examples available. This is done by placing the coding examples at random in a ring. Each generation the next 4639 examples are taken from the ring. This ensures the coding examples are regularly re-used. (Each protein coding example is used once per 2.41 generations.)

2.2 Evolving DNA Motifs

Having created training data we then use a strongly typed tree GP system [Poli *et al.*, 2008] to create an initial random population of motifs. Each generation the best 20% are chosen and a new generation of motifs is created from them using two types of mutation (shrink and

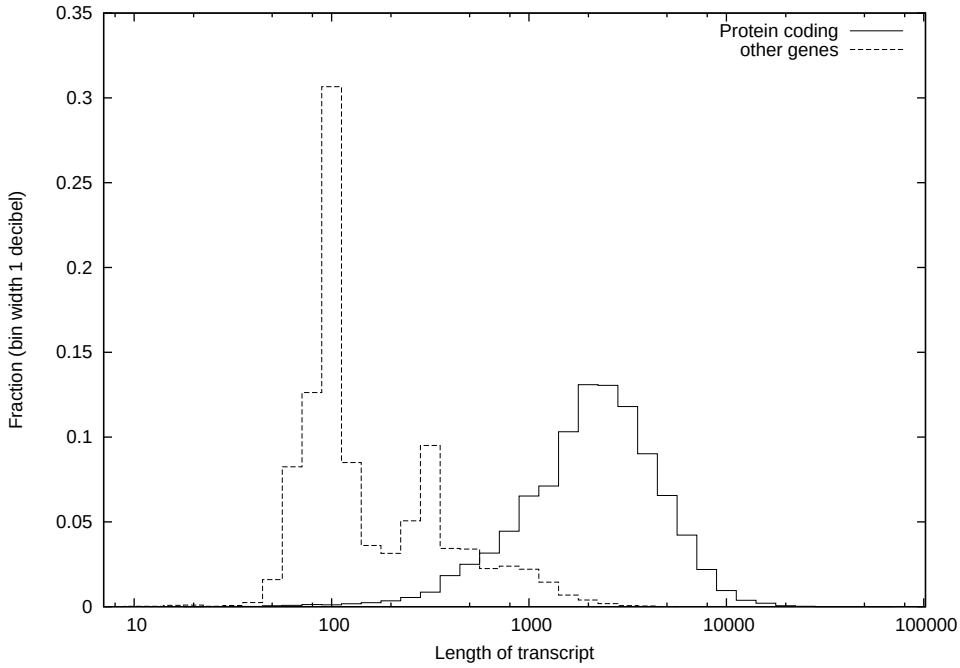

Figure 1: Distributions of number of bases per Ensembl human transcript. Note the length of protein coding transcripts is approximately log normally distributed. Most non-coding genes are shorter than protein coding genes.

subtree) and subtree crossover [Poli *et al.*, 2008; Langdon, 1998]. (The exact parameters are given in Table 2.) Over a number of generations the performance of the best motifs in the population improves. After 50 generations we stop the GP and take the best motif at that point and see how well it does. It is not only tested on the DNA sequences used to train it but, in order to estimate how well it does in general, it is tested also on the DNA sequences kept back (cf. Section 2.1).

2.2.1 Backus-Naur Form Grammar of Motifs

The BNF grammar is given in [Langdon and Harrison, 2009, Figure 8, page 10]. Whilst it could be tuned to each application, this has not been necessary. In fact, we have used the same grammar for a very different task (isolating poorly performing Affymetrix cDNA probes [Langdon and Harrison, 2009]). Technical details and the reasons for its design are given in [Langdon and Harrison, 2009] and [Langdon and Harrison, 2008].

The initial population of motifs is created by passing at random through the BNF grammar using the standard GP algorithm (ramped half-and-half [Poli *et al.*, 2008]). Although this may seem complex, `gawk` (an interpreted language) is fast enough to handle populations of a million individuals.

Table 2: Strongly Typed Grammar based GP Parameters for Pseudogene and non-coding short RNA Prediction

Primitives:	The functions and inputs and how they are combined is defined by the BNF grammar [Langdon and Harrison, 2009, Figure 8, page 10]
Performance:	true positives+true negatives. (I.e. proportional to the area under the ROC curve or Wilcox statistic [Langdon and Barrett, 2004].) Less large penalty if it matches all RNA training sequences or none.
Selection:	(200,1000) generational, non-elitist, Population size = 1000
Initial pop:	Ramped half-and-half 3:7
Parameters:	90% subtree crossover, 5% subtree mutation, 5% shrink mutation. Max tree depth 17 (no tree size limit)
Termination:	50 generations

2.2.2 Creating New Trial Motifs

After each generation, the best 20% of the current population are chosen to be the parents of the next generation. Each parent is allocated (on average) five children. Thus the next generation is the same size as the previous one.

