Evolutionary Computation Method for Promoter Site Prediction in DNA

Daniel Howard and Karl Benson

Software Evolution Centre
Knowledge and Information Systems
QinetiQ, St Andrews Road, Malvern
WORCS WR14 3PS, United Kingdom
{dhoward,kabenson}@qinetiq.com
http://www.qinetiq.com

Abstract. This paper develops an evolutionary method that learns inductively to recognize the makeup and the position of very short consensus sequences, which are a typical feature of promoters in eukaryotic genomes. This class of method can be used to discover candidate promoter sequences in primary sequence data. If further developed, it has the potential to discover genes which are regulated together.

1 Introduction

Sequencing the complete genome for a variety of organisms is taking place at an unprecedented pace. Meanwhile, computer programs are being developed to scan genomes and to identify likely genes, the *trans*-acting sites. Yet, genomic sequencing data also presents an opportunity for pattern recognition of *cis*-acting sites in the genome whose purpose is to attract proteins, the enzymes, which bind DNA to initiate and regulate gene transcription.

By analogy, the *trans*-acting part of the genome represents the memory of a computer and the *cis*-acting part of the genome represents a computer program that acts on this memory. Understanding *cis*-acting sites is essential to providing models of what is the makeup of this computer program and how it operates. Physical and functional differences between two living forms such as a human and a chimpanzee have much more to do with differences in the execution of this computer program than with differences in the building blocks (both human and chimp use essentially the same proteins). Another example is *Drosophila*, where the proteins used for antenna development are essentially similar to those used for the development of legs. The difference between the appearance and function of these parts is controlled by the sequence of gene expression (mutation of the antennapedia gene transforms the antenna into a leg). It is also interesting to note that this computer program executes in a sophisticated and parallel fashion, as exemplified by polymerase pausing (transcription of a gene halts part way through, and remains in this halted state until a protein, a message, arrives at the scene as a trigger to resume transcription).

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1690–1701, 2003.

An important *cis*-acting region is the *promoter* region located typically upstream of the gene's transcription start site (TSS). This *cis*-acting region attracts proteins essential to the binding of RNA polymerase to DNA strands. RNA polymerase is the enzyme that catalyzes production of RNA from the DNA template.

Promoters are of interest to drug discovery research. It has been argued that Huntington's chorea, a monogenetic disease, occurs when a mutant gene product blocks a crucial promoter region elsewhere in the body, in turn reducing expression of a necessary protein, which results in damage to the brain over time [Cattaneo et al., 2002]. Over-expression of an enzyme by a variation in the promoter plays a crucial role in pharmacokinetics (or how a patient's body handles a particular drug). Genetic variants of the enzymes responsible for absorption, inactivation, and excretion may affect the amount of drug that reaches the biological target significantly, with over-expression causing the drug to be absorbed too rapidly and the body to eliminate it before it can take effect [RGEP, 2002].

Computer programs that identify the *cis*-acting control regions have potential for enabling a researcher to use a computer to scan large sequence databases for specific classes of genes which possess a desired program of regulation.

1.1 Eukaryotic Promoter Prediction

In prokaryotes, transcription is regulated by one RNA polymerase while in eukaryotes there exists three forms. RNA polymerase II (pol-II), is of particular interest because it transcribes heterogeneous RNA (the precursors of mRNAs, and also small nuclear RNAs), and also because the regulation of class II genes is the most complex of the three.

Promoters used by pol-II are recognized principally by separate accessory factors rather than principally by the polymerase enzyme as in prokaryotes. A basal transcription machinery is responsible for the recruitment of pol-II.

An "accessory factor" or "transcription factor" is defined to be any protein needed for the initiation of transcription regardless of the mechanics. Important basal transcription factors are: TFIID, TFIIA, TFIIB, TFIIF, TFIIE, TFIIH, TFIIJ and TFIIS. These factors are all general and responsible only for low basal levels of transcription. There also exist many gene-specific transcriptional factors (activators) which recognize elements in the promoter or other regions of the gene, and binding to them are able to recruit the general factors.

