DNA-Inspired Coding for Information Transmission

Kieran Greer,
Distributed Computing Systems, Belfast, UK.
http://distributedcomputingsystems.co.uk

Abstract **–** This paper describes a method for coding and then decoding information that might be sent over the internet. It satisfies the requirements of the Herox Evolution 2.0 competition that the coding should be based on a chemical reaction, be generic and that there should be no designer involved. The paper suggests that coding at both ends might be preferred and that it is possible to pass a set of instructions instead of information across the Internet, leaving the message content more secure.

Keywords: Encoder-decoder, DNA, information transmission, chemical reaction.

1 Introduction

This paper has been written in response to the HeroX Evolution 2.0 competition[1]. The competition asks the question if a DNA-style of encoding and decoding can be used to transmit information. Discovering this will also lead to insights as to how DNA itself codes and reproduces, to generate new life. The rules require that a chemical process is used for the coding and that no designer is involved. The process must be generic, to work with any coding or information set and it should produce unique results for any coder/information combination. The solution that is proposed in this paper is incredibly straightforward, but it is a method that would satisfy these requirements. Please excuse any loose terminology.

[1] http://www.herox.com/evolution2.0

2 Biological Comparison

With DNA and the code of life, we have the gene pool that is the set of chemicals, DNA, RNA or whatever, as the building blocks and the DNA genetic code as the set of coding instructions. This code sequence is added to the nucleus of every cell that is created and it helps the cell to grow and re-produce. The DNA pool is made from only 4 types of an entity called a Nucleotide, which can be either an A, C, G, or T. For DNA, each symbol is in fact a combination of 3 nucleotide letters. The genetic sequences are very long however and so the genetic code is still too complex to be fully understood through computer algorithms. This paper proposes that the information system and coding schemes are simply what we see already. Let the gene pool be the set of letters or symbols in the information system and let the genetic code be the sequence of coding instructions, as described next.

While this is a biological solution, it is helpful to think in terms of the algorithm as well. In the proposed computer algorithm, the individual symbols of the system are added as the first instructions in a list. Each instruction after that then tells the algorithm to combine two of the earlier instructions. The instructions influence how the DNA forms and they fire in order, through the list, from the start to the end. For the DNA code, this would be from one end of the helix strand to the other. Then to simulate the chemical process:

- As each instruction fires, it stimulates or encourages that combination of RNA and proteins in the gene pool to combine.
- But there is also the question of what the gene pool is made from and so if there are statistically more of a particular combination in the pool, then that combination is more likely to happen first.

For example, if we have 3 AAA's, 3 TTT's and 3 CCC's in the pool and the genetic code has AAA-TTT as the first compound instruction and then AAA-CCC:

- At time 1, AAA-TTT fires first and encourages an AAA and a TTT protein to combine. Statistically this gives that combination an edge over the AAA-CCC alternative.
- After that event, there are only 2 AAA's, 2 TTT's and 3 CCC's in the pool.
- Again, at time 2, the AAA-TTT event fires first and then the AAA-CCC event. This time however, an AAA protein is more likely to meet a CCC protein in the pool and so while

the AAA-CCC event fires second, statistically those proteins are more likely to meet and so that event will probably occur instead.

- There are different ways of coding the statistical likelihood, but something like the product of the symbol counts in the pool is an option. That would be 9 for both instructions in the first instance, but then 4 and 6 respectively for the second instance.
- As for discriminating between the instructions, it can simply be the case that the instruction with the best score (product result) wins and if there is a tie, then the earlier instruction in the list wins. This is a very statistical process and it is easy to see that there is no principal designer and that it can be applied to any combination of genes and rules.
- It might also be possible to see that the DNA instructions being added to the code could be statistically influenced by how earlier organisms formed, with their base proteins being added first, followed by what chemicals tried to combine from that.

3 The Coding Algorithm

An algorithm has been written and tested on a computer to show how the process might work. The genetic blocks are represented by 3-letter symbols that make up the ACTG combinations. While the competition only required 32 distinct symbols, in fact any number can be used, including the 64 combinations that occur in nature.

- The gene pool is created by adding all of the base symbols and assigning arbitrary amounts for each symbol – 3 AAA's, 5 AAC's, and so on.
- Then a set of instructions can be generated randomly, which adds each individual symbol type as an instruction first, and then combines any 2 instructions below the current instruction to create it. This results in a list of instructions of increasing complexity, from the bottom to the top.
- To create the information message, it is simply the case of selecting the instruction with the largest score, where an earlier instruction wins a tie. An instruction can only be selected if it combines at least two other instructions.
- The instruction is evaluated by breaking it down into the number of each symbol type required to create it. Then the remaining amount of each symbol type in the pool is

retrieved and the differences between the required and available amounts counted. The product of these differences can be a measure to be how likely it is that the related proteins will meet in the pool.

- The used amounts are then removed from the pool after the instruction's compound is realized and the instruction compound is added to the message, again in the order that it is created.

The coding process produces a sequence of symbols that might look like the following, shown in Figure 1.

TAGTAGTAGTAGTCGTGGTGGTGGTGGTCCTAAATGATGATGAATAATAATTGCATCATCATCTGAT
GAAACAACAGGAGGACCACATATTATTCTTTCTTCTTCTTATTATTATTATCGAGAAAGACATCATTG
ATTTATACTATGTCATACTCAACAAGATACTGTTGAAAATCGTAATCCAGCTCTACTATGTCATACTCA
ACGTATAAATCGACAAGATACAATTTTAGCTCTACGTATAAATCGTCTAGTATATAC

Figure 1. A message created from a 3-letter pool of Protein compounds.

