Iterated Function System Fractals for the Detection and Display of DNA Reading Frame

Dan Ashlock
Mathematics Department and Bioinformatics and Computational Biology
Iowa State University
Ames, IA 50011
danwell@iastate.edu

James B. Golden III
Computational Biology Coordinator
Pioneer Hi-Bred International, Inc.
Bioinformatics Department, PP Box 552
Johnston, Iowa, 50131-0552
goldenjame@phibred.com

Abstract
We report a technique for using an evolutionary algorithm to select the parameters for a data-driven iterated function system. Such iterated function systems are typically driven with uniform random numbers to produce fractals. We instead drive the iterated function system with a biased source mimicking DNA with and without stop codons. An evolutionary algorithm is used to produce fractals that visually display the reading frame DNA. We perform a second set of experiments using the whole genome of mycobacterium tuberculosis in two different reading frames. The fractals located with our evolutionary algorithm correctly separate the DNA into in-frame and out-of-frame for the simulated data and the mycobacterium DNA. The fractals do not give dramatic visual cues to the differences for the mycobacterium data unless points associated with different members of the iterated function system are shaded. Close examination of the fractals yields insight into DNA structure.

1 Introduction
This paper connects three disparate threads in the computational interpretation of data. The first, chaos game fractals and iterated function systems, seek to give useful visual displays of large, possibly complex data sets. The second, bioinformatics, has as one of its goals the automation of analysis and interpretation of large biological data sets. The third, evolutionary computation, is used to connect the first two. We use evolutionary computation to produce data driven iterated function systems whose fractal attractors form a visual presentation of information about the sequences of DNA used in place of the random numbers that usually drive such processes.

1.1 Chaos Game Fractals
A chaos game is characterized as the process of generating a fractal by accumulating the positions of a moving point. This moving point is repeatedly displaced toward one of a fixed set of points, e.g. the vertices of an equilateral triangle. In Figure 1 we see the Serpinski triangle, a chaos game in which a moving point is displaced half way from its present position toward randomly selected vertices of a triangle in each iteration. Figure 1 is plotted over 100,000 iterations of the process.

The character of the fractal resulting from a chaos game is controlled by the number of fixed points being used and the order in which those points are used to specify the direction of motion for the moving point. This latter point is key. The Serpinski triangle is generated by using the apices of the triangle, uniformly at random, in succeeding iterations. If, instead, we have data with some degree of non-randomness then the points in the resulting fractal are a subset of the fractal obtained by driving with random data. At this point it is worth not-
Figure 2: The diagram in the upper half of this figure shows how sequence data subdivides the square in a four cornered chaos game. Such a chaos game, driven by HIV sequence data, is displayed in the lower half of the figure. The gaps in the fractal indicate the lack of methylation sites.

Interpretation of chaos game fractals such as those shown in Figure 2 requires a good deal of biological knowledge. The lack of methylation sites is only obvious in Figure 2 if you know the sequence for a methylation site and can picture where these sites are on the chaos game’s square. This problem becomes more acute when an attempt is made to use these techniques to derive visual representations of protein sequences. Proteins are built out of twenty building blocks, the amino acids, rather than the four bases of DNA or RNA. In [Solovyev, 1993] both placing the 20 amino acids in a circle and extending the fractal into a third dimension are attempted. As one would expect, the interpretation difficulties grow. A number of biological issues can be used to inform the choices made when designing a biological representation for a fractal. The map from nucleic acid to protein reads DNA in triples, to produce 64 codons, which in turn are taken by a many-one map (the genetic code) onto the 20 amino acids and a “stop” codon. This stop codon indicates the end of transcription of a given sequence of DNA. The many-one map that forms the genetic code is the same in almost all organisms, but the choice of which of several possible codons to use to specify a given amino acid has a substantially organism-specific character. These biological considerations will factor into the design of evolvable fractals in the next section.

1.2 Iterated Function Systems

Chaos games are a particular type of iterated function system [Barnsley, 1993]. In an iterated function system a number of maps that take a metric space (in this case the real plane with the standard Euclidean metric) to itself are chosen. These maps are then called in a random order, according to some distribution, to move a point. The orbit of this point in the metric space is called the attractor of the iterated function system. In [Barnsley, 1993] a number of theorems about iterated function systems are established. A function from a metric space to itself is called a contraction map if, for any pair of points, mutual distance decreases with the application of the map. An iterated function system made entirely of contraction maps has a bounded fractal attractor.