As a rule, promoters have elements or nucleotide sequence motifs that are important to their utility, these are present in all promoters and conserved with some variation. In a prokaryote or bacteria, consensus sequences are well known and established owing to the simple form of the initiation of transcription machinery (the overwhelming role of the σ transcription factor). Eukaryotes also have sequence motifs important to the basal apparatus (see below, Table 6). However, a consensus is not clear, and promoters can function without many of these elements (without important ones such as `TATA`).

In eukaryotes there must be tissue specific gene transcription because many different cell types need to be created. Regulation must be selective in different

cells and tissues, and is unique for more than 200 different cell types, (see chapter 6, [Tsonis, 2003]). This gives rise to a complex interaction of many factors which causes the *cis*-acting regions in eukaryotes to contain a far more complicated distribution of motifs than do promoters in bacteria for instance. Characterizing promoter regions in a prokaryote is far simpler than in eukaryotes.

It is important to note that the gene-specific factors do recognize specific motifs, and that these can affect transcription rates by many orders of magnitude (50 times). Mutations that result in a promoter that is less identical to the basal transcription apparatus "consensus" should lead to lower levels of transcription but this is offset dramatically by the presence of the gene-specific motifs in the promoter or elsewhere.

Commonly accepted definitions are the 'core' and 'proximal' promoter regions of length 50 bp and 500 bp respectively [Bucher, 1990]. A length of 250 bp falls between both of these and is commonly used. With eukaryotes the default state of a gene is inactive and pol-II can only initiate transcription when the general transcription factors recruit it and the machinery binds to *cis*-acting sites, which are usually, but not always, located upstream of the gene. However, as already discussed, many transcription factors will bind outside of the promoter region to greatly assist with basal transcription, and in some cases there may not be enough "pattern" in the promoter region alone to recognize it (and a much longer sequence would be needed). Therefore, expect a lower sensitivity than otherwise as it becomes difficult to eliminate false negatives.

By its definition, the promoter region is located in the general vicinity of TSS, but there is also a distinct region known as the enhancer. Known enhancer regions are several kilobases (kb) upstream of TSS and about 100 base pairs (bp) in length. They contain several closely arranged sequences that stimulate initiation, some of which are quite similar to those of the promoter. Proteins bind to enhancer regions and interact with proteins bound to promoter regions to form a large protein complex. DNA must be coiled or rearranged to allow this interaction which complicates the *in vivo* modelling of transcription. Enhancers assist initiation from great distances, functioning in either orientation, and from either side of the gene [Lewin, 2000]. For this reason, the value of promoter prediction software is to propose putative *cis*-acting sequences, potential promoter regions in eukaryotic genomes, to prioritize the costly experimental effort. This argues for not aiming for highest specificity (tolerating certain false positives).

1.2 Evolutionary Computation and Promoter Prediction

A solution for promoter prediction is to use Evolutionary Computation (EC) to discover the predictor by learning the classification rule from a number of known promoter and non-promoter regions. The aim is to predict whether or not the sequence is a promoter (other research predicts TSS, see [Pedersen et al., 1995]).

A typical feature of promoters is a collection of very short *cis*-acting sequences or motifs. This is an important promoter characteristic that can be used for recognizing promoters. It seems sensible to use this knowledge to design the architecture of the solution which is evolved. That is, analyze the problem

in advance using human insight to determine that a certain decomposition is likely to be helpful in solving the problem; establishing an advantageous architecture for the yet to be evolved computer program [Koza, 1999]. One desirable decomposition is to:

1. automatically discover meaningful short sequences;
2. match them to locations;
3. construct meaningful relationships between the location of the sequences;

Additionally, it is desirable to maintain algorithmic simplicity and human-readability for interpretation of the resulting algorithm.

2 Algorithm Description

The combination of Genetic Programming within finite state automata (FSA) or GP-Automata (FSA-GP) was introduced by [Ashlock, 1997]. This representation and architecture was enhanced to include the explicit gathering of state information via the use of state logical functions and successfully applied to machine vision problems [Benson, 2000b]. For the promoter prediction problem the authors enhanced the algorithm further as described in this section.

