Brock University

Department of Computer Science

Evolving Protein Motifs Using a Stochastic Regular Language with Codon-Level Probabilities

Brian J Ross
Technical Report # CS-02-11
May 2002

To be presented at the International Conference on Artificial Intelligence and Soft Computing (ASC 2002), Banff, July 2002.

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Evolving Protein Motifs Using a Stochastic Regular Language with Codon-Level Probabilities

Brian J. Ross
Deptartment of Computer Science
Brock University
St. Catharines, ON, Canada L2S 3A1
email: bross@cosc.brocku.ca

ABSTRACT
Experiments involving the evolution of protein motifs using genetic programming are presented. The motifs use a stochastic regular expression language that uses codon-level probabilities within conserved sets (masks). Experiments compared basic genetic programming with Lamarckian evolution, as well as the use of "natural" probability distributions for masks obtained from the sequence database. It was found that Lamarckian evolution was detrimental to the probability performance of motifs. A comparison of evolved and natural mask probability schemes is inconclusive, since these strategies produce incompatible characterisations of motif fitness as used by the genetic programming system.

KEY WORDS
genetic programming, stochastic regular expressions, protein motif

1 INTRODUCTION

Identifying biosequence characteristics at the primary structure or sequence level is commonly done, and has been a subject of research for a number of years [1]. Advantages of sequence-level characterizations of biosequences are their usefulness as means of database lookup, and their tractability in terms of interpretation and automatic acquisition using a variety of algorithms and machine learning techniques. A popular sequence-level motif is the use of *regular expressions* to characterize the region shared by a family of related proteins sequences. Protein sequence databases such as PROSITE use this style of motif representation [3].

Genetic programming (GP) is a machine–learning paradigm that uses a genetic algorithm to evolve populations of programs [6]. GP has been applied to a number of bioinformatics problems, including the evolution of motif expressions for protein sequences. Hu uses a combination of GP and local motif refinement to evolve protein motifs for unaligned sequences [5]. The target language used is identical to the regular expression language used by PROSITE. Some of his evolved motifs were nearly identical to their PROSITE equivalents. Koza *et al.* apply GP towards motif synthesis for a number of protein families [7]. Their target language is a constrained one that does not use variable-length gaps. In one instance, the evolved motif improved upon the accepted PROSITE motif expression.

Earlier work by this author proposed the use of stochastic regular expressions (SRE-DNA) as a motif language for protein sequences. Firstly, SRE-DNA motifs were synthesized for aligned protein sequences using GP [12]. A number of restricted variations of SRE-DNA were studied, to determine the most practical subset of SRE-DNA for the protein motif problem. Next, SRE-DNA motifs for unaligned sequences were successfully evolved [13]. In summary, motifs for small- to moderate-sized aligned or unaligned sequences were shown to be directly evolvable with GP, without the need for local optimization of motif fields. Evolution benefits from the stochastic nature of the motif language, since sequence probabilities are directly incorporated into fitness scores within the GP system. Furthermore, variable-length gaps within motifs are elegantly handled by SRE-DNA's probabilistic *skip* expression.

This research continues work in evolving protein motifs by addressing a few issues encountered in the aforementioned earlier research. One possible weakness of previous SRE motif representations is that uniform probability distributions are used for converved sets or *masks* of codons. For example, in the mask *[filmv]*, each codon has an equal probability of 0.20 (i.e. 1/(mask size)). It seems reasonable that a mask with variable probabilities for different elements might yield more precise probabilistic characterizations of protein sequences.

An inadvertent weakness in the implementation of SRE-DNA used in previous work is that motifs permitted gap expressions to terminate motif expressions, for example,

$$(sequence\ \ expression) : x^{*.5}$$

Here, the skip term $x^{*.5}$ denotes a variable-length gap. Although this motif schema is not erroneous, we found that it appeared in virtually all evolved motifs. This happens because the skip term reduces the size of sequences to be recognized by the motif, by skipping a large terminating subsequence of each example. This is advantageous dur-

ing evolutionary computation, since expression fitness is improved. Such loopholes are often exploited by genetic algorithms. A more robust motif would not skip important portions of sequences.

