Optimising genes with a genetic algorithm

DNA can code for amino acids, the building blocks of proteins. Three DNA bases (each A, T, G, or C is a “base”) code for one amino acid. Each coding set of three DNA bases, called a *codon*, is translated to a specific amino acid. For example CAA is translated to Glutamine. But there’s more than one way to skin a cat and there’s more than one way to code almost any of the 20 amino acids in the genetic code. Using Glutamine as an example again, CAA and CAG both give us the same amino acid. Different codons can do the same job.

But they aren’t necessarily equally good at that job. Because of this some codons are used more often. For instance when encoding Glutamine, CAG is used 3 times more often than CAA. This bias for particular codons also applies to pairs of codons. Some pairs of codons occur more often than others which code the same pair of amino acids, even more often than you would expect given the frequency of the individual codons. The codon CCG is the rarest codon for Proline and CCC is the most common; by chance you would expect CCGCCG codon pairs to be 7 times rarer than CCCCCC codons but they’re actually more common.

So why is this important? Optimising a gene in this way can allow you to get more of your desired protein. A protein you might need to catalyse a chemical reaction, or produce a therapeutic drug. Optimising single codons is easy, you just pick the most common codon each time. (In reality genes are more complicated than that but as an optimisation problem, pretty simple). But I wanted to go to the next step and optimise codon pairs. Given an amino acid sequence, how do I produce a DNA sequence that encodes it correctly and uses the most common codon pairs?

How do we do it? It’s a little more complicated because each codon pair overlaps one after it. The first codon pair is made of the 1st and 2nd codon, the second codon pair is made of the 2nd and 3rd codon. The 2nd codon is doing two jobs! I originally thought this would be no problem; if I pick the 1st and 2nd codons to give the best codon pair for the first two amino acids then I can pick the 3rd codon based on what pairs with the already chosen 2nd codon. And just repeat this until I had a complete coding DNA sequence.

This is a greedy algorithm, it takes the best thing available at the time and dam the consequences. The trouble with this approach is that it only cares about the score for one codon pair at a time despite its choice affecting the score of the next codon pair. This means that a picking the best codon pair to start with locks us into a really bad pair next, whereas the 2nd best codon pair could have allowed us to pick a really good pair next. It may only make a small difference for each codon pair but it will add up over the whole sequence.

Once I had realised this, I knew that to properly optimise my gene I needed to consider all the codon pairs at the same time. We can’t just brute force it either as the number of possible DNA sequences for an amino acid sequence of length L is approximately 3L. Fortunately there are plenty of approaches to optimise over a large problem space with many local maxima. As we’re optimising a gene, I decided to use a genetic algorithm.

In brief, a genetic algorithm is like a miniature simulation of evolution. A bunch of random possible solutions to the problem are created and scored, in our case the some of the codon pair scores over the whole sequence. Explain in more detail.

More detail about how to frame this problem to minimise problem space and express it as a 1D vector.