2.1 Genetic Programming Modules

The objective of the GP tree structure within the GP-Automata is to find motifs within the promoter and non-promoter regions. The function set consists of a single two valued function called `GLUE`, which concatenates the terminals to produce a motif, and behaves as a logical `AND` returning true if both input conditions are met.

The terminal set is composed of nucleotides `A, C, G, and T`. For example the tree `(GLUE G (GLUE A T))` is the motif `GAT`. This tree will return true if in the genomic sequence there is a `G` in the first position, an `A` in the second, and a `T` in the third; else it returns false. To increase flexibility the terminal set is enriched to include terminals which return true for more than one given nucleotide, using the IUPAC ambiguity codes for nucleic acid sequences listed in Table 1. For example `(GLUE Y R)` will match the patterns `CA CG TA TG`.

2.2 Finite State Automaton Component

The FSA acts as the main program and each state q_i possesses one GP tree or 'function' F_i which gets called as and when the FSA tour invokes it. Each GP tree returns a boolean that indicates which of two transitions τ_i^T or τ_i^F it should follow to a new state, and consequently deciding what GP-tree function to call next. The FSA possesses a start state and an end state and can be halted after a pre-determined number of moves when it cycles.

A logical function, e.g. `NAND`, is associated with each transition τ_i and an integer value that moves a positional pointer over the DNA sequence, e.g. +10, is

Table 1. Terminals used by GP trees at states of the FSA. Wildcards (left) identify a match to two or more nucleotides (right) using the standard IUPAC convention

M	A C
R	A G
W	A T
S	C G
Y	C T
K	G T
V	A C G
H	A C T
D	A G T
B	C G T
N	A C G T

associated with each state q_i of the FSA. Both undergo genetic modification. As a tour is underway and state visits and transitions between states take place, the main program builds a logical relationship between the boolean output of state functions. The logical function stored at the forthcoming transition combines: (a) the current cumulative boolean result with (b) the boolean output of the function that is executed at the current state (see Table 4).

During transition, the positional pointer to the 300 bp long DNA sequence is moved from left to right (downstream) as many bp as indicated by the positional integer, which is held at the destination state. The convention followed being that sequences are ordered such that TSS is at vector index 250. By definition TSS is at sequence position +1 ([Lewin, 2000], pg 234):

vector index:	0	1	2	.....	249	250	252	.....	298	299
TSS sequence position:	-250	-249	-248	.....	-1	+1	+2	.....	+49	+50

The algorithm can be clarified with the aid of Fig. 1. By default, the machine initiates execution at vector index 0 (corresponding to TSS sequence position -250). Execution commences at the start state q_0. The positional integer at this state is 0 meaning that the positional pointer need not be moved. Pattern `SNBHGW` is tested and assuming it does not match, the state returns "false" and the machine transits to state q_1 (F in the arrow joining q_0 to q_1). The positional integer at state q_1 is -79 and this means that the pointer is moved 79 bp to the left (it must now subtract 78 places from vector index 299, periodicity is assumed). This places the pointer at vector index 221 (corresponding to TSS sequence position -29). Next it applies pattern `WW` of state q_1. If a match does not occur the machine transits to state q_2 where the pointer is moved 111 bp to the left, which places the pointer at vector index 110 (corresponding to TSS sequence position -140). Pattern `DV` corresponding to this state is applied there. Assume that there is no match. This causes a recursion at state q_2 and the pointer is once again moved 111 bp to the left which places the pointer at vector index 299 (corresponding to TSS sequence position +50). Pattern `DV` corresponding to this state is applied: letter D at TSS sequence position +50 and letter V at