To address these problems, this paper uses a refined SRE-DNA motif language that incorporates codon-level probabilities within mask terms, and prohibits skip expressions at the end of motif expressions. In addition, the language uses a minimal number of regular operators, foregoing Kleene iteration and choice operators. This makes it similar to PROSITE's non-probabilistic regular language [3]. The notion of canonically conserved codon sets is borrowed from Hu [5], which should improve the rate of evolution by introducing biologically meaningful conserved sets of amino acids into motif expressions. Additional runs that use local search (Lamarckian evolution) to refine mask terms are undertaken, in hopes of improving mask expression quality. We also investigate an alternative approach to mask probabilities, by using empirically measured probabilities from the sequence database within masks.

The experimental setup is presented in Section 2. Results of the experiments are given in Section 3. Some summary discussions and conclusions end the paper in Section 4.

2 EXPERIMENT DESIGN

2.1 Stochastic Regular Expressions

$$
\begin{aligned}
exprtop &::= expr{:}mask \\
expr &::= guard \mid guard{:}expr \\
guard &::= mask \mid mask{:}skip \\
skip &::= x^{+p} \\
mask &::= [a_1(n_1), a_2(n_2), ..., a_k(n_k)]
\end{aligned}
$$

Figure 1. SRE-DNA Grammar

The motif language used is stochastic regular expressions, or SRE-DNA. It is based on a fuller language in [10], and is refined for the problem of biosequence representation. The language presented here is related to SRE-DNA languages used in earlier work [12, 13].

The BNF grammar for SRE-DNA is given in Figure 1. SRE-DNA is a regular expression language [4], and its operators have the usual semantics, but with a probabilistic model overlaying them. See [10, 12, 13] for more details about the semantics of SRE and SRE-DNA.

The salient features of this variant of SRE-DNA are as follows. Skip expressions cannot terminate a motif. This is ensured in the *exprtop* rule, which forces a mask term to end the expression. The *guard* rules ensure that there cannot be consecutive skips terms. Probability values are introduced in *skip* and *mask* rules. The skip term's p is a probability within some user-specified range, for example, between 0.1 and 0.5. The skip expression uses the "+" notation, where $E^+ = E : E^*$, and E^* is Kleene iteration. A mask represents a set of possible codons that can arise at that position of the motif. A mask consists of a set of codons a_i paired with integers n_i. Each a_i has a probability $n_i / \sum_{j=1}^{k} n_j$. For example, in the mask $[f(2), i(15), l(20), v(6)]$, the codon i has a probability $15/43$.

2.2 Biosequence data

Table 1. Protein data summary

Protein (accession #)	I_{max}	**W**	S_1	S_2
Zinc finger 1, C2H2 (PS00028)	0.19	25	29	678
Zinc finger 2, C3HC4 (PS00518)	0.19	10	21	168
Sugar transport 1 (PS00216)	0.32	18	28	190
Sugar transport 2 (PS00217)	0.49	26	27	178
Snake toxin (PS00272)	0.49	21	18	127

Aligned training and testing sequences were obtained from the PROSITE and TrEmbl online databases.[1] Table 1 summarizes the data. In this table, the accession # is the PROSITE identifier; I_{max} is the maximum skip iteration limit permitted in SRE-DNA motifs; *W* is the maximum sequence length or window length that motifs need to cover; S_1 is the size of the training set; and S_2 is the size of the testing set.

Negative training and testing sets are composed of random sequences of codons. Each negative example is the size of a typical positive sequence. The negative sets are the same size as the respective training and testing sets for the protein in question.

2.3 GP Parameters

The DCTG-GP genetic programming system is used [11]. DCTG-GP is a grammatical GP system implemented in Prolog. The target language is defined with a definite clause translation grammar (DCTG), which is a logic-based attribute grammar. The SRE-DNA grammar in figure 1 is directly encoded in the system with DCTG rules. These rules also include the semantics of the operators, which means that the grammar encodes an interpreter for the language.

Table 2 lists parameters used for GP runs, and are based on earlier work outlined in Section 1 . The initial population is oversampled, and culled at the beginning of a run. Reproduction may fail, for example, due to tree size limitations. A maximum of 3 reproduction attempts are undertaken with selected parents before the reproduction is discarded, and new parents are selected. The terminals are

[1] http://ca.expasy.org/

Table 2. GP Parameters

Parameter	Value
GA type	generational
Functions	SRE-DNA
Terminals	amino acid codons, integers, probabilities
Population size (initial)	1500
Population size (culled)	1000
Unique population	yes
Maximum generations	150
Maximum runs	20 (10 Lamarck)
Tournament size	4
Elite migration size	10
Retries for reproduction	3
Prob. crossover	0.90
Prob. mutation	0.10
Prob. internal crossover	0.90
Prob. terminal mutation	0.75
Prob. SRE crossover	0.25
Prob. SRE mutation	0.30
SRE mutation range	0.1
Max. depth initial popn.	12
Max. depth offspring	24
Min. grammar prob.	10^{-12}
Min. skip limit	0.10
Max. mask size	6
Lamarckian elite set size	10
Lamarckian hill-climbing limit	5

the subset of amino acid codons, determined by the alphabet used in the positive training examples.