A rich class of maps that are guaranteed to be contraction maps are similitudes. A similitude is a map that performs a rigid rotation of the plane, displaces the plane by a fixed amount, and then contracts the plane toward the origin by a fixed scaling factor. The derivation of a new point \((x_{\text{new}}, y_{\text{new}})\) from old point \((x, y)\) with a similitude that uses rotation \(t\), displacement \((\Delta x, \Delta y)\) and scaling factor \(0 < s < 1\) is given by:

\[
x_{\text{new}} = s \cdot (x \cdot \cos(t) - y \cdot \sin(t) + \Delta x) \quad (1)
y_{\text{new}} = s \cdot (x \cdot \sin(t) + y \cdot \cos(t) + \Delta y) \quad (2)
\]
To see that a similitude must always reduce the distance between two points, note that rotation and displacement are isometries, they do not change distances between points. This means any change is due to the scaling factor which necessarily causes a reduction in the distance between pairs of points.

2 Evolvable Fractals

Our goal is to use a data driven fractal, generalizing the four cornered chaos game, to provide a visual representation of sequence data. It would be nice if this fractal representation could work smoothly with DNA, protein, and codon data. These sequences, while derived from one another, have varying amounts of information and are important in different cells of operation. The raw DNA data contains the most information and the least interpretation. The segregation of the DNA data into codon triples has more interpretation (and requires us to work on coding, as opposed to intronic, untranslated, or junk DNA). The choice of DNA triplet used to code for a given amino acid can be exploited, for example, to vary the thermal stability of the DNA (more G and C bases yield a higher melting temperature) and so the codon data contains information that disappears when the codons are translated into amino acids. The amino acid sequence contains information focused on the mission, within the cell, of the protein. This sequence specifies the protein’s fold and function without the codon usage information muddying the waters.

Given all of this, we settled on an iterated function system fractal which both evolves the contraction maps used in the system and the choice of which contraction map is triggered by what biological feature. For our first series of experiments we decided to operate on DNA codon data, rich in information but with some interpretation. Our test problem is reading frame detection, a standard and much studied property of DNA. Reading frame refers to the three possible choices of groupings of a string of DNA into triplets for translation into amino acids. Figure 3 shows the translation into the three possible reading frames of a snippet of DNA. Only the first reading frame contains the ATG codon for the amino acid Methionine which also serves as the “start” codon for translation and TAG one of the three possible “stop” codons.

The correct reading frame for a piece of DNA, if it codes for a protein, is typically the frame that is free of stop codons. Empirical verification shows that frame-shifted coding DNA is very likely to contain stop codons, which is also likely on probabilistic grounds for random models of DNA. We remind the reader that random models of DNA must be used with caution; biological DNA is produced by a process containing a selection filter and therefore contains substantial non-random structure. Figure 2 serves as an example of such non-random structure.

<table>
<thead>
<tr>
<th>Interpretation</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>First similitude</td>
<td>( t_1(\Delta x_1, \Delta y_1) s_1 )</td>
</tr>
<tr>
<td>Second similitude</td>
<td>( t_2(\Delta x_2, \Delta y_2) s_2 )</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Last similitude</td>
<td>( t_n(\Delta x_n, \Delta y_n) s_n )</td>
</tr>
<tr>
<td>Index</td>
<td>( i_1, i_2, \ldots, i_{64} )</td>
</tr>
</tbody>
</table>

Figure 3: A piece of DNA translated in all three possible reading frames. Amino acids are given by their three letter codes which may be found in [Setubal and Meidanis, 1997].

2.1 The Fractal Chromosome

The data structure or chromosome we use to hold the evolvable fractal has two parts; a list of similitudes and an index of DNA triples into that list of similitudes. This permits smooth use of the fractal on DNA, DNA triples, or amino acids by simply modifying the way the DNA or amino acids are interpreted by the indexing function. A diagram of the data structure is given in Figure 4. Each similitude is defined by four real parameters in the manner described in Equation 1. The index list is simply a sequence of 64 integers that specify, for each of the 64 possible DNA codon triples, which similitude to apply when that triplet is encountered.