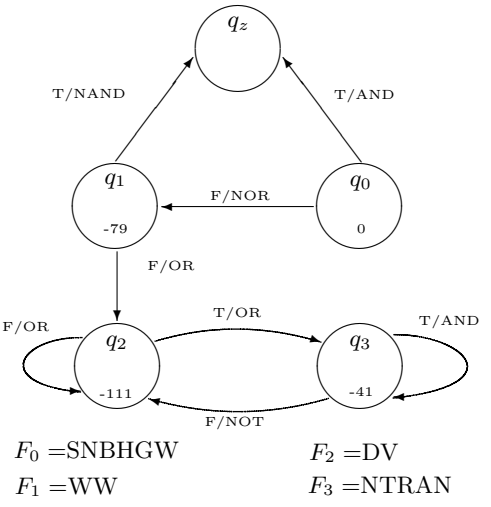

Fig. 1. One of the resulting algorithms from the 5-fold cross validation

TSS position -250 because of the periodicity[1]. Now a match occurs causing the machine to follow the transition to state q_3 where the pointer is moved 41 bp to the left placing it near TSS at TSS sequence position +9 where pattern `NTRAN` is applied.

State q_z is the so-called end state. Each FSA-GP individual carries with it an integer denoting *maximum number of transitions*. While recursion is not a very serious problem because the pointer gets moved along, this maximum number of states that can be visited limit avoids long tours through the graph.

As the machine executes it 'writes' a logical statement with the logical functions which are associated with each transition. This follows a particular convention (Table 4) which illustrates possible paths through the FSA-GP. The FSA-GP is capable of establishing quite complex decision rules for classification of patterns and has several interesting properties:

1. Since the motifs are constructed via GP tree structures, they can be of various lengths. Since IUPAC wildcards are used, they can match more than one pattern in promoters and non-promoters.
2. The algorithm also incorporates a decision-type process. In the example F_2 did not match the current location being inspected so it was looked for upstream. Only after this motif had been found was the next motif (F_3) searched for. A given motif may be looked for in many places accounting for the possibility that it may occur in more than one place.

[1] This does not correspond to the physical reality because the motif is meant to represent a short sequence capable of attracting a protein and it is only biologically meaningful for it to occur all together and at one place.

3. The classification of promoter or non-promoter is not based on only one criterion. Instead, the decision is based on a logical combination of the applied motif responses.

2.3 Evolutionary Scheme

Modification of the GP-Automata is achieved using the various mutation operators given in Table 2. Since these are many, selecting the frequency at which they should be applied is difficult. Self-adaptive mutation rates are used to overcome this difficulty [Benson, 2000a]. The particular results presented in this paper restricted the size of the GP-Automata arbitrarily to a maximum of 5 states (in this case to encourage a simple solution).

Table 2. Mutation operators

1. Add a State.	9. Exchange state GPs.
2. Delete a state.	10. Replace a GP with randomly created.
3. Mutate start of reading site.	11. Headless chicken crossover on a GP.
4. Mutate a transition.	12. Grow a subtree.
5. Cycle two states.	13. Shrink a subtree.
6. Mutate a state logical function.	14. Mutate a GP terminal.
7. Headless chicken crossover on a state.	15. Mutate a GP function.
8. Mutate the maximum number of transitions.	16. Mutate the integer on the transition.

Selection was achieved using a $(\mu+\lambda)$-EP strategy and the fitness function f = sensitivity+specificity, i.e. the negative of the sum of equation 1 and equation 2 (see below). In this paper $\mu = 125$ and $\lambda = 125$. At each generation a tournament of size 4 took place and the top μ individuals produce one offspring each via mutation giving a total of λ children. These $\mu+\lambda$ individuals were then carried over into the next generation. The *maximum number of transitions* for each machine was set to be at least 3 and never to exceed 30.

Obviously, many of these choices were arbitrary and the algorithm could be evolved in many other different ways.

3 Application

A data set of promoters and non-promoters held at fruitfly.org[2] was used to evaluate the effectiveness of the algorithm at identifying promoters. Each sequence was 300 bp long extracted 250 bp upstream and 50 bp downstream of TSS.

The data set already comes arranged to facilitate 5-fold cross validation, thus allowing algorithms to be trained and tested on five sets of data. Each file consists of 565 promoters and 890 non-promoters.