Crossover and mutation use the methods commonly applied by grammatical GP systems that denote programs with derivation trees; see [11] for details. Some SRE specific crossover and mutation operators are used. SRE crossover permits mask elements in two parents to be merged together. SRE mutation implements a number of numeric and mask mutations, for example, number perturbation, and mask element insertion, deletion, and alteration.

The minimum grammar probability value specifies the minimal probability used by the SRE evaluator before an expression interpretation is pre-empted. This stops the interpretation of expressions with negligible probabilities.

Figure 2 shows the 13 conserved substitution groups of amino acids used in the experiments, which were discovered empirically in [2]. The tree shows the hierarchical relationships between these groups, as those lower in the tree are subsets of parent groups. An analysis of the PROSITE database shows that 81.2% of the conserved sets used in all motifs are covered by these 13 predefined groups. These same 13 groups are incorporated into SRE-DNA expression generation, in an attempt to create more rational masks. There is a 50% probability that a predefined conserved group is used when a mask is generated. Otherwise, a random set is generated.

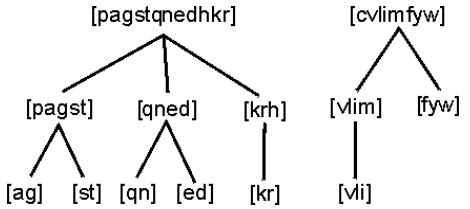

Figure 2. Conserved Groups of Amino Acids

2.4 Lamarckian Evolution

Each experiment involves 10 runs with basic GP, and 10 runs with Lamarckian evolution [9]. The Lamarckian runs are economical, as they apply local refinement to 10 fittest individuals in the population. The local refinement uses a hill-climbing search strategy, in which SRE terminal mutation is applied to mask and skip terms. An expression is repeatedly mutated until 5 failed attempts are tallied, at which time the hill-climber terminates.

After every run, the best solution is optimized with respect to mask fields by deleting extraneous mask elements. This improves probability scores for the motif. The optimization routine uses a hill-climber, in which every mask with more than 1 element is processed as follows: an element is removed, and if the overall fitness of the motif improves, that element is deleted permanently.

2.5 Fitness evaluation

A minor modification of the fitness evaluation strategy from [12] is used. Fitness evaluation measures the ability to recognize N positive training examples, and to reject N negative examples:

$$Fitness = N + NegFit - PosFit$$

where *NegFit* and *PosFit* are the negative and positive training scores. Positive example scoring is calculated as:

$$PosFit = \sum_{e_i \in Pos} maximum(Fit(e'_i))$$

where Pos is the set of positive training examples, and e'_i is a suffix of example e_i (ie. $e_i = se'_i, |s| \geq 0$). For each example in Pos, a positive test fitness Fit is found for all its suffixes, and the maximum of these values is used for the entire example. Fitness evaluation considers both the probability of recognizing an example, and the length of the sequence recognized:

$$Fit(e) = \begin{cases} \frac{1}{2}\left(Pr(s_{max}) + \frac{|s_{max}|}{|e|}\right) & : Pr(s_{max}) \leq \frac{|s_{max}|}{|e|} \\ \frac{1}{2}\left(\frac{|s_{max}|}{|e|}\right) & : otherwise \end{cases}$$

Here, s_{max} is the longest recognized prefix of e, $|s_{max}|$ is its length, and $Pr(s_{max})$ is its probability of recognition. The fitness pressure obtained with *Fit* is to recognize an entire example string with a high probability. The conditional formulae ensure against degenerate conditions, in which small terms with high probabilities overwhelm the population. Negative fitness scoring is calculated as:

$$NegFit = maximum(Fit(n_i)) * N$$

where $n_i \in Neg$ (negative examples). The highest obtained fitness value for any recognized negative example suffix is used for the score.

