Research article

Sequence classification with side effect machines evolved via ring optimization

Andrew McEachern\textsuperscript{a}, Daniel Ashlock\textsuperscript{a,∗}, Justin Schonfeld\textsuperscript{b}

\textsuperscript{a} Department of Mathematics and Statistics, University of Guelph, Guelph, Ontario, Canada N1G 2W1
\textsuperscript{b} Biodiversity Institute of Ontario, University of Guelph, Guelph, Ontario, Canada N1G 2W1

\textbf{A R T I C L E I N F O}

\textbf{Article history:}
Received 8 April 2011
Received in revised form 29 March 2013
Accepted 31 March 2013

\textbf{Keywords:}
Evolutionary computation
Side effect machine
Sequence classification
Finite state machine
Representation
Ring species

\textbf{A B S T R A C T}

The explosion of available sequence data necessitates the development of sophisticated machine learning tools with which to analyze them. This study introduces a sequence-learning technology called side effect machines. It also applies a model of evolution which simulates the evolution of a ring species to the training of the side effect machines. A comparison is done between side effect machines evolved in the ring structure and side effect machines evolved using a standard evolutionary algorithm based on tournament selection. At the core of the training of side effect machines is a nearest neighbor classifier. A parameter study was performed to investigate the impact of the division of training data into examples for nearest neighbor assessment and training cases. The parameter study demonstrates that parameter setting is important in the baseline runs but had little impact in the ring-optimization runs. The ring optimization technique was also found to exhibit improved and also more reliable training performance. Side effect machines are tested on two types of synthetic data, one based on GC-content and the other checking for the ability of side effect machines to recognize an embedded motif. Three types of biological data are used, a data set with different types of immune-system genes, a data set with normal and retrovirally derived human genomic sequence, and standard and nonstandard initiation regions from the cytochrome-oxidase subunit one in the mitochondrial genome.

© 2013 Elsevier Ireland Ltd. All rights reserved.

1. Introduction

Analyzing and classifying DNA sequences according to biological function can be extremely difficult. A substantial part of the difficulty is selecting a distance measure between sequences. Sequences with similar biological function can be of different length, or have differing arrangements of the biologically functional elements. The Basic Local Alignment Search Tool (BLAST) (Altschul \textit{et al.}, 1990) has been the workhorse of DNA string classification in biology, and there are no known techniques that can outperform BLAST when the DNA sequences are long enough and possess substantial homology. When analyzing shorter DNA sequences, BLAST leaves something to be desired in accuracy and efficiency. If the key to classifying a set of DNA sequences is a small substring that occurs at different positions then a \textit{motif searcher} is a more appropriate tool. If the secondary structure of the RNA derived from the DNA or small substring statistics are the important factor, then special purpose software may be needed.

\textit{Side effect machines (SEMs)} (Ashlock and Warner, 2008c) have been shown to be able to classify short DNA sequences. We will offer mathematical demonstrations that SEMs have the potential to classify sequences based on both motifs and small-substring statistics and verify experimentally that evolutionary computation can train them to do so in practice. This study uses two sets of synthetic data, one based on GC-content and the other based on the presence or absence of a particular motif. The study then experimentally tests the ability of SEMs trained via evolutionary computation to classify three biological data sets. The first is a previously published data set containing different types of immune genes. The second is a collection of sequences from the human genome some of which contain endogenous human retroviruses and the rest of which do not. The third is a novel data set derived from the initiation regions on the mitochondrial cytochrome-oxidase subunit one \textit{(Cox1)} gene. These data are drawn from regions near the initiation site for the barcode gene (Herbert \textit{et al.}, 2003b) with the categories for classification being standard and nonstandard initiator sequences.

In some cases a standard evolutionary computation system operating on SEMs provides an excellent solution to a sequence classification problem. In others the classification results are significantly better than chance but not as good as one would wish. In these latter cases this study applies a new technique called \textit{ring}
Ring optimization is a form of spatially structured evolutionary algorithm in which a small competent population, drawn from earlier evolution, is injected into a small portion of a ring-shaped geography. The population spreads around the ring with children migrating small distances from their parents. In earlier studies using ring optimization (Ashlock and von Koningslow, 2008; Ashlock  et al., 2008) it was shown that applying a ring species model for optimization yielded unexpectedly good results. These studies were not focused on optimization, rather they were modeling the biological phenomenon of ring species (Coyne and Orr, 2004; Irwin et al., 2001). In spite of this, the ring-species simulation yielded good results on the self avoiding walk (SAW) test problem and record-breaking results for the Tartarus virtual robotics task (Ashlock, 2006). This study applies the ring optimization to training SEMs for sequence classification. A discussion of why ring optimization might work well appears in Section 13.5.

A paper that applies SEMs to the problem of classifying long terminal repeats (LTRs) and short interspersed nuclear elements (SINEs) (Ashlock and Datta, 2012) examines the problem of extracting biological meaning from side effect machines. Only small side effect machines, with 4–6 states are evolved and aggregates features from many different side effect machines are used. This represents a different approach from the one taken in this study, in which biological data is classified using features from a single SEM with a larger number of states. One of the results in Ashlock and Datta (2012) is that it is possible to extract biological meaning from evolved side effect machines and several examples are presented.

2. The sequence classification problem

There are several different common types of sequence classification problems. This discussion outlines some of the common ones and reviews existing techniques. These techniques fall into three broad categories: motif finding, alignment and substring based techniques, and small substring statistics. The most computationally difficult of these three is motif finding. Alignment and substring matching are essentially a solved problem with tools like BLAST (Altschul et al., 1990) and suffix trees (Gusfield, 1997). Small substring techniques are the basis of many Markov-model and string kernel techniques (Zien et al., 2000).

A DNA motif is defined as a nucleic acid sequence pattern that has some biological significance such as being DNA binding sites for a regulatory protein, i.e., a transcription factor, and plays significant roles in gene expression and regulation (Lewin, 1997). Generally, the pattern is short (5–20 base-pairs long) and the locations of a motif may vary in different homologous sequences. Motif location can be extremely difficult, even computationally intractable, without an alternative computational approach. These approaches have included Gibbs sampling algorithm based approaches AlignACE (Roth et al., 1998), BioProspector (Liu et al., 2001) and Gibbs Motif sampler (Liu et al., 1995), expectation maximization based models (Bailey and Elkan, 1994), greedy approaches such as Consensus (Hertz and Storomo, 1999), statistical methods enhanced by position specificity (Peng et al., 2006), genetic algorithm based approaches such as Finding Motifs with a Genetic Algorithm (FMGA) (Liu et al., 2004) and Motif Discovery with a Genetic Algorithm (MDGA) (Che et al., 2005), and the Hidden Markov model Motif finder (HMMF) (Liu et al., 2009), and hybrid methods (Xia, 2012). A recent paper uses memetic algorithms to find motifs (Chan et al., 2012).

Gibbs sampling randomly chooses a possible motif of a fixed length from one sequence in the data. The algorithm then chooses another sequence and examines each substring of the same length. It then computes the probability that the selected substring is the motif. Then the algorithm randomly selects a substring from the sequence according to the distribution of the probabilities to replace the original motif. This algorithm will run until an acceptable local optimum has been found or until a specified time-out has been reached.

Consensus uses a greedy algorithm to align functionally related sequences and applies the algorithm to identify the binding sites for the E. coli CRP protein (Hertz and Storomo, 1999). Bailey and Elkan (Bailey and Elkan, 1994) develop software, MEME++, by using an Expectation Maximization technique to fit a two component mixture model to find motifs. Genetic algorithms have been applied to the motif finding problem such as FMGA. FMGA was found to have better performance than Gibbs Motif Sampler in both accuracy and computing time. MDGA is another program that uses genetic algorithms to find motifs in homologous sequences. It uses information content to evaluate individuals. MDGA achieves higher accuracy than Gibbs sampling algorithm based approaches with less computation time.

Hidden Markov Models have been in applied in combination with the Smith-Waterman local alignment algorithm (Smith and Waterman, 1981) in (Liu et al., 2009). It is proposed, and shown experimentally, that HMMF can achieve comparable accuracy with other tools such as Gibbs Sampler, BioProspector, and MDGA. The HMMF technique was comparable on sequences with a single binding site motif and its performance on those with multiple binding site motifs is significantly better. The small size of the biological data (less than 20 sequences) suggests that the technique needs to be tested for speed and efficacy on larger data sets.

Substring matching techniques include BLAST and its numerous derivatives and variations. We do not review these techniques in any detail because they are complementary to the system presented in this study. When sequence classification can be accomplished by large substring matching or alignment, an SEM-based system is probably not needed. An excellent book that contains many alignment algorithms as well as supporting computational techniques is Gusfield (1997). Small substring statistics can also be used to generate features that can then be classified with any of a variety of methods. Spectrum string kernel methods are examples of this approach (Filippone et al., 2008; Noble et al., 2002).

3. Side effect machines

A side effect machine is a finite state machine that has side effects associated with each state. This is a simple idea but there are two details that need to be filled in. What side effect will be associated with a state and, with choice of side effect in hand, how do you select a side effect machine that solves your problem effectively. In that study the side effect used is a counter, one per state, that determines how many times each state has been entered. This permits us to harvest a fixed length numerical vector describing a variable length DNA string. The mapping from DNA strings to count vectors is determined by the state transition diagram of the finite state machine. The problem of selecting the correct transition diagram is done with an evolutionary algorithm. The evolutionary algorithm is driven with a fitness function that checks classification accuracy on training data. Details are given in Section 5. We now look at earlier implementations of side effect machines.