[2] http://www.fruitfly.org/seq_tools/datasets/Human/promoter/

Table 3. Parameters used in the experimentation

Parameters	
Number of runs	50
Number of Generations	1000
Population size	250
Max states	5 + halt state
Min states	3 + halt state
Max motif length	16
Min motif length	1
Tree terminals	A, C, G, T, M, R, W, S, Y, K, V, H, D, B, N
Tree functions	GLUE

The non-promoters were taken from the coding sequence (CDS) data component of this data set. Each training data set file contains 452 promoters and 712 non-promoters, and each test data set file contains 113 promoters and 178 non-promoters.

The best individual found on each of the 5 training sets was then applied to its test set, as this is the standard way to evaluate the likely performance of an algorithm.

The evolutionary process, however, produces many such algorithms at different generations but in this instance, no attempt was made to generalize (to prevent over-training). No attempt was made to carry out a search for the simplest model that fits the data. Such procedure would judge when to stop the evolution of the different parallel independent runs to avoid over-training, a procedure which may have selected more generalist individuals than those used to generate the results reported here.

Table 3 lists parameters used in the experiments and Table 5 lists the results of the 5-fold cross validation on the five test sets.

Statistical calculations on the results in Table 5 provide a measure of algorithm performance. In the context of this work, sensitivity is the likelihood that a promoter is detected given that it is present, and specificity is the likelihood that a non-promoter is detected given that it is present. Equations 1 and 2 are mathematical definitions of sensitivity and specificity. Here true positives (TP) are correctly classified promoters; false negatives (FN) are promoters classified as non-promoters; true negatives (TN) are correctly classified non-promoters; and false positives (FP) are non-promoters classified as promoters.

$$\text{sensitivity} = \frac{TP}{TP + FN} = 0.79 \tag{1}$$

$$\text{specificity} = \frac{TN}{TN + FP} = 0.92 \tag{2}$$

Table 4. Paths through the machine and resulting logical decision formulae, i.e. fusion of pattern matches. For simplicity of exposition in this table, the logical formulae presented assumes that 4 is the maximum number of state visits allowed for any FSA tour. The logical function fusing two states is stored in the next transition. Note that the logical function `NOT` is used differently from the other logical functions. `NOT` negates the output at state q_i and the running formula thus far is discarded

path 1	$F_0(-250)$ =T → HALT
path 2	$F_0(-250)$ =F → $F_1(-29)$ =T → HALT
path 3i, 3ii,...	$F_0(-250)$ =F → $F_1(-29)$ =F → $F_2(-140)$ =F → $F_2(+50)$ =F or T → ...
path 4i, 4ii,...	$F_0(-250)$ =F → $F_1(-29)$ =F → $F_2(-140)$ =T → $F_3(-181)$ =F or T → ...

path 1	T = PROMOTER
path 2	F NAND T = PROMOTER
path 3i	((F OR F) OR F) OR F = NON PROMOTER
path 3ii	((F OR F) OR F) OR T = PROMOTER
path 4i	((F OR F) OR T) NOT F = PROMOTER
path 4ii	((F OR F) OR T) AND T = PROMOTER

Table 5. Results of five fold cross validation tests (for the best individual from each set of 50 runs). TP, TN, FP, FN defined in text

	TP	TN	FP	FN
Test 1	92	159	19	21
Test 2	87	165	13	26
Test 3	86	168	10	27
Test 4	91	167	11	22
Test 5	90	160	18	23
$\sum$	446	819	71	119

As already stated the fitness measure that obtained these results combined sensitivity and specificity objectives equally.

