Indirect Molecular Evolution

Adam Prügel-Bennett
NORDITA, Blegdamsvej 17, DK-2100 Copenhagen Ø, Denmark

August 20, 1996

Directed molecular evolution is one of the most promising techniques for developing new macro-molecular reagents [1]. Rather than attempt a rational design, the blind search of evolution is combined with the ability to sift through huge number of molecules (typically 10^{13}) to discover more effective mutant strains. This has recorded notable successes particularly in developing RNA catalysts (see, for example, [2]). A variant of the technique has also been used to evolve proteins [3], but using far few molecules. This number is sufficient for finding single, and even double, beneficial amino-acid substitution, but complex mutations involving multiple substitutions are much less probable. In this letter we suggest a method for exploiting a huge number of molecules in evolving a DNA sequence coding for a protein. This method uses populations of mutated proteins to guide the search but does not require the identification between a fit (well adapted) protein and its DNA, thus alleviating one of the obstacles in applying directed evolution to proteins.

A typical application of directed evolution might be to discover an enhanced RNA catalyst. This would be achieved by combining a step of replication and mutations with a selection step. The first step, often referred to as mutagenesis is commonly accomplished by encoding the RNA as DNA, which can be replicated many millions of times using a polymerase chain reaction (PCR): this can be controlled to give the desired degree of mutations. The DNA is copied to RNA using a *transcriptase* enzyme. The selection or screening step is frequently accomplished by arranging for the effective RNA to stick to a substrate; the less fit RNA being washed away. This screening process can select up to one molecule in a billion. The ability to select a minute fraction of the population makes it possible to explore a relatively large portion of the space of possible sequence. Using a *reverse-transcriptase* enzyme the RNA may be converted back to DNA and the whole processes repeated.

Unfortunately this process cannot be directly applied to the evolution of proteins since there are no *reverse-ribosome* allowing an effective protein to be copied back to RNA or DNA. Directed evolution of proteins have been achieved by confining the DNA and protein production within cells [3]. This allows the DNA coding for the effective proteins to be identified but at the cost of greatly reducing the number of mutant strains that can be screened.

However, by relaxing the requirement to go directly from the phenotype back to the genotype it is possible to exploit very large number of molecules. Three ingredients are required:

I. a mutagenesis step such as PCR. This should replicate DNA with some small probability for mutation.

II. a way to produce proteins from a large number of DNA sequences. For example, by using *transcriptase* to go from DNA to RNA and *ribosomes* to go from RNA to proteins.

III. a screening method which measures the performance of the best few proteins in a large population. The effectiveness of this approach will depend on how small a fraction x of the best members of the population can be measured. The "fitness" of the population is taken to be the average performance of this fraction x.

With these three processes we propose the following "bio-algorithm" for evolving proteins

1. A DNA sequence coding for some good initial protein is chosen as the first "founder" sequence. Using process I the DNA is replicated to create P "progenitor" sequences. P should be some manageable number, say 100.

2. From the P progenitors P "populations" of DNA sequences are produced by again using the mutagenesis step I. This time the population should consist of a large number of DNA molecules, say 10^{13}.

3. Using process II the DNA molecules are converted to proteins. This is done for each of the P populations separately.

4. Using process III the fitness of each population is measured.

5. The best population is selected. When two populations have the same, or almost the same, fitness a larger fraction x of the population is tested so that the population with the largest yield of good proteins is chosen.

6. The progenitor of the winning population is used as the founder of the next generation. P new progenitors are chosen randomly from the best population. Step 2 is repeated until there is no more improvement.

This algorithm is much less direct than conventional directed evolution. The populations are used merely as a "guide field" for selecting progenitors. It provides information about how close the progenitors are to a good protein. The progenitors themselves need not even code for viable proteins as long as a sufficient proportion of their mutated offspring are viable.

Is it possible for such an indirect evolutionary strategy to work? As a plausibility test we simulated the evolution of a problem with a similar combinatoric complexity, namely to evolve a line of text towards a target expression, chosen to be, *"nature hath framed strange fellows in her time"*. The aim is to reach the target expression starting from an expression composed of random words of the same length. The expression are scored as follows: the expression must consist of recognized words (recognized words are those existing in the UNIX dictionary, no distinction is made between upper and lowercase letters); furthermore when a word in the expression is identical to a word in the target the expression gets an additional score equal to the length of the word. Thus only the correct multiple mutations are rewarded.

We assume that we have the same three ingredients that we would for evolving proteins. To make the example concrete we suppose that we have a mechanism for replicating an expression to produce a population of 10^{13} expression which differ form the initial expression by 2 mutations on average. For this problem process II is redundant as there is no difference between the genotype and phenotype, nevertheless we assume that we measure the phenotypes so that all genetic information is lost. In process III we assume that we can measure the average fitness of the best 10^4 expressions in the population. Thus we are able to measure a fraction $x = 10^{-9}$ of the expressions. To make the problem more realistic we assume that there is some error associated with measuring the fitness of the population. We model this by adding a random Gaussian noise with a standard deviation equal to 5% of the fitness.