Chaos Automata (Ashlock and Goldin, 2003) are an early example of side effect machines, different from those in this study. A chaos automata associates a similitude (an isometry of the Cartesian plane) with each state of a finite state machine. This permits the finite state machine to serve as an iterated function system (Barnsley, 1993) with memory and to generate effective fractal
visualizations of DNA strings. Chaos automata were evolved to generate fractals that yielded spatially disjoint attractors when driven with different types of DNA.

Another example of an application of finite state machines with side effects associated with their states appears in Ashlock et al. (2002). In this study finite state classifiers were trained to recognize two categories of PCR primers designed to amplify targets in Zea mays, those that amplified correctly and those that did not. Using finite state machines that classified primers based on a final recognizing state performed poorly. The finite state machines were modified so that every state carries a label (“good primer”, “bad primer”, or “don’t know”) and the sum of good states minus bad states encountered as a primer was run through the machine was used for classification. The side effect in this case is modification of two external state variables that counted the number of good or bad states.

3.1. Specification of the SEMs used

In this study counters on each state are used as the side effects associated with states. Before a DNA sequence is run through a SEM, all counters are set to zero. The DNA string is presented to the machine one base at a time, driving transitions, and each time the machine enters a state the associated count is increased. When the SEM is finished processing a DNA sequence, the vector of counts is normalized to obtain a unit vector indexed by the states of the SEM. This normalization is intended to remove the effects of string length on the numerical features. An example of a side effect machine appears in Fig. 1. The machine always starts in state 1 and so the first counter always gets one free count.

The primary utility of side effect machines is that they serve as data transducers, transforming variable-length string data into fixed-length numerical data. Once this has been done, any of a plethora of statistical or machine learning techniques may be applied easily to the transformed data. In this study, side effect machines are trained with an evolutionary algorithm. The fitness function checks the quality of classification generated by applying K-nearest neighbor classification (Knn classification), to the normalized count vectors generated by the SEMs. In addition to performing well on both synthetic and biological data within the study, side effect machines evolved to enable high performance by K-nearest neighbor techniques are likely to provide numerical features that are clean and useful to other clustering or classification algorithms.

Many machine learning techniques were originally designed for working with numerical data (Cherkassky and Muller, 1998) and work in classifying sequence data often deals with the variable length of sequence data by transforming it into numerical data. String kernels (Zien et al., 2000) are a standard method of doing this. The numerical features provided by side effect machines can serve as drop in replacements for string kernels and, unlike string kernels, they are able to use state-conditioned memory to select their features. This permits side effect machines to instantiate different sequence-based features in very different parts of a sequence. If, for example, a motif is significant only if it is downstream from a start codon, a finite state machine can first recognize the start codon and only then look for the motif.

4. Mathematical properties of side effect machines

This section presents a mathematical analysis of side effect machine properties and capabilities. We demonstrate that side effect machines can detect small substring statistics and motifs. It is also shown that the number of states a side effect machine requires to classify strings grows as the log of the length of the strings being classified.

One simple but biologically significant feature of sequence data is its GC-content, the fraction of bases that are either G or C.
islands or areas of the genome with clusters of genes have enhanced GC-content for example (Lewis, 2008). We begin with an illustrative lemma that shows side effect machines can efficiently detect this feature.

**Lemma 4.1.** If we wish to separate two classes of sequences with different GC-content then there is an optimal solution with two states.

**Proof.** Examine the SEM given in Fig. 2. State one counts the number of $G$ or $C$ bases, state two counts the number of $A$ or $T$ bases. The normalized counts provided by this machine place points into $\mathbb{R}^2$ so that their fraction of GC content is exactly their projection onto the line $y=x$ providing perfect linear separability of classes of sequences that can be separated by GC-content. \hfill $\square$

The feature “GC-content” is a very simple kind of small-substring statistic. The next result generalizes Lemma 4.1. The proof of this generalization is a reinterpretation of the construction of a well known combinatorial object.

**Definition 4.2.** If $A$ is an alphabet then a DeBruijn sequence with parameter $n$ is a circular string $D$ over $A$ so that every string in $A^n$ appears exactly once as a substring of $D$.

The following example is of a DeBruijn sequence of parameter $n = 3$ for the alphabet $A$. The directed graph is the core of the constructive existence proof that there are DeBruijn sequences for all alphabets and parameters.

**Definition 4.2.** On the left is a directed graph used, in a manner explained in the subsequent proof, to construct the circular DeBruijn sequence at the right. Notice that every length-three string appears once as three consecutive letters in the circular string.

**Theorem 4.3.** There is a DeBruijn sequence of parameter $n$ for any finite alphabet $A$.

**Proof.** Construct a directed graph $D(V, E)$ where $V = A^{n-1}$ and $V$ is all pairs of strings $(x, y)$ so that the $(n-2)$-suffix of $x$ is the $(n-2)$-prefix of $y$. **Example 4.2** gives this digraph for $A = \{0, 1\}$ and $n = 3$. This digraph is regular with the indegree and outdegree of each vertex being $|A|$. Note that by concatenating any two strings of length $n - 1$ and then looking at their successive substrings of length $n - 1$ we can construct a directed path from any vertex to any other, demonstrating that the digraph is connected. These two facts form the necessary and sufficient condition for $D$ to possess an Euler cycle (Wilson and vanLint, 2001). The numbering of the edges of the digraph in **Example 4.2** form an Euler cycle. If we start with the string that is first vertex of the Euler cycle and following it, adding the final character of the head of each arrow in the cycle, the resulting cycle is a DeBruijn sequence for $A$ of parameter $n$. This is because each edge corresponds, uniquely, to the substring consisting of its tail plus the last character of its head. Constructively this yields every string of length $n$, once each, as a substring. \hfill $\square$

The above proof is a well-known one and is included because the digraph constructed during the proof is needed for the following result.

**Definition 4.4.** Call the directed graph used in the proof of **Theorem 4.3** the DeBruijn digraph.

**Theorem 4.5.** A side effect machine exists that can count how many of each length $k$ substring are present in a string.

**Proof.** Finite state machines are directed graphs with labeled edges. In the DeBruijn digraph each edge is associated with the addition of one character to the DeBruijn sequence. If we label the edges with their associated characters, then pick a starting state, the DeBruijn digraph can be interpreted as a finite state machine. If we use this finite state machine as a side-effect machine each state, after a $k - 1$-character burn in period, is incremented if the preceding $k$ characters are a specific substring. This follows constructively from the way the DeBruijn digraph was constructed. This means that this side effect machine compiles length $k$ substring statistics. The length $k - 1$ burn-in period can be avoided by adding a tree-structured set of states that process the first $k - 1$ characters. \hfill $\square$

**Theorem 4.5** shows that if classes of strings can be separated by small-substring statistics then a side effect machine exists that can do the job. At the other extreme a **motif** is a single substring or collection of similar substrings that are associated with a class of strings. A useful tool for defining motifs are the IUPAC ambiguous base codes for DNA. The DNA alphabet uses the letters $C, G, A, T$ but a position in a motif might be occupied by $C$ or $G$ (or any non-empty subset of $\{C, G, A, T\}$). **Table 1** gives the standard one letter abbreviations for these fifteen nonempty subsets.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>IUPAC codes for ambiguous bases (or subsets of ${C,G,A,T}$).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Subset</td>
</tr>
<tr>
<td>A</td>
<td>${A}$</td>
</tr>
<tr>
<td>G</td>
<td>${G}$</td>
</tr>
<tr>
<td>R</td>
<td>${A,G}$</td>
</tr>
<tr>
<td>M</td>
<td>${A,C}$</td>
</tr>
<tr>
<td>W</td>
<td>${A,T}$</td>
</tr>
<tr>
<td>B</td>
<td>${C,G,T}$</td>
</tr>
<tr>
<td>H</td>
<td>${A,C,T}$</td>
</tr>
<tr>
<td>N</td>
<td>${A,G,C,T}$</td>
</tr>
</tbody>
</table>

**Definition 4.6.** A **motif** is a string over the alphabet of ambiguity codes given in **Table 1**. Notice that gaps or spaces in the motif may be encoded with the code $N$ that matches everything.

The following lemma is a direct implication of the famous theorem that says finite state machines can recognize regular languages (Gill, 1962). We therefore give a terse proof.

**Lemma 4.7.** A side effect machine exists that can count the number of occurrences of a motif in a sequence.

**Proof.** Create a sequence of states so that transitions into those states are labeled, successively, with the (possibly multiple) characters of the motif. This chain is constructed so that the first state is entered when one of the DNA bases that is a member of the first character in the motif is encountered. When this SEMs focus leaves this sequence of states, the transition back must loop to the correct point in the chain based on the longest prefix of the motif currently encountered. If this loop-back procedure requires separation of the
ambiguity characters into different groups then multiple states may be required at a given position in the chain of states. □

**Definition 4.8.** A number \( n \) of states in a side effect machine fully resolves the strings of length \( q \) over an alphabet of size \( m \) if the number of count vectors possible with the given number of states equals or exceeds the number of strings.

The goal of **Definition 4.8** is to create a surrogate for detecting when a side effect machine has too many states. If an SEM fully resolves a string space then it has more states than it needs to classify sets of strings within that space. As past experience (Ashlock and Warner, 2008a) has shown SEMs excel at over-training when given too many states.

**Theorem 4.9.** The number \( n \) of states required to fully resolve the strings of length \( q \) over an \( m \)-character alphabet grows in direct proportion to the string length.