2.6 Natural mask probabilities

An alternative method for obtaining mask probabilities is investigated. A given SRE-DNA motif is applied to the database of training and testing sequences, and the frequencies of using each mask element during interpretation is recorded. These frequencies are then incorporated into the original motif. This results in a "natural" probability distribution for mask elements.

3 RESULTS

Table 3 summarizes the performance of the experiments. *Type* denotes the basic (B) or Lamarckian (L) runs. *avg* $\overline{pr}$ is the average probability for the training example, averaged over all the solutions obtained in the 10 runs. The *best* $\overline{pr}$ is the average example probability for the best solution of the 10 runs. The next 3 columns are testing results for the best solution in *best* $\overline{pr}$. The *%tp* is the percentage of true positives recognized, and $\overline{pr}$ is the average probability over the positive testing set. Similarly, *%fn* is the percentage of false negatives (negative examples falsely identified as positive). High *%tp* and low *%fn* scores are desirable.

Analyzing the results thus far, note that there are two aspects of performance that must be considered when evaluating the solution motifs: overall probabilities during sequence recognition, and overall true positive and false negative performance when recognizing testing sets. This dichotomy is seen in the construction of the fitness strategy in Section 2.5. These measurements are somewhat mutually exclusive, as one must decide whether it is more important to recognize fewer examples with a higher probability, or more examples with a lower probability. Fortunately, the fitness strategy used balances these performance aspects fairly well.

In terms of probability performance, the Lamarckian runs perform worse than the basic GP motifs in all but one case (snake toxin). This indicates that Lamarckian optimization of terminals is not beneficial. The Lamarckian solutions have better training performance in terms of true positive scores. However, given that all the best solutions (B and L) almost always identified all the training examples, this is unclear whether this testing performance difference is due to the use of local optimization or not.

Table 4. PROSITE comparison

Family	#FP (tot)	#FN (tot)
Zinc 1	4 (12)	1 (3)
Zinc 2	0 (5)	0 (12)
Sugar 1	9 (10)	7 (10)
Sugar 2	4 (10)	9 (10)
Snake	-	0 (10)

A selection of false positive and false negative sequences were obtained from PROSITE. These are sequences that are erroneously classified by the accepted PROSITE motif. These sequences were given to the best SRE-DNA motifs from the basic GP runs, and the results are shown in Table 4. Here, low FP scores and high FN scores are preferred.

The final 3 columns of Table 3 report testing performance for the natural probability distributions for masks, which were derived for the motifs from the basic GP runs only. The "natural masks" resulted in higher probability performance only for the zinc finger families, while other families resulted in lower probabilities. The *% higher* and *% lower* columns indicate the percentage of the testing set that saw improved or worse probabilities with the use of natural masks, compared to the original evolved masks. These measurements indicate whether a natural mask might improve the scores for more examples, even though the overall probability for the testing set was reduced. Indeed, this was the case for the Sugar 1 and Snake examples.

Figure 3 shows a few evolved motifs. The figure shows the PROSITE regular motif (P), an example motif evolved in earlier work in [12] (O), the best evolved SRE-DNA motif having evolved mask probabilities (E), and the same motif with natural mask probabilities (N). Note that the old motifs (O) use a different variation of SRE-DNA than used here, as masks terms have the same probabilities, and entire terms can be subject to iteration (see the Sugar 2 (O) example). First, it is clear that the motifs evolved here (E and N motifs) have solved the problem of skip terms terminating motifs; both the O motifs have terminating skip terms. The similarity between the PROSITE and SRE-DNA motifs for zinc finger 2 is especially apparent. The new motifs also have more occurences of mask terms than the old motifs. This is clear for zinc finger 2, where the old (O) motif has no masks at all. The use of conserved groups (Figure 2) during evolution was probably a strong factor in their more frequent occurrence.