With these ingredients we can build an algorithm for evolving expressions exactly analogous to that described for evolving proteins. At each generation we use $P = 100$ populations. To compensate for the noise we make two refinements. We consider two populations to have the same fitness if their fitnesses differ by less than 10% of each other. If two populations have the same fitness we decide between them by looking at the average fitness of a fraction $10 \times x$ of the populations. The second refinement is to make 20 progenitors exact copies of the founder. The other 80 progenitors are selected from the mutant population of the founder. This reduces the chance of the true fitness degenerating due to random noise.

```
0   harold cast foster soldier caloric if ceq cafe
4   harold cast foster soldier caloric if ceq tafe
7   harold cast foster soldier caloric if ceq tape
8   harold cast foster soldier caloric is ceq tape
20  harold cast foster soldier caloric is heq tape
21  harold cast foster soldier caloric is hex tape
22  harold cast foster soldier caloric is hex tame
30  harold cast foster soldier caloric is hex time
36  harold catt foster soldier caloric is hex time
37  harold cath foster soldier caloric is hex time
44  harold hath foster soldier caloric iz hex time
49  harold hath foster soldier caloric ia hex time
72  harold hath foster soldier caloric va hex time
77  harold hath foster soldier caloric ea hex time
90  harold hath foster soldier caloric en hex time
99  harold hath foster soldier caloric in hex time
105 harold hath fosmer soldier caloric in hex time
114 harold hath frsmer soldier caloric in hex time
119 harold hath framer soldier caloric in hex time
125 harogd hath framer soldier caloric in hex time
134 harogd hath framer soldier caloric in her time
138 harozd hath framer soldier caloric in her time
164 harozd hath framer soldier caeoric in her time
171 harozd hath framer soldier canoric in her time
174 harozd hath framer soldier canonic in her time
181 harocd hath framer soldier canonic in her time
190 harocd hath framer soldier canonic im her time
194 harocd hath framer soldier canonic if her time
```

Figure 1. List of founder expressions at different generations. Only the expressions that differ from that of the previous expression are shown.

A typical evolution of the expression sequence is shown in figure 1. Only the founders which differ from those in the previous generation are shown. Figure 2 shows the number of overlapping letters between the founder expression and the target expression for different generations.

To calculate the evolution we need only know the probability of a word mutating to a non-word and to the target word: we do not need to know what strings actually exist in the population. Consequently the simulation takes only around one minute to run.

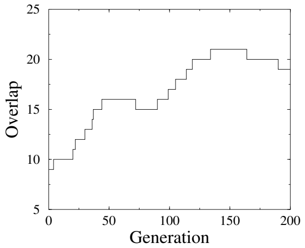

Figure 2. Plot of the number of common letters between the founder expression and the target expression versus the number of generations. The data is taken from the same simulation as figure 1.

The evolution is quite slow as it requires a beneficial mutation in one of the P progenitors. The combinatorics makes these events rare. The two most important factors in determining the efficiency of this approach are the smallness of the fraction x that can be measured and the size of the noise in measuring the fitness. We have attempted to tune the algorithm to get optimal performance since the optimal parameters are likely to depend on the details of the problem. The algorithm, however, appears quite robust; giving a similar performance over a wide range of parameter choices. The evolution always appears to reach an equilibrium where the rare good mutations are cancelled out by the far more common bad mutations. In the example above the equilibrium was around an overlap of 20. Reducing the mutation rate appears to increase the fitness of the equilibrium although it also increases the time taken to reach the equilibrium. It is likely to be beneficial to reduce the mutation rate over time. There are in fact two mutation rates in the algorithm; that for the population and that for progenitors. It is the latter mutation rate which should be reduced. In the simulation we used the same mutation rate so the progenitor could be drawn at random from the population. The mutations in the population could also be introduced in process II where the DNA is copied to proteins. An alternative and possibly better algorithm would be to select the progenitors for the next generation, not from a single population, but from the best few populations; these may even be recombined to form mixtures of the sequences in the previous generation [4]. This would be more in the spirit of genetic algorithms [5].

The proposed algorithm is undoubtedly successful for the word problem. Words requiring a large number of simultaneous mutation have been found. These would be exceedingly unlikely without using the population to guide the search. Words requiring a very large number of simultaneous mutations are not, however, discovered.

References

[1] G. F. Joyce. Directed Molecular Evolution. *Scientific American*, 267(6):90–97, 1992.

[2] D. P. Bartel and J. W. Szostak. Isolation of new ribozymes from a large pool of random sequences. *Science*, 261:1411–1419, 1993.

[3] J. Moore and F. H. Arnold. Directed Evolution of a para-Nitrobenzyl Esterase for Aqueous and Organic Solvents. *Nature Biotechnology*, 14:458–467, 1996.

[4] W. P. C. Stemmer. Rapid evolution of a protein *in vitro* by dna shuffling. *Nature*, 370:389–391, 1994.

[5] Melanie Mitchell. *An Introduction to Genetic Algorithms.* Complex Adaptive Systems. Bradford, 1996.