**Proof.** Since a string has length \( q \) the count vector must total \( q \) and so the number of count vectors may be determined with the balls-in-bins formula and we see there are \( \binom{q + n - 1}{q} \) possible count vectors. There are \( m^n \) strings of length \( q \) over an \( m \)-member alphabet. Assume that \( m \geq 3 \) (a longer, more sophisticated proof will work for binary alphabets, but all applications in this paper use \( m = 4 \)).

\[
\binom{q + n - 1}{q} \leq m^n \tag{4.1}
\]

\[
\frac{(q + n - 1)!}{q!(n - 1)!} \leq m^n \tag{4.2}
\]

By summing a row of Pascal’s triangle we obtain:

\[
\frac{(q + n - 1)!}{q!(n - 1)!} \leq 2^{q+n-1} \leq m^n \tag{4.3}
\]

\[2^q 2^{n-1} \leq m^n \tag{4.4}
\]

\[2^{n-1} \leq \left( \frac{m}{2} \right)^q \tag{4.5}
\]

\[n - 1 \leq q \log_2 m - q \tag{4.6}
\]

\[n \leq q \left( \log_2 m - 1 \right) + 1 \tag{4.7}
\]

Yielding a linear upper bound on the number of states necessary to generate a unique count vector for any given string. We start again with inequality (4.1), reversed to give the lower bound.

\[
\binom{q + n - 1}{q} \geq m^n \tag{4.8}
\]

\[
\frac{(q + n - 1)!}{q!(n - 1)!} \geq m^n \tag{4.9}
\]

Using Stirling’s approximation,

\[
\ln n! = \sum_{j=1}^{n} \ln j, \text{Then} \sum_{j=1}^{n} \ln j \approx \int_{1}^{n} \ln x \, dx = n \ln n - n + 1 \tag{4.10}
\]

\[\frac{(q + n - 1)\ln(q + n - 1) - (q + n - 1) + 1 - (q \ln q - q + 1)}{-((n - 1)\ln(n - 1) - (n - 1) + 1) \geq q \ln m} \tag{4.11}
\]

\[\frac{(q + n - 1)\ln(q + n - 1) - (n - 1)\ln(n - 1) \geq q \ln(m) + 1}{q \ln(n) - q + 1} \tag{4.12}
\]

\[\text{Let} f(x) = x \ln x \text{ and } f'(x) = \ln x + 1 \tag{4.13}\]

Using the mean value theorem

\[q'(c) = f(q + n - 1) - f(n - 1) \tag{4.14}\]

\[q \ln(q + n - 1) + q \geq f(q + n - 1) - f(n - 1) \geq q \ln(m) + 1 \tag{4.15}\]

\[\ln(q + n - 1) \geq \ln(mq) + \frac{1}{q} - 1 \tag{4.16}\]

\[q + n - 1 \geq mq \frac{q-1}{q} \tag{4.17}\]

\[n \geq mq \frac{q-1}{q} - q + 1 \tag{4.18}\]

Yielding a linear lower bound on the number of states necessary, the theorem follows. □

**Theorem 4.9** can be interpreted as saying the number of states a side effect machine needs grows linearly as the length of the strings it is dealing with. The theorem also suggests that SEM states should be dealt out with an eye dropper, not a shovel. It does not give a rule-of-thumb for how many states should be used. It is probable that this must be determined experimentally.

### 5. Training side effect machines

In **Section 4** we showed that side effect machines can theoretically separate classes of sequences based on small substring statistics or based on the presence of motifs. The demonstrations were constructive, suggesting that side effect machines could be built to deal with these types of feature sets. These mathematical results are valuable because these feature sets are common separating principles for sequence data. On the other hand, if either of these well known types of features could do the entire job on their own there would be little use for side effect machines. Other, simpler techniques would suffice. The collection of features that a side effect machine can detect, given that a machine learning technique is to be applied to the feature vectors it produces, is combinatorially large. If we do not know a good deal about the relevant features in advance the problem of constructing a side effect machine to detect them is daunting. For this reason, we train side effect machines with evolutionary computation in this study. A key part of the training is a measure of the agreement of two partitions of a set of data.

**Definition 5.10.** Rand index Let \( D \) be a set of data and let \( P \) and \( Q \) be two partitions of this data into classes. The Rand index comparing \( P \) and \( Q \) is the fraction of pairs from \( D \) that are either in the same class in both \( P \) and \( Q \) or are in distinct classes in both \( P \) and \( Q \).

Side effect machines are trained with evolutionary algorithms. The training data are divided into exemplar sequences, evaluation sequences, and unused sequences. Algorithm performance is improved by rotating the training sequences among these three roles in a manner specified in the individual descriptions of experiments. The unused sequence category reduces the number of sequences used in a given fitness evaluation, improving speed. It is important to note that the separate crossvalidation sequences do not participate in this rotation. The algorithm for fitness evaluation of a side effect machine in this study is given in **Fig. 3**. To generate features for a sequence, it is run through the SEM and then the vector of counts from the states are normalized. This normalization means that the length of the sequence, which may vary, is not incorporated into the feature generated by the SEM. Fitness evaluation incorporates \( Knn \) classification. The fitness of an SEM is computed from the results of \( Knn \) classification applied to the normalized count vectors produced by a SEM. This computation uses the Rand index (Batagelj, 2006) to calculate the agreement of the known classification with that produced by \( Knn \) classification.
Input: A collection of $K$ classes of DNA sequences
A side effect machine $M$

Output: Rand index for classification quality

Details:
Run each exemplar sequence through $M$ to obtain example normalized count vectors
Run each DNA evaluation sequence through $M$ to obtain evaluation count vectors
$Knn$-classify the evaluation vectors using the exemplar vectors as the class examples.
Return the Rand index of $Knn$ classification vs known classification.

Fig. 3. Fitness function for side effect machines.

Fig. 4. An example of an arrangement of data in two categories that will confound $k$-means but not $Knn$ classification.

The Rand index, given in Definition 5.10, is a measure of similarity of two partitions of the same data set. The Rand index has a value of 1 when two classifications are in complete agreement. The minimum value depends in a complex fashion on the number of categories and relative size of individual classes. The expected score of a random partitioning rises with the number of classes of data. There is an adjusted Rand index (Hubert and Arabie, 1985) that corrects the Rand index to account for the effects of chance. This study only requires relative ranking of side effect machines for selection in evolution and so uses the simpler and more rapidly computed Rand index.

Earlier implementations of side effect machines (Ashlock and Warner, 2008c) used $k$-means, rather than $Knn$ classification, to assign the normalized data vectors to categories. This system was found to deal badly with classification problems similar to the one appearing in cartoon form in Fig. 4. Using a $Knn$-classifier instead of $K$-means improves results for many data sets, but it also introduces an additional complexity. The training examples must be divided into exemplars and sequences used for evaluation. Exemplars are sequences with known class that are used as nearest neighbors for $Knn$ classification while evaluation sequences are those that are to be classified by the SEM to evaluate its quality. To retard over-training, a danger that Theorem 4.9 suggests is a salient one for SEMs, as well as premature convergence, sequences in the training data are rotated between exemplars, evaluation sequences, and unused sequences during the course of evolution. Both exemplar and evaluation sequences are transformed into normalized count vectors by an SEM before they are used to evaluate its fitness.

In all experiments, other than those using ring optimization, a steady-state evolutionary algorithm incorporating size seven single tournament selection (Ashlock, 2006) is used. This form of tournament selection provides a moderate level of selection pressure and follows that used in Ashlock and Warner (2008c,a), Ashlock and McEachern (2009). All experiments use 5-nearest neighbor classification during the $Knn$ phase of fitness evaluation. Earlier studies suggest this parameter is not critical and so it was not varied. The synthetic data studies use 4 and 6 state machines on sequences with lengths uniformly distributed in the range 150–250. The results in Section 4 suggest that this is a generous allowance of states for this problem.

Previous studies suggest that machines with 12–30 states work well with biological data using sequence lengths of approximately 1000 DNA bases and demonstrate that over-training can happen with 36–48 states. For that reason the biological experiments use 24 state SEMs, with the exception of the most difficult of the biological classification problems. These were the nonstandard initiator data, Section 12, where a parameter study was conducted comparing 6, 12, and 24 state machines.

Both population size and the division of the data into exemplars and evaluation sequences were varied for the biological data. The values for these parameters are given in the individual experiments.

Evolutionary time in the experiments is measured in mating events. A mating event consists of randomly selecting a collection of seven SEMs. The two with the best Rand index are copied over the two with the worst Rand index. The copies undergo two-point crossover of the list of states defining the SEMs. A state contains its outward transitions, which specifies the effect of crossover on transitions. Each copy is then subjected to a number of mutations selected uniformly at random between one and the maximum number permitted in the experiment. This maximum is specified individually in the experiments. Mutation consists of changing the destination of a transition. Both the transition and its new destination are selected uniformly at random.

6. Ring optimization

This section describes a new type of spatial evolutionary algorithm called a ring optimizer. The ring is implemented as a population stored in an array where the first and last element of the array are considered to be adjacent, forming a ring. The elements of this array are called population slots or slots. A small initial
population of competent structures is placed in the ring and spreads in a manner described subsequently. We now review the biological system that inspired the ring optimizer to motivate its design.

6.1. Biological background on ring species

Ring optimization is inspired by an anomalous natural situation called a ring species. Biological ring species develop when an ancestral population expands around a geographic barrier and differentiates until terminal populations come back into contact. Adjacent populations are fertile—fertility declines with distance, and the terminal populations are mutually infertile. Ring species require a single founding population, gene flow between adjacent populations about the ring, a lack of geographic barriers, and evidence that the terminal populations (where the ring closes) are the most recent (Irwin et al., 2001; Coyne and Orr, 2004).