Children are created by either mutating high scoring parents or by recombination of two high scoring parents, cf. [Poli *et al.*, 2008, Figure 2.5]. In all cases the changes are made so that the resulting offspring obeys the BNF syntax rules and so are valid motifs. Therefore their performance can be estimated and (although some may perform badly) they are all still comprehensible motifs.

2.2.3 Evaluating the Motifs

Each generation each trial motif in the population is tested against the DNA sequences of the 4639 unique non-protein coding 60 base sequences available for training and 4639 protein coding 60 base unique sequences selected for use in this generation. Their performance is the sum of the number of non-coding sequences they match and the number of protein coding they do not match. However motifs which either match all or fail to match any are penalised by subtracting 4639 from their score.

3 Results

At the end of the first run, with a population of 1000 (cf. Table 2 and Figure 2) genetic programming produced the motif `TACT|TGAT..|TA+TAT.|TA+(.CA+|T)(C|T)`. (This motif can be understood by noting the vertical bar `|` indicates options. That is, if a sequence contains `TACT` or `TGAT..` or `TA+TAT.` or `TA+(.CA+|T)(C|T)` the motif is said to match it. The last two vertical bars are inside brackets `()` which must be taken into account before the vertical bar they enclose. Thus the `(C|T)` means either a Cytosine or a Thymine placed immediately after bases which match `TA+(.CA+|T)`. The dots "`.`" mean one of the four bases must occur here. Finally `A+` means a run of at least one Adenines.

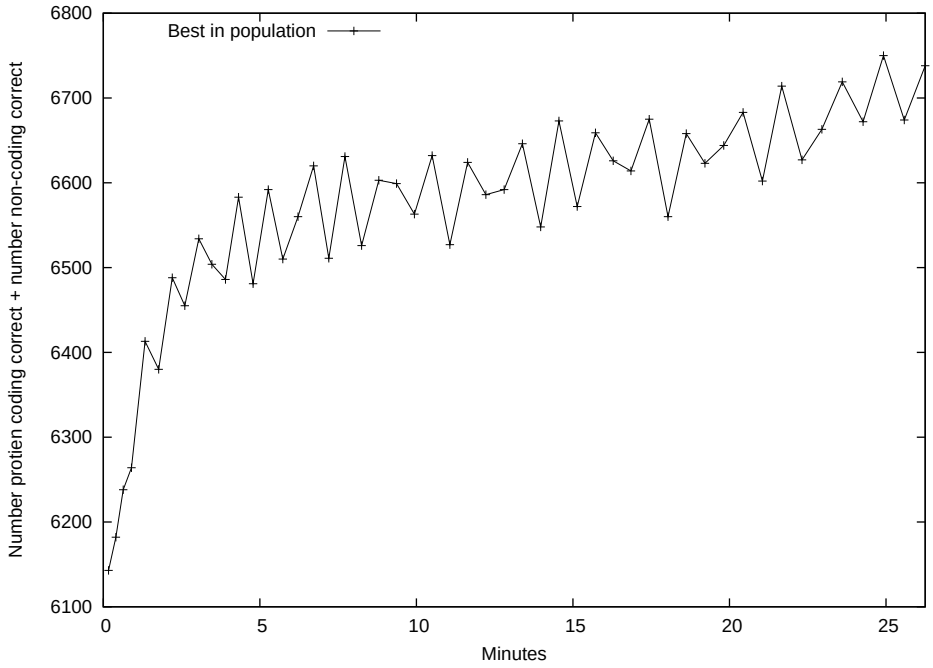

Figure 2: Evolution of breeding population of motifs trying to locate human protein coding genes. Each generation the protein coding training cases are replaced leading to fluctuations in the measured performance. However the trend is steadily upwards.

Confusion matrices are a compact way to show the performance of prediction algorithms. They are particularly useful where there are many more examples of one class (e.g. protein coding) than another. An inept classifier which always said "protein coding" would often be correct and so have a high percentage accuracy. However it would be useless. By showing how well it does on all types of transcript a confusion matrix reveals its real performance. The matrix says how well the classifier does on each actual class (the columns). Where there are many classes, confusion matrices can also be helpful by showing where the classifier's predictions (the rows) are wrong. An good classifier will have a matrix with high values only on its leading diagonal.