In order to derive a fractal from DNA, the DNA is segregated into triplets with a specific reading frame. These triplets are then used, via the index portion of the gene, to choose a similitude to apply to the moving point. This permits evolution to both choose the shape of the maximal fractal (the one we would see if we drove the process with data chosen uniformly at random) and also to choose which DNA codon triplets are associated with the use of each similitude. Any contraction map
has a unique fixed point. The fixed points of the eight similitudes we use play the same role that the four corners of the square did in the chaos game shown in Figure 2.

If we are to apply an evolutionary algorithm to the chromosome given in Figure 4 then we must specify the variation operators. We employ a single two-chromosome variation operator (crossover operator) that performs one point crossover on the list of eight similitudes and two point crossover on the list of 64 indices. We have two single chromosome variation operators (mutations) available. The first, termed a similitude mutation, modifies a similitude selected uniformly at random. It does this by picking one of the four parameters that define the similitude, uniformly at random, and adding a number selected uniformly in the interval \([-0.1,0.1]\) to that parameter. The scaling parameter is kept in the range \([0,1]\) by reflecting the value at the boundaries so that numbers \(s\) in excess of 1 are replaced by \(2 - s\) and values \(s\) below zero are replaced by \(-s\). Our second mutation operator, called an index mutation, acts on the index list by picking the index of a uniformly chosen DNA triple and replacing it with an index selected uniformly at random.

3 Experimental Design

The most difficult task, after selecting a fractal representation, was selecting a fitness function. The moving point is plotted a large number of times and we want the fractal in question to look “different” for different types of data. In the experiments reported here we seek to separate only two sorts of data, in- and out-of-frame DNA. For this relatively simple task we chose to track the average position of the moving point when it was being driven by the two types of data and make the fitness function the distance between the two average positions. A more precise description follows.

We performed two sets of simulations. The first used a pair of random sources, one of which was free of stop codons and the other of which generated DNA with stop codons. The second set of simulations used the genome of mycobacterium tuberculosis, in two different reading frames, cyclically, in place of the random sources. For a given fractal we evaluated fitness by computing the mean position of the moving point for each type of data: with/without stop codons or in-frame/out of frame natural DNA. There was an initial “burn in” period of 1000 iterations in which the moving point was subjected to fully random data. During this period the mean position of the point and maximum distance from it were estimated to permit normalization of the fractal image. By “fully random data” we mean that each of the eight similitudes was selected uniformly at random during the burn in period. Subsequent to burn in, the moving point was acted upon by data driven similitudes for about 200,000 iterations, with each type of data being used in bursts of 50-5050 iterations. The length of these bursts was selected uniformly at random. The fitness evaluation ceased when the point had just finished a burst of one type of data and was over 200,000 iterations.

Individual fractals were initialized with similitude parameters chosen as follows: \(0 \leq t \leq 2\pi, -1 \leq \Delta x, \Delta y \leq 1\), and \(s\) is the average of two uniformly distributed random variables on the interval \([0,1]\). The evolutionary algorithm used is steady state [Syswerda, 1991]. In each mating event, four individuals are chosen uniformly at random, without replacement, from the population. The two most fit are then crossed over and the resulting new fractals replace the two least fit. One of the new fractals is subject to a similitude mutation, the other is subjected to an index mutation. Each simulation is run for 10,000 mating events. We performed 30 simulations using the two random sources and 20 using the mycobacterium tuberculosis genome.

4 Experimental Results

![Figure 5](image5.png)

Figure 5: The attractor for the fractal receiving best fitness in simulation 16 with two random sources. The upper image displays the points plotted with stop codons in the data, the lower displays points plotted with no stop codons in the data.

A number of preliminary simulations were performed to find workable settings for population size, number of
mating events, and frequency of mutation. A relatively low mutation rate seemed to generate steady progress in the preliminary simulations on the random source data and so a relatively low rate was chosen for the reported simulations. We found fitness to increase during the simulation, indicating nominal functioning of selection in the evolutionary algorithm, though the mycobacterium tuberculosis data generated substantially higher fitness values. In retrospect, we believe this is because the biological data generated the more difficult problem and led the simulation to exploit a defect in our fitness function which we discuss in detail in Section 5.