There are only a few examples of biological ring species. The Larus gulls have a circumpolar distribution and meet many of the criteria for ring species, but they have not dispersed far enough to close the ends of the ring (Liebers et al., 2004). If dispersal of this complex continues they may yet form a ring species. The salamander species Ensatina eschscholtzii encircles the Central Valley of California with an ancestral population in Oregon or Northern California and terminating with adjacent reproductively isolated sub-populations in Southern California (Stebbins, 1986). However, rather than one continuous population, it is currently believed that the salamander complex is composed of several subspecies based on morphological differences (Stebbins, 1986) and the potential for severe restrictions on gene flow may violate the ring species criteria. The presence of sub-population structuring suggests that the ring species has broken down. The most widely supported ring species is the greenish warbler complex Phylloscopus trochiloides in Asia (Irwin et al., 2001). Phylloscopus trochiloides is believed to have dispersed around the Tibetan Plateau from an ancestral population in the Himalayas terminating with cessation of gene flow in adjacent sub-populations in central Siberia (Irwin et al., 2001). The sub-populations of greenish warbler differ in plumage and male song with recognition of song, and consequently sexual attraction, decreasing with geographic distance (Irwin et al., 2001).

In Ashlock and von Konigslow (2008), it was shown that a simulation model of rings species had excellent optimization properties. This excellence neither depends on nor exploits the terminal infertility barrier. While optimization was not the goal of the study, the algorithm broke the previous record for the Tartarus task (Teller, 1994; Ashlock, 2006). This provides strong impetus to study the effects of ring optimization on other problems. In Ashlock and McEachern (2009) the ring simulation is developed into a specific ring-optimization model. Those results are expanded in Section 9 of this study.

6.2. Specification of the ring optimizer

Following the biological inspiration, the ring-structured population is seeded with a block of competent structures in adjacent slots within the ring. This founder population is chosen from a population evolved with a standard evolutionary algorithm. We do not study the impact of the size of the founding block in this study. Unreported results suggest this is a soft parameter. A group of ten organisms, forming the founder population, are chosen from best-of-run side effect machines taken from replicates of runs of a standard evolutionary algorithm. When more than ten replicates are available (always the case in this study) the founders are chosen uniformly at random with replacement from the available competent machines.

After injection of the founder population into the ring, the population spreads around the ring by executing mating events. A mating event within the ring optimizer is performed as follows. An occupied slot in the ring is chosen at random and its occupant becomes the parent. A slot is chosen in the ring; selected uniformly at random within the breeding radius R of the parent. If the slot is unoccupied then the algorithm examines slots successively closer to the parent until an occupied slot is found (it may be the parent itself). The occupant of this slot becomes the co-parent. The parent and co-parent breed, producing a single offspring via crossover and mutation. The fitness of the offspring is computed. A slot is then chosen within the migration radius M of the parent. If this slot is empty, the offspring occupies it. If the slot is occupied the offspring replaces the current occupant if the offspring’s fitness is no worse. The value of these parameters is fixed in this study, both having a value of 5. The algorithm is steady-state, performing mating events with the parent selected uniformly at random from the current population. Ring optimizations in this study are all run for 10,000 mating events.

Earlier studies found that the breeding and migration radius should be relatively small but that there is not a strong effect of the exact value (Ashlock and von Konigslow, 2008; Ashlock et al., 2008). Ring size is an important parameter and, for optimization, earlier studies suggest it is good if the ring is large enough that satisfactory optima are located before the ring closes. For this reason a relatively large ring size of 4000 was used for all the ring optimization experiments in this study. The number of mating events used, 10,000, is sufficient on average to roughly fill the ring. The fraction of the ring filled depends on the number of offspring that occupy empty slots as opposed to undergoing comparison with existing population members, which is random.

It is worth noting that one and two dimensional spatially structured evolutionary algorithms have long been used in evolutionary computation (Tomassini, 2005). These algorithms differ from the ring optimizer by initializing all available population slots. The large empty ranges in the early part of ring optimization permit any solution to survive and so permit extensive exploration by the algorithm which shifts to exploitation as the ring fills in. The impact of this on solution quality and robustness is discussed later in the paper.

7. Non-linear projection

This section describes a visualization technique used in all experiments. In Schonfeld and Ashlock (2006) a simple evolutionary technique for visualizing data, called Non-linear Projection (NLP) was presented. It is an evolutionary form of non-dimensional multi-metric scaling (Legendre and Legendre, 1998) or multidimensional scaling (Borg and Groenen, 1997). The goal is to provide a projection of points from a high-dimensional space into a two-dimensional space that distorts the inter-point distances as little as possible. The projection forms a visualization of the higher dimensional data set. The original form of NLP used evolutionary computation to minimize the squared error between the distance matrix of the original data and the Euclidean distance matrix of the the projection of the data into two dimensions. The representation consists of a simple list of point coordinates. The optimization problem of locating a good projection is treated as a standard real-valued evolutionary optimization.

An important topic within multidimensional scaling is the choice of function to minimize in order to get a useful projection. Since the technique is exploratory this choice is not quantitative but qualitative. In this study we maximize the Pearson correlation, Eq. 7.1, between the entries of the distance matrices for the original and projected data sets.

\[
\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)sxsy}
\]  

(7.1)
where for $z \in \{x, y\}$, $\bar{z}$ denotes the sample mean and $s_z$ denotes the sample variance. A problem with minimizing squared error between original and projected distances is that the optimizer must both get the inter point distances right and, in order to do so, must find the scale of the data. Estimating the scale of the data is possible but is an added and, as it turns out, unnecessary, complication. Maximizing the Pearson correlation, given in Eq. 7.1, of the original distances with distances of points in two space avoids this issue. Pearson correlation is invariant under translation and scaling of either of the data sets whose correlation coefficient is being given and so permits the evolutionary algorithm to solve the problem of relative distance without worrying about scale.

The evolutionary algorithm used to perform nonlinear projections uses a population of ten tentative projections stored as lists of points $(x, y)$. The points are initially generated to lie within the unit square with corners $(0, 0)$ and $(1, 1)$. The fitness of a collection of points is the Pearson correlation outlined above. The model of evolution is tournament selection of size seven. Variation operators are two point crossover of the lists of points (points are treated as atomic objects) and two mutation operators, each used 50% of the time. The first mutation operator randomly replaces a point with a new point selected uniformly at random within the unit square. The second adds a Gaussian random variable with a standard deviation of 0.1 to both coordinates of a point. From 1 to 3 mutations, with the number selected uniformly at random, are performed on any new structure. The algorithm is run for 100,000 or 400,000 mating events with the latter being used if the former still shows an upward trend in correlation.

8. Synthetic GC data experiments

The first synthetic data set used in this study is randomly generated DNA sequences, created to have two distinct classes. One of these has roughly 45% GC content while the other has roughly 55% GC content. Sequences of length 250 with exactly these fractions of GC bases are created, the bases are shuffled randomly, then a prefix of the shuffled sequence of length 150–250 with the length selected uniformly at random within the interval is used. This prefix selection means that the GC content has a binomial distribution with a mean equal to the expected fraction of GC content and the two classes can overlap at the boundary. Separate training and cross-validation data sets were created, each consisting of 500 sequences from each category of sequence.

A parameter study was performed on this synthetic data consisting of a full factorial design for populations sizes of 10, 100, and 1000, four or six states in the SEMs, and one, two, or three mutations per new side effect machine produced. In these experiments the thousand available training examples were divided randomly into 600 evaluation sequences, 200 exemplars, and 200 unused sequences. Which of the thousand available training sequences were used in each of the roles was changed every 40 mating events by making a new uniformly random selection. The experiments performed are summarized in Table 2 which also give the letters used to present the results of the parameter study in Fig. 5. The machine with the best training fitness in each run of the evolutionary algorithm is tested on the cross-validation data, dividing those 1000 sequences into 800 evaluation examples and 200 exemplars yielding the cross-validation fitness.

The highest cross-validation fitness for any machine produced in the parameter study is 0.968 achieved by the machine shown in Fig. 6. This machine is clearly evolved to solve the GC-classification problem. It has four available states, one of which is not used. State one counts the number of A or T bases while states two and

Table 2
List of values used in the parameter study performed for the synthetic GC data.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Population size</th>
<th>States</th>
<th>Mutations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>100</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>100</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>100</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>1000</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>1000</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>I</td>
<td>1000</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Letter</th>
<th>Population size</th>
<th>States</th>
<th>Mutations</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>10</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>K</td>
<td>10</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>10</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>100</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>100</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>O</td>
<td>1000</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>P</td>
<td>1000</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Q</td>
<td>1000</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>R</td>
<td>1000</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

![Fig. 5](image-url) Results of the parameter study for the synthetic GC data set. Shown are 95% confidence intervals on training (horizontal error bars) and cross-validation (vertical error bars) fitness.

![Fig. 6](image-url) The evolved side effect machine with the highest cross-validation fitness from the parameter study on synthetic GC data. State 3 was not used and so is not shown.
four count the G and C bases. The counts in states 2 and 4 are not even as any G/C that follows and A/T contributes a count to state two while G/C bases that follow G/C bases alternate between the two. This machine will provide a linearly separable collection of normalized count vectors in three-space. The lack of perfect cross-validation performance suggests that the inaccuracy in exact GC-content caused by subsequence selection when the synthetic data were collected yielded a small region of overlap between the two classes of data. Fig. 7 confirms that one sequence was in fact placed in the wrong category due to an low-probability event in the data generation.