The following pair of confusion matrices give the evolved motif performance on its own training data (i.e. the training data used in the last generation) and on all the data used by GP. Of course the actual non-coding examples are the same in the two cases. However the motif performs equally well on all the protein coding training examples as it does on the protein coding examples randomly selected for us in the last generation. (I.e. they are not significantly different, χ^2, 1 dof.) This suggests the strategy of randomly changing training examples every generation has worked well.

	Last GP generation				GP training data			
	non protein		protein coding		non protein		protein coding	
non protein	3483	(75%)	1403	(30%)	3483	(75%)	3390	(30%)
protein coding	1156	(25%)	3236	(70%)	1156	(25%)	7801	(70%)

The next pair of confusion matrices are included for completeness. The left hand side gives the evolved motif's performance on the first 60 bases of the whole of the training data (i.e. including duplicates). The right hand confusion matrix refers to when the evolved pattern is applied to the whole transcript, rather than just its first 60 bases.

All training data (60)

	non protein		protein coding	
non protein	3572	(75%)	3447	(30%)
protein coding	1196	(25%)	7899	(70%)

All training data (whole transcript)

	non protein		protein coding	
non protein	4535	(92%)	11227	(99%)
protein coding	375	(8%)	143	(1%)

The next pair of confusion matrices contain the evolved motif's performance on all the holdout data (selecting only one transcript per gene).

Holdout data (60)

	non protein		protein coding	
non protein	3609	(76%)	3503	(31%)
protein coding	1159	(24%)	7844	(69%)

Holdout data (whole transcript)

	non protein		protein coding	
non protein	4529	(92%)	11207	(99%)
protein coding	382	(8%)	163	(1%)

The last pair of matrices include all transcripts for each of the hold out genes. The motif holds its performance when applied to the first 60 bases of each Ensembl transcript. However the shortness of the motif and the fact it can match the transcript at any point means the start of the transcript must be selected before using the motif otherwise performance falls. (Cf. the right hand of the previous two pairs of confusion matrices and the right hand of next pair.)

Holdout data (all transcripts, ≤ 60)

	non protein		protein coding	
non protein	3683	(75%)	6883	(30%)
protein coding	1234	(25%)	16101	(70%)

Holdout (all transcripts, whole transcript)

	non protein		protein coding	
non protein	4541	(92%)	22778	(99%)
protein coding	376	(8%)	206	(1%)

Unlike many machine learning applications, there is no evidence of over fitting. Indeed the corresponding results for the holdout set are not significantly different (χ^2, 3 dof) from those on the whole training set. (Both when looking at the first 60 bases or the whole transcript).

Table 3 gives a break down of the evolved regular expression motif both by Ensembl human transcript type and by its components. (Note `TA+(.CA+|T)(C|T)`) has been re-expressed as the union of four expressions: `TA+.CA+C`, `TA+.CA+T`, `TA+TC` and `TA+TT`.) The last part of the motive (i.e. `TA+(.CA+|T)(C|T)`) typically scores more highly than the first three. However the evolved pattern succeeds at separating the non-protein coding from the protein genes by working together.

It is sufficient for just one of the seven patterns to match the beginning of the gene. In many cases either several of the seven match and/or they match the DNA more than once. However the patterns are usually distinct in that, even in a gene which is matched by more than one of the 7 patterns, a part of the DNA which matches one is unlikely to also match another.

Although the evolved motif has some similarity with the TATA box motif, it does not match the consensus sequence `TATAAA` [Yang *et al.*, 2007] exactly. `TATAAA` occurs in the first

60 bases in 1.1% (106) of the 9 836 non-protein transcripts and 0.6% (290) of the 46 319 protein transcripts. Depending on the expected prevalence of the four bases, this is about what would be expected by chance.

4 Discussion

The combination of genetic programming and a BNF grammar designed for the production of intelligible patterns can be a viable way to automatically find interesting motifs in DNA and RNA sequences. The prototype system is available via `ftp://cs.ucl.ac.uk/genetic/gp-code/RE_gp` It has been demonstrated on a large biological DNA problem: discriminating non-protein coding from protein coding genes.

The automatically generated motif `TACT|TGAT..|TA+TAT.|TA+(.CA+|T)(C|T)` suggests that Thymine followed by one or more Adenine bases (particularly if the run is terminated by another Thymine or a Cytosine and Thymine) at the start of a transcript, indicates the transcript may be a short non-coding RNA sequence rather than from a protein-coding gene.

Acknowledgement

This work was supported by the UK Biotechnology and Biological Sciences Research Council. under grant code BBSRC BBE0017421.