Table 5 shows some testing comparisons between the best basic solutions from Table 3 (B entries) and some motifs evolved in earlier work in [12]. The motifs from this earlier work generally show better performance than the motifs from this research. However, those motifs were sim-

Table 3. Motif Performance

Family	Type	*Training* avg $\overline{pr}$	*Training* best $\overline{pr}$	*Testing (best soln.)* %tp	*Testing (best soln.)* $\overline{pr}$	*Testing (best soln.)* %fn	*Testing, natural prob. (best soln.)* avg $\overline{pr}$	*Testing, natural prob. (best soln.)* % higher	*Testing, natural prob. (best soln.)* % lower
Zinc 1	B	4.54e-8	2.03e-7	86.4	3.02e-7	0	3.52e-7	37.9	48.7
	L	2.78e-8	7.93e-8	83.0	7.83e-8	0	-	-	-
Zinc 2	B	6.48e-3	1.15e-2	72.6	8.97e-3	0	2.26e-2	66.1	6.5
	L	6.23e-3	6.88e-3	85.7	3.57e-3	0	-	-	-
Sugar 1	B	2.14e-6	7.52e-6	74.7	1.65e-6	0	1.00e-6	47.4	32.6
	L	8.08e-7	1.40e-6	81.6	4.82e-7	1.6	-	-	-
Sugar 2	B	9.82e-8	3.68e-7	86.0	1.30e-7	1.7	2.71e-8	25.8	60.1
	L	7.63e-8	2.22e-7	88.2	2.38e-7	0.6	-	-	-
Snake	B	4.80e-5	1.23e-4	55.9	8.64e-5	0	7.56e-6	42.5	13.4
	L	7.94e-5	2.58e-4	70.9	6.07e-5	0	-	-	-

Zinc 2

$P \Rightarrow c : x : h : x : [filmvy] : c : x(2) : c : [ailmvy]$

$O \Rightarrow c : x^{+.1} : h : x^{+.19} : c : x^{+.19} : c : x^{+.1}$

$E \Rightarrow c : x^{+.1} : h : x^{+.1} : [(f, 97), (i, 65), (l, 6), (m, 49), (v, 3)] : c : x^{+.19} : c : [(i, 79), (l, 31), (m, 89), (v, 22)]$

$N \Rightarrow c : x^{+.1} : h : x^{+.1} : [(f, 112), (i, 6), (l, 16), (m, 4), (v, 5)] : c : x^{+.19} : c : [(i, 71), (l, 43), (m, 7), (v, 22)]$

Sugar 2

$P \Rightarrow [filmv] : x : g : [afilmv] : x(2) : g : x(8) : [fily] : x(2) : [eq] : x(6) : [kr]$

$O \Rightarrow [filmv] : x^{+.19} : (g : x^{+.48} : g : x^{+.48} : [fgily] : x^{+.48} : [ailtv] : (x)^{+.48})^{+.21}$

$E \Rightarrow [(f, 23), (i, 97), (l, 27), (m, 80), (v, 32)] : x^{+.12} : g : x^{+.48} : g : x^{+.48} : [(f, 23), (t, 27), (v, 32)] : x^{+.48}$
$: [(e, 43), (g, 91), (l, 27), (v, 89)] : x^{+.48} : [(a, 39), (p, 85), (t, 23), (v, 32)] : x^{+.47} : [(k, 7), (r, 73)]$

$N \Rightarrow [(f, 15), (i, 52), (l, 56), (m, 7), (v, 50)] : x^{+.12} : g : x^{+.48} : g : x^{+.48} : [(f, 35), (t, 66), (v, 79)] : x^{+.48}$
$: [(e, 44), (g, 22), (l, 64), (v, 50)] : x^{+.48} : [(a, 69), (p, 36), (t, 20), (v, 55)] : x^{+.47} : [(k, 40), (r, 140)]$

Figure 3. Motif Comparisons: PROSITE (P), Old SRE-DNA (O), SRE-DNA evolved mask (E), SRE-DNA natural mask (N)

Table 5. Testing Performance Comparison (best solns)

Family	*New* %tp	*New* %fn	*Old* %tp	*Old* %fn
Zinc 1	86	0	93	1
Zinc 2	73	0	100	0
Sugar 1	75	0	88	0
Sugar 2	86	2	100	12
Snake	56	0	51	0

pler than the ones here, in that skip expressions were permitted at the end of the motifs. This is readily seen in the two examples shown in Figure 3 for the old (O) entries, and especially in the sugar transport 2 motif, which ends with a $x^{+.48}$ term.

4 CONCLUSION

The motifs in this paper successfully solve the problem of skip terms terminating motifs, as were common in earlier work [12, 13]. A result of this, however, is that the sequence recognition problem is more difficult. Consequently, testing scores reported in Table 3 were often worse than earlier experiments with respect to true positive recognition rates. In other words, the motifs are overly discriminatory. On the other hand, these motifs yield higher overall probabilities for some protein families compared to their equivalents in previous work. It was also found that the evolved motifs made more use of mask terms, likely due to the use of conserved groups during expression generation.