The best cross-validation fitness occurred in experiment G with population size 1000, four states, and one mutation. Fig. 5 shows that, for both training and cross-validation, there are small but statistically significant advantages to using a large population. The experiments fall into three groups segregated by population size. The GC-problem is a very simple problem, recall that Lemma 4.1 shows that it has an optimal solution with two states. Each experiment performed in the parameter study had 100 replicates. This means that 1800 machines that were saved as being those with the best training fitness in some run of the evolutionary algorithm. Of these fifteen achieved perfect training fitness with cross-validation fitness varying from 0.883 to 0.951 for these machines. A total of 135 machines with less than perfect training fitness achieved a higher cross-validation fitness than this.

In order to demonstrate the impact of evolving side effect machines we used non-linear projection to compare the result of running 120 points of the GC-data set through a random side effect machine and a highly fit one. The resulting projections are shown in Fig. 7. The random machine has a slight bias toward separating the two classes while the evolved machine creates a clean division with a single exception. Notice the single black box intermingled with the white boxes. This box is an outlier of the type that we earlier speculated was caused by the randomized portion of the synthetic data preparation technique. This suggests that side effect machines can, when using nonlinear projection to visualize the results, be used to identify outliers within a data set.

9. Synthetic motif data experiments

The next set of experiments were performed on a set of synthetic data which fell into two categories: containing or failing to contain the motif (A|G)CACCCAT. This motif is a widely conserved DNA binding motif (Akiyama et al., 1996). This set of experiments is the first to use ring optimization. A parameter study was done on optimizing the division of the training data into exemplars, evaluators, and unused sequences when training side effect machines with a standard evolutionary algorithm prior to testing ring optimization for this classification task. The machines were trained on a synthetic data set of 1000 sequences, divided evenly between positive and negative examples, while a second similar collection of 1000 sequences was saved for cross-validation.

Following parameter setting studies for motif-location in Ashlock and Warner (2008c), populations of 10 SEMs with 24 states were used in all the experiments in this section. In a given run of the algorithm some number E of sequences were used as exemplars for k-nearest neighbor classification, another collection of T sequences was used to evaluate the side effect machine, while any remaining sequences were left unused.

The experiment was performed with a maximum of three mutations per new SEM. Evolution continued for 10,000 mating events with each experiment containing 100 replicates. The machine with the highest fitness of each run was set aside for use in ring optimization with the ten initial machines selected uniformly at random from these 100 elite machines. The ring optimization was also, after initialization, run for 10,000 mating events. A baseline study was performed by running the standard algorithm for 20,000 mating events to provide a comparison that used the same number of fitness evaluations as the ring optimization studies.

The replacement of k-means with k-nearest neighbor classification meant the division of the training data into exemplars and evaluation sequences is an unexplored parameter. Table 3 gives the divisions of the training data used for the baseline and ring optimization studies. These experiments form a parameter study for the parameters E and T. In all cases cross-validation was performed on the full 1000 member set of reserved cross-validation sequences.

| Table 3 Numbers of example E and evaluation T sequences used as a partition of the training data in baseline and ring optimization experiments. |
|---|---|---|---|
| Baseline | Ring | Baseline | Ring |
| E | T | E | T |
| 100 | 500 | 200 | 800 |
| 300 | 100 | 400 | 600 |
| 400 | 100 | 500 | 500 |
| 400 | 200 | 600 | 400 |
| 600 | 300 | 800 | 200 |
| 600 | 400 | – | – |
| 800 | 100 | – | – |
| 800 | 200 | – | – |

Fig. 7. Non-linear projection of 120 randomly selected point of the GC-content data from the training data set. The left panel shows the result of projecting the normalized count vectors of a random side effect machine while the right panel gives the same result for a highly fit evolved machine. The correlation coefficient for both projections exceeded 0.95.
The superiority of ring optimization found in this study is entirely in agreement with the earlier papers that used a ring structured algorithm. This study is the first to perform a direct comparison of the ring algorithm with an evolutionary algorithm permitted the same number of fitness evaluations. Past studies simply noticed that the ring-structured algorithms found record-breaking results. This study thus sharpens the result that ring optimization exhibits superior performance on some problems.

Comparison of the baseline and ring-optimization results in Fig. 8 highlights a startling result. In the baseline studies, parameter setting for the number of examples and evaluation sequences is important: there are substantial, statistically significant differences in performance based on the setting of these parameters. There was little difference and no significant difference between the ring-optimization runs. The ring-optimization algorithm thus demonstrates an unexpected robustness to this parameter choice. Since using fewer sequences does speed up the algorithm in direct proportion to the number of sequences not used, this may represent an additional advantage of ring optimization.

Fig. 9 shows a nonlinear projection of a normalized data vector processed with a random side effect machine and the machine evolved via ring optimization with the best cross-validation fitness. Notice that, unlike the analogous result for the GC-content data, that we do not obtain clean hyperplane separability in the projection. The points are distributed in 24-dimensional space in a manner that makes 5-nearest-neighbor classification highly effective. The machine used to produce the evolved projection in Fig. 9 had a fitness of 0.95. This is an example where Knn-classification works far better than k-means. The positive examples include the points that form the stripe of open boxes in the lower left corner of the evolved projection; this suggests that these points represent clear detections of the motif or some portion thereof.

Another feature of the ring optimization was its consistency of training accuracy. The variation in training and cross-validation accuracy was similar in the baseline runs and that level of variation also appeared in the cross-validation accuracy of the ring optimization runs. The training accuracy of the ring-optimization runs, however, was high and also showed low variation. The reason for this robustness to the division of the data into exemplars and evaluation sequences is not clear. As genes spread from the initial block of 10 machines through the empty regions of the ring almost any machine can survive. This means that the phase of the algorithm where the ring is filling is highly exploratory. In portions of the ring that have filled in, the algorithm moves to an exploitive phase because of its local elitism. In the baseline studies the trade-off between exploration and exploitation is managed by the E and T parameters. The algorithm is more exploratory when fewer sequences are used for evaluation. The exploration/exploitation trade-off remains fixed for the entire duration of the baseline studies while the ring-optimization makes a transition from highly explorative to highly exploitative as the ring fills in. We thus conjecture that the performance variation in the baseline studies results from choosing good or bad balances between exploration and exploitation; the ring-optimization algorithm manages this choice differently and apparently more effectively.

10. Immune sequence experiments

The human leukocyte antigen system (HLA) is the name of the major histocompatibility complex (MHC) in humans. The locus contains a large number of genes related to immune system function in humans. The IMGT/HLA Database is part of the international ImMunoGeneTics project and provides a specialist database for sequences of the human MHC and includes the official sequences...
for the WHO Nomenclature Committee For Factors of the HLA System. In addition to the physical sequences the database contains detailed information concerning the material from which the sequence was derived and data on the validation of the sequences (http://www.ebi.ac.uk/imgt/hla/).

The HLA data is divided into two classes. HLA class I antigens (A, B, C, D, E, F and G) present peptides from inside the cell (including viral peptides if present). These peptides are produced from digested proteins that are broken down in the proteasomes. The peptides are generally small polymers, about 9 amino acids in length. Foreign antigens attract killer T-cells (also called CD8 positive cells) that destroy cells. HLA class II antigens (DRA, DRB, DQA1, DQB1, DPB1, DPA1, DMA, DMB, DOA and DOB) present antigens from outside of the cell to T-lymphocytes. These particular antigens stimulate T-helper cells to multiply, and these T-helper cells then stimulate antibody-producing B-cells to produce antibodies to that specific antigen. MHC gene products are involved in the pathogenesis of many diseases, including autoimmune disorders. The exact mechanisms behind MHC associated risk of autoimmune diseases remain to be fully understood (Ma et al., 2009).

The available HLA data is substantial. A training and a cross-validation set, evenly divided between 1000 HLA class I and 1000 HLA class II sequences was selected randomly from the available data. The sequences have lengths from 300 to 800 bases with an average length slightly over 500. With parameters already set in earlier studies, a single initial set of runs consisting of 100 replicates of the basic algorithm were performed. The fitness function used 200 exemplars, 600 evaluation sequences and 200 unused sequences. This setting is slightly less than optimal and resulted from an initial incorrect analysis of the parameter study on motif sequences. The results suggest that there is nothing to be gained by re-running the experiments with the optimal settings. The sequence’s roles (exemplar, evaluation, unused) were randomized every 40 mating events. Side effect machines with 24 states were used, with a population size of 10, and a maximum of 3 mutations per SEM.

Out of 100 runs performed, 100 achieved perfect (1.0) training fitness, 91 achieved perfect cross-validation fitness, and the lowest cross-validation fitness was 0.935. These results leave nothing for the ring optimizer to do and so ring optimization was not attempted on the HLA data.

Fig. 10 shows the results of non-linear projection on the normalized count vectors for a random side effect machine and one that evolved to have perfect cross-validation fitness. The projection for the random machine strongly suggests that the problem is not difficult. One class of the data in the random-machine projection form a far more compact group than the other, though the groups are nested. The projection for the evolved side effect machine demonstrates excellent separation of the data.

11. HERV experiments

This section reports experiments on the classification of sequences from the human genome that are either derived from Human Endogenous Retroviruses (HERVs) or are human sequences not known to be derived from a retrovirus. Retroviruses are RNA viruses. They enter a human cell, reverse-transcribe their RNA into DNA which is then inserted into the host genome. The inserted DNA then (usually) directs the creation of more viruses. Sometimes the insertion does not generate new viruses, the insertion also can happen in germ-line cells, those that give rise to reproductive cells. In this case the DNA copy of the retrovirus becomes a heritable part of the human genome. Roughly 8% of the current human genome is thought to have arisen from retroviral sequences (Griffiths, 2001).