Table 3: Performance of motif on first 60 bases by components and Ensembl transcript type

transcript type	TACT		TGAT..		TA+TAT.		TA+.CA+C		TA+.CA+T		TA+TC		TA+TT		Combined	
pseudogene	158	(10%)	269	(17%)	41	(2%)	67	(4%)	42	(2%)	171	(11%)	196	(12%)	676	(44%)
snRNA	739	(55%)	448	(33%)	109	(8%)	120	(8%)	753	(56%)	217	(16%)	737	(55%)	1237	(92%)
misc_RNA	166	(15%)	671	(64%)	18	(1%)	55	(5%)	73	(7%)	389	(37%)	429	(41%)	992	(95%)
miRNA	197	(20%)	161	(16%)	154	(15%)	42	(4%)	64	(6%)	102	(10%)	327	(33%)	652	(67%)
scRNA_pseudogene	463	(54%)	157	(18%)	12	(1%)	36	(4%)	52	(6%)	131	(15%)	134	(15%)	671	(79%)
snoRNA	142	(19%)	395	(55%)	75	(10%)	43	(6%)	96	(13%)	144	(20%)	212	(29%)	588	(82%)
Mt_tRNA_pseudogene	68	(11%)	179	(29%)	52	(8%)	72	(11%)	125	(20%)	168	(27%)	235	(38%)	518	(85%)
retrotransposed	69	(12%)	75	(13%)	4	(0%)	23	(4%)	26	(4%)	66	(11%)	57	(10%)	237	(41%)
snRNA_pseudogene	201	(40%)	210	(41%)	34	(6%)	82	(16%)	169	(33%)	66	(13%)	208	(41%)	465	(92%)
snoRNA_pseudogene	121	(24%)	301	(61%)	23	(4%)	7	(1%)	93	(19%)	26	(5%)	94	(19%)	437	(89%)
rRNA_pseudogene	39	(11%)	176	(51%)	4	(1%)	98	(28%)	39	(11%)	55	(16%)	35	(10%)	263	(77%)
rRNA	28	(8%)	285	(85%)	2	(0%)	222	(66%)	24	(7%)	52	(15%)	7	(2%)	320	(95%)
V_segment	35	(14%)	26	(11%)	6	(2%)	7	(2%)	2	(0%)	17	(7%)	41	(17%)	89	(37%)
tRNA_pseudogene	10	(7%)	32	(24%)	3	(2%)	6	(4%)	11	(8%)	26	(20%)	15	(11%)	77	(59%)
J_segment	17	(17%)	15	(15%)	1	(1%)	10	(10%)	5	(5%)	18	(18%)	22	(22%)	60	(60%)
C_segment	0	(0%)	4	(11%)	0	(0%)	4	(11%)	0	(0%)	3	(8%)	3	(8%)	8	(22%)
D_segment	5	(15%)	2	(6%)	0	(0%)	0	(0%)	0	(0%)	1	(3%)	9	(28%)	10	(31%)
Mt_tRNA	2	(9%)	6	(27%)	2	(9%)	4	(18%)	3	(13%)	7	(31%)	5	(22%)	18	(81%)
miRNA_pseudogene	1	(4%)	3	(14%)	0	(0%)	2	(9%)	1	(4%)	1	(4%)	1	(4%)	7	(33%)
misc_RNA_pseudogene	2	(28%)	2	(28%)	0	(0%)	0	(0%)	1	(14%)	1	(14%)	2	(28%)	5	(71%)
Mt_rRNA	1	(50%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	1	(50%)	2	(100%)
scRNA	0	(0%)	1	(100%)	0	(0%)	0	(0%)	1	(100%)	0	(0%)	1	(100%)	1	(100%)
totals	2464	(25%)	3418	(34%)	540	(5%)	900	(9%)	1580	(16%)	1661	(16%)	2771	(28%)	7333	(74%)
protein_coding	3565	(7%)	4637	(10%)	767	(1%)	1325	(2%)	1190	(2%)	3351	(7%)	4077	(8%)	13751	(29%)

References

[Baird *et al.*, 2006] Stephen D. Baird, Marcel Turcotte, Robert G. Korneluk, and Martin Holcik. Searching for IRES. *RNA*, 12(10):1755–1785, October 2006.

[Brameier and Wiuf, 2007] Markus Brameier and Carsten Wiuf. Ab initio identification of human microRNAs based on structure motifs. *BMC Bioinformatics*, 8:478, 18 December 2007.

[Brameier *et al.*, 2007] Markus Brameier, Andrea Krings, and Robert M. MacCallum. NucPred predicting nuclear localization of proteins. *Bioinformatics*, 23(9):1159–1160, 2007.