This sequence could be passed over the internet as the coded message, for example, when a decoder on the other side would then decode it back into the original message. Tests show that if there is even a single change to the protein amounts in the pool, that will result in a different message using the same encoder. Also, if the coded message is changed by a single protein symbol, that will result is a different pool after it is decoded. The changes would also be relatively small however and would reflect how small or large the pool change was. A change made to one of the coder's instructions is significant, depending on whether the coding process used all of the symbols or if any was left over. For example, a change might not result in a difference, if the instruction change resulted in one left-over symbol instead of another left-over symbol.

Looking at the message, it is easy to re-produce the original gene pool from it, because each symbol can be counted. This can of course be avoided by using a second mapping to hide what each symbol is and then decoding from the second mapping table back to the original

symbols first. It is however, more difficult to reproduce any structure that was created as part of the original message. While this should be possible with a decoder, the author wonders if the coding-decoding process is actually the most appropriate one. For the biological organism, for example, it does not really want to decode the message as well. It only wants the combination of DNA in the pool with instructions to be unique. If writing a coding system to transfer information over the Internet, for example, then the instruction set should probably be generated randomly, so that it has a unique and hidden association with the genetic pool. If the genetic pool has a strong influence over the creation of the genetic code, then the code can possibly be worked out from the pool contents. If this is not possible, then there must be other factors that make the two slightly different statistically.

The author does recognize however that the coding could be influenced by the composition of the DNA in the pool and resulting statistical order of growth, which means some type of feedback (decoding) from the created cells to the genetic code. However, the process for transmitting information can work successfully with only an encoder and so an alternative or equivalent to decoding, may be to pass the gene pool as the message and then ask the receiving side to code it again, using the same set of instructions. Without those instructions it will not be possible to know what the structure is and a different gene pool will produce a different result. Therefore, if the test is to produce the same message structure at the other side, using the same coding scheme, then this process can work, but it may be possible to improve it further, as described next.

3.1 Pass Instructions instead of Information

Traditional methods pass the information itself across the Internet. An alternative that is possible with this method would be to pass a list of instruction indexes instead, in the order that they were created. This list would simply be a sequence of index numbers that would contain absolutely no details about the information itself. It would even be possible to add empty or similar instructions into the instruction list, to confuse any patterns that are generated. Further, because instructions can be combinations of other ones, the index list could be much smaller in size than the potentially larger sequence of symbols that it creates and so there should be some saving on the length of message that is transmitted. Re-

executing the instructions in order would still preserve the order of the symbols in the message and therefore its' original structure. The instruction indexes for the coded message in Figure 1 are shown in Figure 2.

19-19-19-19-27-31-31-31-31-26-16-7-7-7-1-1-1-30-6-6-6-28-28-2-2-15-15-10-8-17-17-25-22-22-22-20-20-20-20-78-81-48-40-56-70-35-41-57-34-70-35-64-47

Figure 2. A list of instruction indexes that can create the message shown in Figure 1.

4 Complex Structures

While this method should be OK for something like passwords, it is more complicated for information in general, such as natural language. If information is passed as the message, then the structure is inherent with it. If the instructions are passed instead, then they need to code the structure as well, so that they can re-create it. For chemicals that create life, the instructions are based on the relative composition of the genetic pool and how that has caused the chemicals to react, statistically. This is still the properties of the genetic pool, but the structure is not a very complicated, being based on 4 nucleotide combinations. For something like natural language then, the pool set is a lot more diverse and it may not be realistic to try to code that into instructions. However, with current computer systems that can use AI with thousands or more of parameters to make conversations, it might actually be possible. Another problem would be that each data type with a different structure would require a different set of instructions, but maybe there is only text, image and video, for example. For each of these, it is the same process of extracting the rules that can create the structure of the information and it should be possible to do this, probably at a generic level. The next section gives one example.

4.1 Generic Text Encoder

Consider coding arbitrary text sentences, for example. A coder with the instruction set needs to be created. Let the symbol set by the full 26 letters of the alphabet, plus some punctuation, such as space, full stop, comma and hyphen, leading to 30 symbols in total. These can

represent the first 30 instructions in the coding table and in any order. Then the next set would combine any two of these instructions. That requires 30 x 30 instructions, which is 900 more instructions in total. Another level can then combine the lower levels into 3-4 symbol groups. That is a value somewhere below 930 x 930 = 865,000 instructions (1 million). If the system goes up to combining 5-8 symbols, then that would require up to 865,000 x 865,000 = 7.48×10^{11} more instructions. That might be too many, less than 1 trillion, but 3-4 symbol groups look to be OK. Note that every instruction can be in a random order and only requires the correct indexes to any other instructions it uses. Then it would be a matter of breaking the full sentence down into blocks of these symbols and replacing that block by the instruction index.

For example, consider the name 'Jack' and the instruction set: I-1 = j+a, I-2 = c+k, I-3 = I-1+I-2. Then Jack could be coded using 2 instructions I-1 then I-2, or the single instruction I-3.

5 Conclusions

This paper describes an encoder-decoder that can work with DNA-like information and code it using a process similar to a chemical reaction. The process has been expanded to encoding information in general, where in theory it can still work, but might require coding instructions that are specific to the type of information. A process for doing this is suggested however that is based on the same DNA style of system, but that uses existing structure and not a structureless gene pool. If a very secure environment is required and this process can be accommodated, then instruction indexes instead of the information itself can be passed as the message, making it a very difficult code to break. There are probably different ways to implement this type of solution.