The fractals located during our simulation has little trouble separating either the random source data or the mycobacterium tuberculosis data into the distinct reading frames, in the sense that fitness increased during the run, as shown in Figures 7 and 8. The fractals located plot distinct distributions of points for the different sorts of data. For the two-random-source data the separation was often visual, as can be seen in the example shown in Figure 5. The visual separation was far more subtle for the mycobacterium tuberculosis data; at best it was clearly visible when the dots were plotted in distinct hues. An example of the resulting image without such chromatic enhancement is shown in Figure 6. In both cases the attractor splits into two distinct regions, but, for the mycobacterium tuberculosis data, these regions have very similar shapes and are simply displaced from one another a modest distance. For the two-random-source data, many of the fractals manage to map the data into distinct parts of the attractor. One of the clearer examples of this is shown in Figure 5.

The fitness plot shown in Figures 7 and 8 are somewhat remarkable for the output of an evolutionary algorithm. The fitness is not leveling off with time, rather the rate of fitness accumulation seems quite steady. This is discussed in detail the next section, but to first approximation it indicates a problem with the fitness function. There is a manner, detailed subsequently, in which the descendants of a good chromosome can increase their fitness constantly without solving our problem any more effectively.

Figure 7: The mean fitness over all 30 simulations performed with two random sources. The error bars represent one standard deviation.

Figure 8: The mean fitness over all 20 simulations performed with mycobacterium tuberculosis data. The error bars represent one standard deviation.
5 Discussion and Conclusions

The experiments presented here represent a mixed success. The goal of giving clear visual separation of distinct data types was achieved only on the very clean data from the random sources. The biological data, drawn from the mycobacterium tuberculosis genome, did not separate in a visually striking way. We can salvage part of this goal by noting that the evolutionary algorithm did find fractals that performed the separation in a manner a computer would have no trouble interpreting. The problem with this is that many statistical methods with less computational overhead that our fractal technique can also perform such a separation. It may also be that using the entire mycobacterium tuberculosis genome, which is not all cleanly in frame or out of frame, was simply too hard a training set. We might have done better to train on a “cleaned” subset of the genome and then use the resulting fractal to separate reading frames. This is a potential direction for future work.

As far as proof of concept for an evolvable fractal representation goes we can claim success. The similitude and index table representation functioned very smoothly on the clean synthetic data and there is good hope that experience learning to use the representation will give us elegant representations on biological data. The success of the representation occurred in spite of what was, in retrospect, a less than perfect fitness function. Alert readers will have already spotted a potential flaw in our fitness function.

![Figure 9: Mean population fitness for three of the runs performed with two-random-source data. Samples are taken every 100 mating events.](image)

In the action of any similitude there is a displacement term \((\Delta x, \Delta y)\). The fitness function used is the distance between the mean positions of the moving point for each of two types of driving data. All other things being equal (which they may sometimes be in this instance), mutations that increase displacement for similitudes used in conjunction with only one of the two types of data drive up fitness without changing the shape of the fractal much. Given that we normalize the fractal images to a fixed size, this is a source of fitness essentially irrelevant to the problem we set out to solve.

The fitness landscape induced by the fitness function used in this work is somewhat convoluted near the origin of parameter space. As the similitude parameters grow, however, particular populations of fractals find themselves on infinite uphill ridges that correspond to scaling the displacement terms in their similitudes. Examining individual fitness traces, three of which are shown in Figure 9, we see substantial differences in the rate of accumulation of fitness. We conjecture that many of the basins of attraction for our selection process contain infinite uphill regions. The actual character of the uphill slopes probably vary. If, for example, three displacement terms must be scaled in roughly equal amounts to stay on the uphill “ridge” the hill climbing induced by mutation will operate more slowly than if increasing a single parameter permits uphill progress. If the continued fitness increase throughout each simulation has the cause we conjecture, then a simple normalization of the points plotted into a unit circle will render the maximum difference of mean positions fitness function more useful.

The effect of this peculiar fitness landscape on our simulations was probably not too great, in the sense that acceptable solutions are among those able to exploit vacuous fitness amplification. We did notice that some solutions tended to collapse the points plotted for one type of data into a very small part of the attractor. The vacuous fitness amplification probably made those solutions substantially worse, visually. As mutation increases the displacement terms of similitudes associated with the larger portion of the attractor the normalized fractal image may experience reduction in the displayed size of the smaller portion of the attractor. Those fractals that placed the points associated with each of the two types of data on opposite sides of the fractal did not, apparently, experience degradation of visual separation as a side effect of vacuous fitness amplification. Looking at the fitness plots for the two different sources of data, Figures 7 and 8, it is plausible that vacuous fitness amplification amplification was more marked on the harder problem.