4 Discussion

Inductive learning by evolution was used to discover knowledge about promoter classificational identification. This can be compared with the straightforward application of knowledge about promoters. Several texts and papers [Lewin, 2000] [Bucher, 1990] establish that the following patterns are useful for recognizing eukaryotic promoters:

1. `TATA` pattern located approximately 30 bp upstream of TSS;
2. `CAAT` or `TAAC` located approximately 75 bp upstream of TSS;
3. `GGGCGG` or `GGCGGG` located approximately 90 bp upstream of TSS;
4. `YYANWYY` located at TSS (known as Inr pattern);

Table 6 presents TP, FP, specificity and sensitivity values obtained by applying the patterns listed above individually and together, to the data from one of the

Table 6. Application of known basal transcription patterns to the data set

	patterns	location (buffer)	TP	FP	sensitivity	specificity
Train 1	all of below		260	69	0.58	0.90
	TATA	-30 (10:10)	203	14	0.45	0.98
	CAAT or TAAC	-75 (10:10)	58	32	0.13	0.95
	GGGCGG or GGCGGG	-90 (10:10)	20	6	0.04	0.99
	Inr	0 (3:3)	45	20	0.10	0.97
Test 1	all of below		66	20	0.59	0.89
	TATA	-30 (10:10)	50	5	0.45	0.97
	CAAT or TAAC	-75 (10:10)	16	11	0.14	0.94
	GGGCGG or GGCGGG	-90 (10:10)	9	1	0.08	0.99
	Inr	0 (3:3)	6	6	0.05	0.97

5 folds. It appears that nearly half of the promoters are either TATA-less or have variations of the patterns listed and thus cannot be predicted easily. The low statistical significance of these consensus sequences and the better success rate of the TATA-box among them is a typical finding (the position-weight matrices obtained by [Audic and Claviere, 1997] on vertebrate sequences from the Eukaryotic Promoter Database (EPD)). Finally, it was not possible to discover the so called TATA downstream promoter pattern in this data set which sometimes appears in DNA.

Figure 1 is a successful algorithm arising in one of the 5-fold cross validation experiments. The four discovered sequences are very short, quite general, and in themselves cannot be discriminatory. Note from Table 4, however, that pattern WW, which stands for AA, AT, TA or AA, gets applied at -29 and that this seems plausible as this is the same TSS sequence location where the TATA box is expected. Yet, it would seem that the evolved FSA relied on a lot of movement to achieve the promoter recognition, most of them evolving to use 30 for the maximum number of transitions.

It is much harder to discriminate promoter regions from non-coding sequences than from CDS. A future implementation will evolve the FSA-GP using non-coding sequences (rather than CDS) for examples of non-promoter regions.

5 Conclusions

Ashlock's GP-Automata [Ashlock, 1997] with logical functions [Benson, 2000b] was enhanced with positional pointers to evolve an FSA-GP promoter predictor. Early results show that it has potential to discover candidate promoter sequences in primary sequence data. EC was used to: (a) discover motifs of various lengths in automata states; (b) discover integer values stored at states to position the testing of the motifs on the DNA string; and (c) combine the motif matches using logical functions to arrive at a promoter identification decision. The resulting FSA-GP could be said to operate comparably to an explicit sequence similarity-based method which looks for the local accumulation in the 250 bp window of

matches with previously recognized transcriptional elements [Pestridge, 1995]. It is argued that this scheme is consistent with domain knowledge. For example, it is known that there exist a number of short *cis*-acting sequences in promoters (popular ones appear in Table 6) and that these tend to occur at fixed locations. Some are expected to occur more than once but many occur only once and at a particular location.

The scheme had the required simplicity to enable analysis of the resulting programs to potentially facilitate discovery of putative and meaningful *cis*-acting sites. In spite of the small size imposed (a maximum of 5 transitions) it had the required complexity to arrive at context sensitive decisions for classification of sequences.

The more successful FSA-GPs detected promoters by relying on a great number of visits, and on building an involved truth table of numerous matches to very small (2 or 3 bp long) patterns. More analysis of results on test cases is needed to determine whether the emerging promoter prediction strategy is correct biologically. If it is not, future research should establish how to design the FSA-GP to reflect molecular biology more faithfully than with the current approach.

Simpler EC schemes than the one presented in this paper can be applied to predict core and proximal promoters (weighted matrices with evolution of the weights, a weight is associated with each nucleotide type at every location in the sequence). The scheme presented in this paper could become useful to identify *cis*-acting motifs for gene regulation *in vivo* as part of a computational system that processes much longer DNA sequences (including enhancers and promoters upstream, downstream and inside of the gene). A property of the FSA-GP is code reuse and the creation of structure (function subprograms at the GP states of the FSA-GP). Both can become important for discovery of subtle mechanisms. The algorithm presented applies motifs at precise locations in a sequential testing tour and, if required, a more powerful variant could be less precise when applying its motifs or work in a holistic (parallel) fashion.