Two kinds of data were used for the HERV experiments: human endogenous retrovirus data and negative data. The HERV data comes from RetroSearch (www.retrosearch.dk) (Villesen et al., 2004). The negative data consists of sequences taken at random from the human genome that are not known to be genes or endogenous retroviruses. The endogenous retroviral regions that are excluded are taken from RepeatMasker (Smit et al., 1996–2004). The 356 HERVs in the HERV data set were chosen to have minimum length 5000, minimum open reading frame (ORF) length 100, at least 3 ORFs, and a minimum identity with known retroviruses of 0.90. Negative data sets for each experiment are chosen to have the same number of sequences as the positive set with the same distribution of lengths. Both sets of data were broken into blocks of length 2000 with 1000 base overlap between adjacent block. Two sets of 1000 blocks, divided evenly between HERV and NEG data were chosen for use as training and crossvalidation sets.

Initial experiments on HERV data used a population of 10 side effect machines with 24 states. New machines were subject to a maximum of 3 mutations of their state transition diagram with the number of mutations selected uniformly at random. Training data was divided randomly into exemplars, evaluation sequences, and unused sequences, with the role of sequences reassigned every 40 mating events. A sensitivity study similar to that on the motif data was performed to evaluate the best allocation into exemplars and evaluation sequences using the numbers given in Table 4. The algorithm was run for 10,000 mating events with the SEM with
best training fitness in each replicate subject to crossvalidation and saved for later analysis and seeding of ring optimization. Thirty replicates of ring optimization were performed using ten machines selected at random from previous non-ring optimization runs with $T = 800$, $E = 200$ used to seed the ring.

Fig. 11 shows that performance on the HERV data increases significantly when more sequence data is used for fitness evaluation. This mirrors the results on the Motif data for the standard evolutionary algorithm. The machines with the best crossvalidation scores, shown as Xs in Fig. 11 were substantial outliers; a few runs apparently discovered much more effective sets of sequence features than the others. The highest training fitness was 0.951 while the highest crossvalidation fitness was 0.872.

After the first parameter study to determine a good allocation of training data into exemplars and evaluation sequences a second parameter study was performed on population size. Population sizes of 10, 100, and 1000 were tested, otherwise leaving algorithm parameters the same as in the data allocation trials. The population members with the best training score in the final population were saved and 100 replicates were performed for each population size. The best-of-run genes were then used to initialize ring optimization with the same parameters that were used in the motif experiments. Thirty replicates were performed in each of the ring optimization experiments. Fig. 12 shows box plots for final crossvalidation fitness for the three experiments and the ring optimizations that they were used to initialize. These indicate that all advantages in Fig. 12 are statistically significant with $p < 0.05$ except the improvement of the population size 100 run by their corresponding ring optimization.

The ring optimizations in Fig. 12 do not use different populations sizes, rather they are initialized with blocks of machines from populations evolved at different sizes. In all three cases, the ring optimization improved the results. The worse the founder
population was, the more the ring optimization improved the results, but the quality ordering of the populations was retained in the ring optimizations. This result is similar to that reported in Fig. 11 in which the ring is less sensitive to an algorithmic parameter, though this is a much weaker demonstration of the effect.

The best result was a classifier that achieved a Rand index of 0.901. This classifier was the top outlier for population size 100, not one located in the ring optimization. The shifting of the roles of sequence data between exemplars and evaluation sequences means that the algorithm, while technically elitist, can lose a classifier with a superior crossvalidation score. The reassignment of roles among the sequences in the training data, in effect, mean that we are changing the fitness function every forty mating events.

Fig. 13 shows a nonlinear projection of the normalized feature vectors for 200 randomly sampled points of training data. Dark squares correspond to positive examples and open squares correspond to negative examples. The random projection is separating the classes considerably better than randomly: while the best evolved SEM does a better job the random SEM does a remarkably good one on the HERV data. This observation suggests that checking the fitness trajectory over evolution for the SEMs being trained to classify HERV data would be a good idea. Fig. 14 displays this data.

The mean fitness plot reveals something very interesting: the average behavior in a population of random side effect machines is a classification of HERV sequences considerably better than random chance. This in turn indicates that SEMs are well suited to this particular classification task. The task itself is not trivial, but random side effect machines half-solve it.

12. Nonstandard initiator experiments

This experiment deals with variant genetic codes in invertebrate mitochondrial DNA. The mitochondrial gene cox1, encoding the protein Cytochrome c Oxidase subunit 1 (Cox1), is fundamental in modern biodiversity studies. This gene contains the DNA barcode region for animals, a stretch of DNA that is highly diagnostic of species. The DNA barcoding effort represents an attempt to sequence the same region of DNA, 650 base pairs from cox1, for every animal species on the planet (Herbert et al., 2003b; Hebert et al., 2003a). Sequencing even a partial gene from every species is an enormously challenging task, not just collecting specimens, but also developing primers for amplifying the barcode gene which are effective on diverse species. Understanding the genetics, evolution, and molecular biology of cox1 is key to achieving this goal.

RNA is translated into protein residues or amino acids in blocks of three bases, called codons. The translation of three base pairs of RNA to an amino acid is accomplished through the use of mediator molecules called transfer RNAs (tRNAs). Each tRNA contains a three base pair anti-codon which binds to the RNA and a 3′ terminal site which binds to a protein residue. Since there are 20 protein residues and 64 codons, there is a many-to-one mapping from codons to amino acids. The standard start codon is AUG, and it codes for the amino acid Methionine. Several other alternative start codons have been located via experimental analysis, most of these are variations of AUG such as AUA, and AUC.

The genetic code for translating RNA triples into codons has a most common form often referred to as “the” genetic code. A few organisms use different mappings or translation tables for transforming codons into protein residues. The invertebrate mitochondrial genes, for example, recognize AGA as coding for Serine, while the standard code recognizes AGA as coding for Arginine. In addition to changing codon to protein residue mappings, as taxonomic groups change so does alternative start codon usage. Invertebrate mitochondrial genes recognize five alternative start

![Fig. 13](image1.png)  
Fig. 13. Non-linear projection of 200 points selected at random from the training data for the HERV experiments. The left panel shows the result of projecting the normalized count vectors of a random side effect machine while the right panel gives the same result for machine that achieved cross-validation accuracy of 0.90 (the best found). The correlation coefficient for the random SEM projection is 0.93 while it is 0.89 for the evolved SEM projection.

![Fig. 14](image2.png)  
Fig. 14. Displayed above are 95% confidence intervals, sampled every 200 mating events, for the population size 100 runs of the standard evolutionary algorithm of the mean fitness. This mean is the mean of mean population fitness over all 100 replicates in this experiment. The mean fitness trajectories for the replicates are displayed in random shades of light gray in the background for reference.
codons (in addition to the standard AUG): UUG, AUU, AUC, AUA, and GUG. With the sole exception of UA, each of the alternative codons is only a single base pair mutation away from the standard, AUG.

12.1. Initiation of translation in cox1

Early sequencing of genes for the flies Drosophila melanogaster and Drosophila yakuba reported something unusual; cox1 did not appear to have an in frame start codon (Clary and Wolstenhome, 1983; deBrujin, 1983). There was, however, a start codon (AUAA) base pair out of frame. Further work in sequencing Drosophila and other insects found more unusual instances of cox1 (Ballard, 2000; Lunt et al., 1996).

Three distinct hypotheses were proposed to explain this phenomenon:

(i) initiation of translation began with a specific tRNA which had a 4bp anti-codon instead of the traditional 3bp anti-codon,
(ii) the ribosome frame shifts after recognizing the start codon, or
(iii) the mRNA is edited after transcription in order to bring the start codon into alignment.

There has, however, been no direct experimental evidence to support any of the three hypotheses proposed above. A fourth hypothesis was proposed in 1993 by Beard et al. (1993):

(iv) cox1 utilizes a sixth alternative start codon, UGC.

Krzywinski et al. provided some support for this hypothesis by showing that the mRNA transcript of cox1 for Anopheles funestus began with the codon UGC (Krzywinski et al., 2006). However, UGC is not present in all instances of cox1 with an unknown start site. If the fourth hypothesis is true then there would need to be several more unrecognized start codons in addition to UGC.

The issue of initiation in cox1 remains unresolved (Szafranski, 2009). This creates a great deal of ambiguity in the annotation of cox1 in genomics databases. Often researchers will either propose a new 4bp recognizing tRNA (Satta et al., 1987), a new start codon (Beard et al., 1993), or simply fail to annotate the start of the cox1 gene. As more and more mitochondrial genomes are sequenced we now have the opportunity to explore this irregularity more fully.

The experiments in this section address one primary question: are there patterns in the sequence data surrounding the putative initiation sites we can use to distinguish between sequences with unknown start codons and sequences with known start codons?

12.2. Nonstandard initiator data

The primary data set for the experiments reported in this section consisted of 275 full mitochondrial genomes from the Phylum Arthropoda.

A mitochondrial genome typically encodes 13 genes: NADH dehydrogenase 1, NADH dehydrogenase 2, NADH dehydrogenase 3, NADH dehydrogenase 4, NADH dehydrogenase 4L, NADH dehydrogenase 5, NADH dehydrogenase 6, Cytochrome B, Cytochrome c Oxidase 1, Cytochrome c Oxidase 2, Cytochrome c Oxidase 3, ATP synthase 6, and ATP synthase 8.

The full mitochondrial genomes were obtained from Genbank. Only genomes which were part of the (Pruitt et al., 2007) project were used. If two or more genomes were present from the same species, a single representative was selected at random.