There are a large number of sources of non-random data that also do not have obvious patterns. The similitude and index list fractals described here are one type of evolvable visualization for any such data set. The key issue is the utility of this visualization. At present we clearly demonstrate we can evolve such visualizations but do not yet make a clear case for their utility. Let us speculate about possible sources of utility. In a sequence database there are a large number of sequences which look quite similar unless various database search tools or visualization engines are applied. If we associate a small
fractal (we in part chose our representation for rapidity of generation) with each record in a browser then that fractal would serve as a kind of visual signature for the sequence. If each fractal were fairly unique then there would be a modest utility in giving the sequences a visual tag - but this function is already fulfilled by more informative sequence annotations. To have novel utility, the fractals we associate with our sequence browser should convey some sort of meaningful information about the sequence.

This meaningful information would probably come from associating portions of the attractor of a single iterated function system with common motifs in the sequence database. Stop codons are a simple type of motif, and we managed to associate distinct parts of an iterated function system’s with them in the experiments reported here. This suggests that we should fix a few motifs, evolve an iterated function system that provides a clean visual separation of these motifs, and then use fractals derived from that single iterated function system as the added visual annotation for a sequence browser. Such annotations would permit a form of simple visual summary to help a researcher spot patterns in sequence data. These remarks can be adapted to other types of data.

6 Future Work

An obvious direction for future work is to apply the evolvable fractal visualization to other types of data. More carefully selected biological data is high on our list. Another natural application is as a web browser enhancement. If we associated similitudes with lists of evolved or rationally selected keywords, phrases, or other content metrics then we could annotate the results produced by a web search engine with visual cuing fractals. The web browser domain may be an easier one in which to demonstrate some utility in using fractal annotation.

A second thread we would like to follow is the one implicit in our discussion of the flaws in the fitness function used in this work. For an evolutionary algorithm to function efficiently it is good if the fitness function, inside a tight loop, is not too slow. Similitude based fractals can be computed quite rapidly, as can the mean position of a moving point. As Barnsley’s work has show a large number of pictures can be built up using similitudes, our current choice of representation and fitness function permitted rapid initial investigation. Several other fitness functions are possible. One could assign a region of the plane to each type of data and score the number of times the moving point is within the correct region. Such a fitness function has only a modest additional cost and would likely permit somewhat more control over the appearance of the resulting images. In particular we hope it would help avoid the “just moved over a little” problem we encountered with the mycobacterium tuberculosis data. There are also various functions that can compute when two images “look alike” [Ashlock and Davidson, 1999], [Ashlock and Davidson, 1998]. These might be useful to train an iterated function system to have distinct appearances when presented with distinct types of data.

One property of iterated function systems is that they are forgetful. Any contraction map moves the entire metric space towards its fixed point. A system of such maps moves the entire space toward the attractor of the iterated function system. This process degrades information about the previous position of the moving point used to sample the attractor. For the four cornered chaos game the number of steps (data items) in the past that have any influence on the current position of the moving point is $\log_2(P)$, where $P$ is the number of pixels along one side of the square. With similitude based fractals we replace the 2 in the preceding expression with the reciprocal of the largest scaling factor of any similitude, but the effect is the same; only recent data matters. Finite state machines have been a part of machine learning since the 1960’s [Fogel et al., 1965] and can be used to reduce the problems that similitude-based chaos game fractals have with forgetfulness. It is simple to picture at least one representation for a finite state iterated function system. Take a finite state machine and permit it to be driven by the data. Instead of associating patterns in the data with particular similitudes we associate each state or transition of the finite state machine with a similitude. These would be, respectively, Moore or Mealy chaos automata. Such modified iterated function systems can encode long-term memory in the finite state transition function, permitting evolvable fractals to be trained to search for longer range correlations.

7 Acknowledgements

The authors gratefully acknowledge the support this research has received from Pioneer Hi-Bred International. The authors would like to thank the ISU Maize Genome project for the use of its computers for performing the simulation studies reported in this paper. Dr. Stephen Willson of the Iowa State Departments contributed many helpful suggestions and served as a repository of wisdom on the subject of iterated function systems and fractals.

References