[Dobson *et al.*, 2009] Richard J B Dobson, Patricia B Munroe, Mark J Caulfield, and Mansoor Saqi. Global sequence properties for superfamily prediction: a machine learning approach. *Journal of Integrative Bioinformatics*, 6(1):109, 2009.

[Dyrka and Nebel, 2009] Witold Dyrka and Jean-Christophe Nebel. A stochastic context free grammar based framework for analysis of protein sequences. *BMC Bioinformatics*, 10(323), 2009.

[George and Tenenbaum, 2009] Ajish D. George and Scott A. Tenenbaum. Informatic resources for identifying and annotating structural RNA motifs. *Molecular Biotechnology*, 41(2):180–193, February 2009.

[Handstad *et al.*, 2007] Tony Handstad, Arne J H Hestnes, and Pal Saetrom. Motif kernel generated by genetic programming improves remote homology and fold detection. *BMC Bioinformatics*, 8(23), January 25 2007.

[Hoang *et al.*, 2008] Tuan-Hao Hoang, Daryl Essam, R. I. (Bob) McKay, and Nguyen Xuan Hoai. Developmental evaluation in genetic programming: The TAG-based frame work. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 12(1):69–82, 2008.

[Hu *et al.*, 2000] Yuh-Jyh Hu, Suzanne Sandmeyer, Calvin McLaughlin, and Dennis Kibler. Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics*, 16(3):222–232, 2000.

[Hubbard *et al.*, 2009] T. J. P. Hubbard, B. L. Aken, S. Ayling, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, L. Clarke, G. Coates, S. Fairley, S. Fitzgerald, J. Fernandez-Banet, L. Gordon, S. Graf, S. Haider, M. Hammond, R. Holland, K. Howe, A. Jenkinson, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, D. Lawson, I. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, D. Rios, M. Schuster, G. Slater, D. Smedley, W. Spooner, G. Spudich, S. Trevanion, A. Vilella, J. Vogel, S. White, S. Wilder, A. Zadissa, E. Birney, F. Cunningham, V. Curwen, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, A. Kasprzyk, G. Proctor, J. Smith, S. Searle1, and P. Flicek. Ensembl 2009. *Nucleic Acids Research*, 37(Database issue):D690–D697, 2009.

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

[Langdon and Barrett, 2004] W. B. Langdon and S. J. Barrett. Genetic programming in data mining for drug discovery. In Ashish Ghosh and Lakhmi C. Jain, editors, *Evolutionary Computing in Data Mining*, volume 163 of *Studies in Fuzziness and Soft Computing*, chapter 10, pages 211–235. Springer, 2004.

[Langdon and Buxton, 2001] William B. Langdon and Bernard F. Buxton. Evolving receiver operating characteristics for data fusion. In Julian F. Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G. B. Tettamanzi, and William B. Langdon, editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 87–96, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.

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

[Langdon and Harrison, 2009] W. B. Langdon and A. P. Harrison. Evolving DNA motifs to predict GeneChip probe performance. *Algorithms in Molecular Biology*, 4(6), 19 March 2009.

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

[O'Neill and Ryan, 2001] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, August 2001.

[Pappa and Freitas, 2009] Gisele L. Pappa and Alex A. Freitas. Automatically evolving rule induction algorithms tailored to the prediction of postsynaptic activity in proteins. *Intelligent Data Analysis*, 13(2):243–259, 2009.

[Poli *et al.*, 2008] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *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).

[Ross, 2001] Brian J. Ross. The evaluation of a stochastic regular motif language for protein sequences. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 120–128, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[Whigham and Crapper, 1999] Peter A. Whigham and Peter F. Crapper. Time series modelling using genetic programming: An application to rainfall-runoff models. In Lee Spector, William B. Langdon, Una-May O'Reilly, and Peter J. Angeline, editors, *Advances in Genetic Programming 3*, chapter 5, pages 89–104. MIT Press, 1999.

[Whigham, 1996] P. A. Whigham. Search bias, language bias, and genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 230–237, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[Won *et al.*, 2007] Kyoung-Jae Won, Thomas Hamelryck, Adam Prugel-Bennett, and Anders Krogh. An evolutionary method for learning HMM structure: prediction of protein secondary structure. *BMC Bioinformatics*, 8(357), 2007.

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

[Yang *et al.*, 2007] Chuhu Yang, Eugene Bolotin, Tao Jiang, Frances M. Sladek, and Ernest Martinez. Prevalence of the initiator over the TATA box in human and yeast genes and identification of DNA motifs enriched in human TATA-less core promoters. *Gene*, 389(1):52–65, 2007.