Acknowledgements. The authors would like to thank the anonymous reviewers for their careful reading and suggestions, and David Corne for bringing the promoter prediction problem to their attention.

References

[Ashlock, 1997] Ashlock, D. (1997). GP-Automata for dividing the dollar. In Koza et al. (Eds) *Genetic Programming: Proceedings of the Second Annual Conference*, 18–26, Stanford University.

[Audic and Claviere, 1997] Audic, S. and Claviere, J.-M. (1997). Detection of eukaryotic promoters using Markov transition matrices. Computers Chem. Vol 21, No 4, pp 223–227, 1997.

[Benson, 2000a] Benson, K.A. (2000). Evolving Finite State Machines with Embedded Genetic Programming for Automatic Target Detection within SAR Imagery. In *Proceedings of the Congress on Evolutionary Computation*, 1543–1549, La Jolla, San Diego, USA.

[Benson, 2000b] Benson, K.A. (2000). Performing Automatic Target Detection with evolvable finite state automata. Journal of Image and Vision Computing, Volume 20, Issue 9–10, Elsevier.

[Bucher, 1990] Bucher, P. (1990). Weight matrix description of four eukaryotic RNA Polymerase II promoter elements derived from 502 unrelated promoter sequences. Journal of Molecular Biology, 212, pp. 563–578.

[Cattaneo et al., 2002] Cattaneo E., Rigamonti D.,Zuccato C. The Enigma of Huntington's Disease. *Scientific American*, December 2002.

[Fessele et al., 2002] Fessele S., Maier H. Zischek C., Nelson P.J. and Werner T. Regulatory context is a crucial part of gene function. *Trends in Genetics*, vol 18, issue 2, pp 60–63, 2002.

[Handley, 1995] Handley S. Predicting whether or not a nucleic acid sequence is an *E. coli* promoter region using Genetic Programming. Proc. of First International Symposium on Intelligence in Neural and Biological Systems (INBS'95), pp. 122–127, IEEE Comp. Soc. Press, 1995.

[Hannenhalli and Levy, 2001] Hannenhalli S. and Levy S. (2001). Promoter prediction in the human genome. In *Proceedings of the 9th International Conference on Intelligent Systems for Molecular Biology.* Copenhagen, Denmark, July 21–25, 2001, Bioinformatics, Vol 17, Supplement 1, pp. S90–S96. ISSN: 1367–4803.

[Koza, 1999] John R. Koza (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*, MIT Press.

[Lewin, 2000] Benjamin Lewin (2000). Genes VII, Oxford University Press.

[Orphanides et al., 1996] Orphanides, G., Lagrange, T., Reinberg, D.. The general transcription factors of RNA polymerase II. *Genes. Dev.*, vol. 10, 2657–2683, 1996.

[Tsonis, 2003] Tsonis P.S. Anatomy of Gene Regulation: A Three-Dimensional Structural Analysis. Cambridge University Press, 2003.

[Pedersen et al., 1995] Pedersen, A.G. and Engelbrecht, J. Investigations of *Escherichia coli* promoter sequences with ANN. Proceedings: Third International Conference on Intelligent Systems for Molecular Biology, pp. 292–299, 1995.

[Pedersen et al., 1999] Pedersen, A.G., Baldi, P., Chauvin, Y., Brunak, S. The biology of eukaryotic promoter prediction – a review. *Computers and Chemistry*, vol. 23, 191–207, 1999.

[Pestridge, 1995] Pestridge D.S. (1995) Predicting pol-II promoter sequences using transcription factor binding sites. *Journal of Molecular Biology*, vol. 249, 923–932, 1995.

[RGEP, 2002] Roche Genetics Education Program CD-ROM. *Scientific American*, December 2002.