The number of nonstandard start codons for each gene is given in Table 5.

<table>
<thead>
<tr>
<th>Gene</th>
<th>No. w/ start codons</th>
<th>No. w/o start codons</th>
</tr>
</thead>
<tbody>
<tr>
<td>NADH dehydrogenase 1</td>
<td>272</td>
<td>3</td>
</tr>
<tr>
<td>NADH dehydrogenase 2</td>
<td>270</td>
<td>5</td>
</tr>
<tr>
<td>NADH dehydrogenase 3</td>
<td>275</td>
<td>0</td>
</tr>
<tr>
<td>NADH dehydrogenase 4</td>
<td>273</td>
<td>2</td>
</tr>
<tr>
<td>NADH dehydrogenase 4L</td>
<td>274</td>
<td>1</td>
</tr>
<tr>
<td>NADH dehydrogenase 5</td>
<td>275</td>
<td>0</td>
</tr>
<tr>
<td>NADH dehydrogenase 6</td>
<td>273</td>
<td>2</td>
</tr>
<tr>
<td>Cytochrome B</td>
<td>275</td>
<td>0</td>
</tr>
<tr>
<td>Cytochrome c oxidase 1</td>
<td>164</td>
<td>111</td>
</tr>
<tr>
<td>Cytochrome c oxidase 2</td>
<td>272</td>
<td>3</td>
</tr>
<tr>
<td>Cytochrome c oxidase 3</td>
<td>272</td>
<td>3</td>
</tr>
<tr>
<td>ATP synthase 8</td>
<td>275</td>
<td>0</td>
</tr>
<tr>
<td>ATP synthase 6</td>
<td>275</td>
<td>0</td>
</tr>
</tbody>
</table>

ACC, ACG, CAA, CCG, CGA, CUA, CUG, CUU, GAA, GCG, GUU, UAU, UCA, UCG, UUA, and UUU. Start codons are often inconsistently reported as either Methionine or the traditional translation product of the 111 sequences containing non-standard start codons 18 were reported as beginning with the amino acid Methionine in the provided translation in Genbank.

In instances where a standard start codon was not present the site of initiation was usually determined through comparison with cox1 sequences from other species.

From the 275 full genomes in the primary data set a subset of 225 were selected to be used in the experiments, discarding sequences with large fractions of unknown bases. From these 225 full cox1 sequences two sets of sequence data were extracted, sequences of 134 bases upstream of the initiator and sequences of 134 bases downstream from the initiator. The former are non-coding while the latter are a coding portion of the cox1 gene. The 225 sequences were split into 165 training and 60 cross-validation sequences. Duplicate sets of experiments were performed for the upstream and downstream sequences. These data are collectively referred to as the nonstandard initiator (NSI) data sets.

The reported initiation codon was excluded from the data so that the machines did not have the option of simply recognizing that codon. The difficulty of doing this is not clear (the codon lies deep within the sequence) but our goal is to classify cox1 sequences that use nonstandard codons when those codons are not yet identified or are of an unknown type.

12.3. Experimental design for NSI data

A parameter study was performed for two factors that were identified in earlier experiments as being problem dependent: number of states and population size. In this study each set of data is used in a parameter study for each possible pair of population size 10, 100, or 1000 and 6, 12, or 24 states in the SEMs. The results of this study are shown in Fig. 15 and Table 6. The boxplots in Fig. 15 show that the problem is very difficult with only a few outliers performing well. After constructing this figure a second statistic was extracted to quantify performance: the number of classifiers that achieved cross-validation fitness with a Rand index better than random chance with p = 0.05. Table 6 also gives the Rand index of the best classifier found in each experiment. These results show that it is far easier to achieve classification with the downstream data than the upstream data and that the problem is a very hard one.

It is interesting to note that the best classifiers from different experiments within the parameter study often have the same Rand index. The grid-like arrangement of the outliers in Fig. 15 demonstrate this. It is also the case that many of the outlier impulses in Fig. 15 correspond to multiple best-of-run automata. Not apparent in Table 6 is that the numbers shown as agreeing to two decimals
actually agree to five or more decimals. This strongly suggests that
the evolutionary computation system is discovering a relatively
small number of partitions of the data that yield good crossvali-
dation fitness. A given classification scheme that can be realized
as a side effect machine can be implemented by a combinatorial
space of such machines by simply changing state labels and manip-
ulating the structure of unused states or transitions. The fitness
data suggest that only five fundamental solutions were located
in these experiments. Of these, four exhibit classification perfor-
ma nce that is better than random. These classify are from among
those located by the 900 replicates of the evolutionary algorithm
summarized in Fig. 15 and Table 6.

If we pool the data across the six, twelve, and twenty-four state
experiments and apply a normal approximation to the binomial
distribution to estimate the probability of finding a better-than-
random classifier then the performance of 100 member populations
(22/300) outperform 10 member populations (17/300) which in
turn outperform 1000 member populations (10/300). These dif-
f er ences are all significant at the \( p = 0.05 \) level. This indicates
that for the NSI data the intermediate population size is supe-
rior.

Fig. 16 gives the results of nonlinear projection of normalized
count vectors from a random 12-state SEM and the SEM (also with
12 states) that got the best crossvalidation fitness on the down-
stream data. The SEM used to create the right-hand panel of Fig. 16
got a crossvalidation fitness of 0.8. This makes the NSI data set
the poster child for using \( K_{nn} \) rather than \( K \)-means for computing fitness during training of SEMs.

Ring optimization was tried on the NSI data set, both by using
ten randomly selected best-of-run machines and by assembling
a premium initializer set consisting of all machines with a given
number of states that achieved better-than-chance crossvalidation.
Both methods of ring optimization yielded results that had no bet-
ter crossvalidation fitness than the original evolutionary algorithm.
It is interesting to note that when the result of the ring optimiza-
tion possessed better-than-chance crossvalidation, its Rand index
agreed to five decimals with the Rand index of one of its proge-
nitors. The variation operators are capable of rearranging unused
transitions in a side effect machine while preserving essential func-
tionality. The lack of any improvement in fitness strongly suggests
that the good results of ring optimization on the NSI data were
repackaged versions of the solutions present in the initializing
populations.

13. Conclusions and discussion

Ring optimization was tested on two synthetic data sets, one
where the classification principle was GC-content and the other
when a motif was present or absent in two data categories. Three
biological data sets were used. The first asked SEMs to sepa-
rate two categories of immune system sequences, the second
asked them to distinguish human genome sequence derived from
retroviruses from sequence not known to be of retroviral or-
gin, the third consisted of portions of the mitochondrial genomes
near an initiation codon that was either standard or nonstan-
dard.

The standard evolutionary algorithm essentially solved the
GC-content and immune-system classification problems. Ring opti-
mization was applied to the other three problems. Use of ring
optimization yielded substantial improvement for the synthetic
motif data set, substantial improvement of average performance
for the retroviral data, and no improvement for the nonstan-
dard initiator data. Given its earlier stellar performance on the
Tartarus problem this suggests that ring optimization is a good
option to consider but that its results should be compared to
those of a standard algorithm rather than used automatic-
ly.

13.1. Fitness: \( K_{nn} \) vs \( K \)-means

Figs. 7, 9, 10, 13 and 16 have the same form, contrasting two
nonlinear projections. One is of a sampling of the normalized count
vectors from a random side effect machine on the training data,
the other is similar but uses the side effect machine with the best
available crossvalidation score. In this study, side effect machines
are trained using a \( K_{nn} \) classifier rather than \( K \)-means clustering;
\( K \)-means was used in the first wave of side effect machine studies.
For \( K \)-means to be part of an effective training system for SEMs the
normalized training vectors for the two data categories would need
to form compact sets. We can use the nonlinear projections in these
figures to place an informal lower bound on how well \( K \)-means
might have done.
- **GC-content data**
  With the exception of a single odd sequence the evolved side effect machine yielded a projection that exhibits hyperplane separability. It is likely that K-means would have performed well in this case and related experiments in Ashlock and Warner (2008a) verify that K-means works well on this type of data.

- **Motif data**
  The data exhibit a region that is mostly of one type and another region that is mostly of the other. K-means is likely to perform better-than-chance but not as well as KNN on these data.

- **HLA data**
  The evolved side effect machine yielded a projection that exhibits hyperplane separability. It is likely that K-means would have performed well on the HLA data.

- **HERV data**
  These data look similar to the motif data with the projection of the two types failing to form compact groups. Nevertheless there are large compact groups of one type or other other suggesting K-means would have turned in an intermediate level of performance on this data.

- **NSI data**
  The projections produced yield no hope that K-means could have solved this problem, but KNN only made a moderate dent in this problem.

It is important to remember that the projections used to estimate the probable success of K-means were evolved to function with KNN classification. This means that a K-means friendly projection is good evidence that K-means would work well while a K-means hostile projection says little. Nevertheless the projection suggest that using KNN is a good choice in spite of the added complexity of partitioning the training data into exemplars and evaluation data.

13.2. Parameter setting

Earlier studies identified number of states and population size as important parameters with population size being a critical one. In this study it was demonstrated that both of these parameters are specific to the data set. The GC-data, for example, required a small number of states and performed slightly better with four states than six. The best machine evolved for the GC-data, nominally possessing four states, used only three. In the one biological example where the number of states was varied, the NSI data set, twelve states were found to be marginally better than six or twenty-four.

For the number of states parameter the actual number of states is a relatively soft choice as long as (i) the number of states is not so small as to prevent solution but (ii) not so large that over-training occurs.