It is surprising that Lamarckian evolution is not beneficial in our runs. Although the Lamarckian solutions were always more concise with respect to mask size, this did not result in better probability scores, contrary to our expectations. It is unclear whether the higher testing scores for the Lamarckian solutions is accidental, or an effect of local optimization.

A new question arising from this research is how the evolved and natural mask probability distributions compare to one another. It was an unexpected discovery that the evolved mask probability distributions did not conform to the training sequence distributions. Moreover, when these masks were replaced with the natural probability distributions, motif probability scores often decreased. After some analyis, it was determined that the evolved mask probabilities arise from a combined effect of the GP fitness strategy, and the nature of probability assignment by SRE-DNA expressions. Not all the sequences belonging to a family have equal probabilities. Shorter sequences typically have higher probabilities than longer ones, because skip expressions are not used as often. Each skipped element in a longer sequence reduces the overall probability. During fitness evaluation, the scores for these shorter, high probability sequences often dominates the motif's fitness score, and often by a significant amount. This in turn directly influences the composition of mask probability distributions: mask probabilities are favoured that result in higher scores for the dominant higher-probability, short sequences.

To confirm this hypothesis, some separate genetic algorithm runs were performed to evolve mask probabilities directly for given motifs. The results were invariably the same: one element dominates the probability for the entire mask; for example, *[(l,99),(i,1),(m,1),(v,1)]*. This occurs because this *l* codon is used within short, high-probability sequence(s), and the overall score for the motif made higher when this codon is given top priority.

One solution that we investigated is the use of natural probabilities. Although this does not always result in better overall scores for motifs, it may result in better performance for more sequences in a family. Whether this is a preferable solution in general is still an open question. A possibility worth investigating is a new scoring scheme, that attempts to factor in sequence length with the probability of a sequence. In any case, since SRE-DNA motifs naturally favour short sequences, there is no clear solution to this issue at present.

Work is under way to compare the motifs obtained here with conventional statistical motifs, for example, Hidden Markov Models [8]. The results of this work may lend insight into the above mask probability issue.

Acknowledgement: Thanks to Sheridan Houghton and Betty Ombuki for their helpful comments. This research is supported through NSERC Operating Grant 138467.

References

[1] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the Automatic Discovery of Patterns in Biosequences. *Journal of Computational Biology*, 5(2), 1998, 279–305.

[2] C.G.Nevill-Manning, T.D. Wu, and D.L. Brutlag. Highly specific protein sequence motifs for genome analysis. *Proc. Natl. Acad. Sci. USA*, 95, 1998, 5865–5871.

[3] K. Hofmann, P. Bucher, L. Falquet, and A. Bairoch. The PROSITE database, its status in 1999. *Nucleic Acids Research*, 27(1), 1999, 215–219.

[4] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. (Addison Wesley, 1979).

[5] Y.-J. Hu. Biopattern Discovery by Genetic Programming. In J.R. Koza *et al*, editor, *Proceedings Genetic Programming 1998*, (San Francisco: Morgan Kaufmann, 1998), 152–157.

[6] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. (MIT Press, 1992).

[7] J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. (San Francisco: Morgan Kaufmann, 1999).

[8] A. Krogh, M. Brown, I.S. Mian, K. Sjolander, and D. Haussler. Hidden Markov Models in Computational Biology. *Journal of Molecular Biology*, 235, 1994, 1501–1531.

[9] B.J. Ross. A Lamarckian Evolution Strategy for Genetic Algorithms. In L. Chambers (Ed.), *The Practical Handbook of Genetic Algorithms*, 3 (Boca Raton: CRC Press, 1997), 1–16.

[10] B.J. Ross. Probabilistic Pattern Matching and the Evolution of Stochastic Regular Expressions. *International Journal of Applied Intelligence*, 13(3), 2000, 285–300.

[11] B.J. Ross. Logic-based Genetic Programming with Definite Clause Translation Grammars. *New Generation Computing*, 19(4), 2001, 313–337.

[12] B.J. Ross. The Evaluation of a Stochastic Regular Motif Language for Protein Sequences. In L. Spector *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, (San Francisco: Morgan Kaufmann, 2001), 120–128.

[13] B.J. Ross. The Evolution of Stochastic Regular Motifs for Protein Sequences. *New Generation Computing*, 20(2), 2002, 187–213.