Population size was examined for four of the five data sets (for the Motif data this was done in Ashlock and Warner (2008c), not in this study). The GC-content data experiments found population size 1000 yielded the best results by a small but significant margin. For the Motif data experiments population size 10 had a large, statistically significant advantage. In the HLA data population size was not examined as a parameter but population size ten gave excellent performance. For both the HERV and NSI data, representing a difficult and a very difficult biological data set, population size 100 yielded superior performance. This performance advantage of size 100 populations was statistically significant for both the HERV and NSI data. This indicates that “Classifying Sequence Data with Side Effect Machines” is not a single type of evolutionary optimization. Rather the character of the sequence data determines the character of the adaptive landscape with very different adaptive landscapes present for different data.

In the original SEM studies using K-means in the fitness evaluation of SEMs every type of data exhibited a strong small population effect (Ashlock, 2006). A small population effect is present when very small populations, roughly of size \( n \leq 12 \), exhibit significantly superior performance on an evolutionary computation task. The 3-parity task using a parse trees representation is another example of a system that exhibits a small population effect as is the non-linear projection software used in this study (data not shown). When fitness evaluation was shifted to a KNN-based system then the small population effects became far less common; only one of the four data sets in this study for which populations sizes were compared exhibits a small population effect.

13.3. A failure of normalization

The authors thank Wendy Ashlock for the example shown in Fig. 17 that highlights a potential problem with SEMs that speaks to preparation of data. After running a sequence through a side effect machine, we obtain a count vector, which is then normalized to remove the influence of sequence length. If we run a long sequence of \( n \) DNA bases through the machine shown in Fig. 17 then the normalized count vector will be very close to

\[
\begin{pmatrix}
0.1 \\
0.1 \\
0.1 \\
0.1
\end{pmatrix}
\]

![Fig. 16. Non-linear projection of all 165 members of the downstream NSI sequence data from the training data set. The left panel shows the result of projecting the normalized count vectors of a random side effect machine while the right panel gives the same result for machine that achieved cross-validation accuracy of 0.80 (the best found). The correlation coefficient for the random projections is 0.93 while it is 0.96 for the evolved SEM.](image-url)
This machine arose during initial testing of an SEM-based system in which the positive examples had lengths from 10,000 to 12,000 bases and the negative examples all had lengths of exactly 8000 bases. The machine achieved perfect accuracy based entirely on the first coordinate of the normalized feature vector which contained the reciprocal of the sequence's length. The single count that the first state is granted permitted the system to transform the normalization, intended to remove length information, into a perfect length detector.

13.4. Feature selection versus feature induction

A standard task in machine learning is that of feature selection, picking which of a set of available features to use. When training a side effect machine with \( n \) states we are, in effect, selecting a collection of \( n \) numerical features, the entries of the normalized count vector, one per state. These features have some nonstandard properties. The fitness function used to evolve side effect machines considered these features in aggregate and in fact only in aggregate. The meaning, or worth, of the individual numbers in a normalized count vector is untested. In addition there are \( n^m \) side effect machines on \( n \) states. The space of 24-state side effect machines for DNA thus contains roughly \( 3 \times 10^{12} \) members. Given that some states are not used in many machines and that relabeling the states will yield \( n! \) equivalent forms for any given side effect machine the number of machines exhibiting different behavior is probably much smaller - but still quite large.

The absurd size of the search space of side effect machines suggests that the evolutionary search, on those search problems that prove experimentally to be tractable, is incremental. It has the potential to recognize a useful feature for classifying sequence and then adding others subsequently. Subsequent sequence features may not be from the collection of all useful features, rather these features will be chosen from those that work well with those already selected. This means that side effect machines will tend to locate sets of numerical features that work well together. It also seems likely that some of the entries of the normalized count vector do not contain useful information. Rather they correspond to structural components within the side effect machine that enable the computation of other features.

Suppose, for example, that a sequence classification problem is partially solved by noticing many of the negative examples have relatively low GC-content. A population of machines evolves that classifies based on GC content. Not all of the high GC-content sequences, however, are positive examples. The positive examples also contain a motif that is also present in some of the low GC-content sequences. Having evolved finite state structures for the relatively easy GC-content problem, evolving a sequence of states that recognize the motif only when it appears in a high GC-content sequence is probably within the capabilities of a side effect machine.

Given the size of the feature space searched when training a side effect machine with even a moderate number of states it may be more reasonable to speak of side effect machines as performing feature induction, the creation of new made-to-order features, rather than feature selection.

13.5. The behavior of the ring optimizer

In the first two ring-algorithm studies, those attempting simulation of biological ring species rather than optimization, it became apparent that the ring optimizer was remarkable good at solving the Tartarus virtual robotics task. Explorations connected to the Tartarus fitness landscape may be found in Ashlock et al. (2004), Ashlock and Warner (2008b). While the fitness landscape of this problem is not well understood, it is clear that there are a huge number of peaks in the adaptive landscape; an outline of a mathematical proof that an optimal solution exists suggests all optima thus far located are not global optima. The ring optimizer found 50 distinct solutions that were better than the then-record solution. This result motivated the exploration of ring optimization as a potentially powerful new type of spatially structured evolutionary algorithm.

For the three data sets on which ring optimization was attempted in this study it was found to be highly effective on the motif data, modestly but significantly helpful on the HERV data, and no better than a standard algorithm on the NSI data. The data on which population sizes are more effective for the standard algorithm on these data suggest that the associated SEM optimization problems for these data have different fitness landscapes. This, in turn, suggests that ring optimization is effective on a particular type of adaptive landscape. We now engage in theoretic speculation to attempt to illuminate what sort of landscape this might be.

In any evolutionary algorithm there is a tradeoff between exploration (searching for new hills in the adaptive landscape) and exploitation (climbing the hill or hills already located). Consideration of the way the ring optimizer behaved on the motif data suggests that the ring optimizer manages this tradeoff automatically with the ring size controlling the character of the tradeoff. The ring optimizer starts by initializing a small block with competent structures. These structures then spread through the ring. In relatively empty parts of the ring, new structures often find empty slots and so are placed in the population without examination of their fitness. In portions of the ring where the population is denser, survival requires comparison with a previously existing structure. This means that the relatively empty regions of the ring are engaging in exploration while the more densely populated portions are engaging in exploitation. The population members in empty parts of the ring are given great latitude to blunder over the landscape to new hills of the adaptive landscape; the larger the ring the longer exploration continues. A very large ring corresponds to an algorithm that can explore for new hills for a long time. In the portions of the ring that fill in, exploitation commences. Variations on the locally dominant type compete to make incremental improvement to that type. This means that, other than a very short initial phase, both exploration and exploitation are taking place in different parts of the ring with the balance shifting toward exploitation as times passes.

When parameters, such as mutation rate, population size, or the fraction of training data used as exemplars, are chosen in an evolutionary algorithm one of the results of of these choices is control of the relative amount of exploration and exploitation performed by the algorithm. In the experiments with the motif data the ring optimizer demonstrated remarkable robustness to parameter choice for fraction of training data used as exemplars. On the HERV data a
weaker but significant robustness to population size was observed. This suggests that, to the degree exploration/exploitation trade-offs are determined by parameter setting, the ring optimizer does not require such parameter setting. Simply using a very large ring ensures both types of search will be used extensively.

In Ashlock et al. (2008) the technique of hybridization of competent structures from multiple distinct evolutionary runs was studied. This technique was found to be helpful for a number of different evolutionary computation problems. If different portions of a ring have different dominant local types, hybridization between those types continues at boundaries between those types within the ring. This potentially helpful effect is added to the smooth transition from exploration to hybridization as the ring fills in to explain why a ring optimizer might work well. This also gives at least a plausible structure for the type of adaptive landscapes that a ring optimizer can do well on. A landscape on which a ring optimizer is effective is one with many adaptive hills that are not too far apart, as measured by the likelihood of the variation operators permitting travel from one hill to the other in a relatively small number of steps. This is consistent with the identification of a very small number of better-than-random optima in the landscape connected with the NSI data and the unimpressive performance of the ring optimizer for those data.

14. Next steps

The numerical features generated by a side effect machine, in the form of normalized count vectors, are used only with 5-nearest neighbor classification in this study. Once the features have been induced by an evolutionary algorithm there is substantial potential to use them with a more sophisticated technology such as support vector machines. In addition, as noted previously, it may be that subsets of the entries of the normalized feature vectors may contain the useful information with other parts of the count vector corresponding to states in the SEM that serve a structural purpose. Features corresponding to unused states are uniformly zero and so contain no information. A good topic for future work would be the application of feature selection techniques to the entries of the normalized feature vectors.

This study is the first to compare the behavior of evolutionary optimization of SEMs for a broad range of different types of sequence data. The results suggest that different types of sequence data yield substantially different evolutionary optimization problems. A theoretical basis for understanding what qualities of a sequence classification problem creates these different types of optimization problems is an early priority for additional research. In particular such a theory might give a good indication when ring optimization might be a useful follow-on to a standard optimization. This would involve testing side effect machines on many additional types of sequence data.

The robustness of ring optimization against parameter choice should be studied in much greater detail. Robustness against choice of mutation rate, for example, is an unstudied area. Early studies established that for a few relatively difficult problems that larger rings are more effective and that the breeding and migration radius are soft parameters. In the current study the ring was typically mostly filled with a small sparsely occupied portion remaining. A parameter study of the impact of ring size and of the behavior of the system in densely populated and sparsely populated regions may help illuminate how to use ring optimization efficiently.

Acknowledgments

The authors thank Wendy Ashlock for assembling the HERV data set and for pointing out the existence of the SEM given in Fig. 17.

References


