An algorithmic and computational approach to Open Reading Frames in short dsDNA sequences: Evaluation of "Carr's Conjecture"

[1,2,4,*] **Steven M. Carr,** [2]**Todd Wareham, and** [3]**Donald Craig**
[1]Departments of Biology and [2]Computer Science,
[3]eHealth Research Unit (Faculty of Medicine),
Memorial University of Newfoundland, St John's NL A1B 3X9 Canada,
[4]Terra Nova Genomics, Inc., St John's NL A1C 2R4
*Correspondence author, e-mail scarr@mun.ca

Abstract - *In the genomic data-mining era of genetics and bioinformatics, a frequent task is the exploration of new and (or) unannotated double-stranded DNA (dsDNA) sequence data for the occurrence of protein-coding regions. The characteristic expectation is that five of the six possible three-letter reading frames for amino acids will be "closed" by one or more "stop" triplets in the Genetic Code: the sixth will be an "Open Reading Frame" (ORF) without "stops" that specifies a polypeptide sequence. The same constraint, which we designate the "5&1" condition, occurs in short dsDNA exemplars (ca. $15 \leq L \leq 25$ bp) used in genetic and bioinformatic education. We describe an algorithmic and computational evaluation of "Carr's Conjecture," that no dsDNA sequence of L~10 or less exists that satisfies the "5&1" condition. We show there are no solutions for $L \leq 10$, 96 for L=11, and that the number of solutions thereafter increases exponentially. Enumeration is practically limited by CPU time beyond L=25. We describe implementation of the algorithm as a web application that generates appropriately constrained dsDNA exemplars of length $L \leq 100$ bp, and their pedagogic utility and application.*

Keywords: DNA, mRNA, Genetic Code, stop codons, Open Reading Frames, data mining, webapps in genetics and bioinformatics

1 Introduction

The "cracking" of the Genetic Code by means of a rapid series of experiments and logical inferences is arguably the first instance of a "big science" approach in the history of molecular genetics [1]. Theoretical considerations had already indicated that any nucleic acid code words must comprise a minimum of three letters [2]. After demonstration in 1961 that an artificial poly-U RNA template directs incorporation of the amino acid proline into a polypeptide, and thus that UUU was the "code" for PRO, Marshall Nirenberg's lab had by 1963 deduced an incomplete "dictionary" of 50 three-letter "code words" [3], and a substantially complete Genetic Code table by 1965 [4] (Fig. 1). The iconic 4x4x4 table is now a standard feature of biology textbooks, and has been incorporated as a fundamental feature of bioinformatic computational schemes.

We consider here properties of short segments of Genetic Code that are of interest both theoretically as an unexplored computational challenge, and practically, as a pedagogic challenge for students and instructors. Taken together, solution of these challenges at the intersection of computational and biological science provides reciprocal illumination to each as an example of biological computation in 2014.

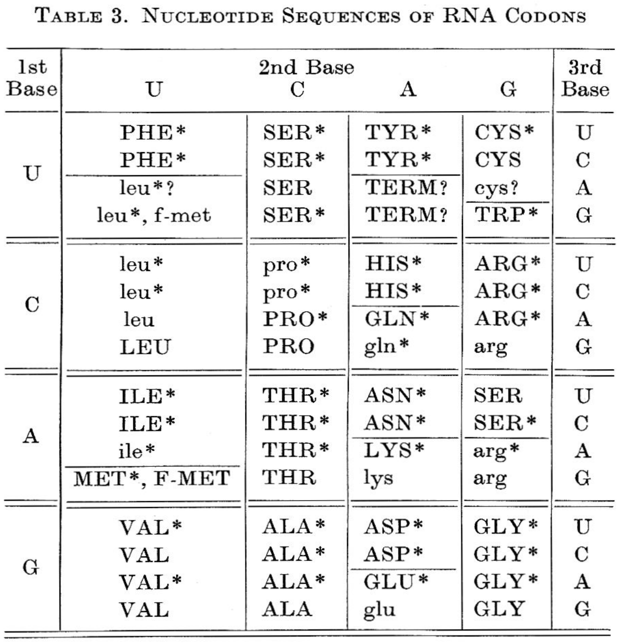
TABLE 3. NUCLEOTIDE SEQUENCES OF RNA CODONS

1st Base	2nd Base U	C	A	G	3rd Base
U	PHE*	SER*	TYR*	CYS*	U
	PHE*	SER*	TYR*	CYS	C
	leu*?	SER	TERM?	cys?	A
	leu*, f-met	SER*	TERM?	TRP*	G
C	leu*	pro*	HIS*	ARG*	U
	leu*	pro*	HIS*	ARG*	C
	leu	PRO*	GLN*	ARG*	A
	LEU	PRO	gln*	arg	G
A	ILE*	THR*	ASN*	SER	U
	ILE*	THR*	ASN*	SER*	C
	ile*	THR*	LYS*	arg*	A
	MET*, F-MET	THR	lys	arg	G
G	VAL*	ALA*	ASP*	GLY*	U
	VAL	ALA*	ASP*	GLY*	C
	VAL*	ALA*	GLU*	GLY*	A
	VAL	ALA	glu	GLY	G

Figure 1: The Genetic Code, 1965

2 Molecular genetic and bioinformatic considerations

2.1 Molecular genetics of DNA ➔ RNA ➔ Protein

Deoxyribonucleic Acid (DNA) is famously a double-stranded molecule (dsDNA) that comprises two polymeric sequences of four bases (A, C, G, and T) in an aperiodic order that conveys bioinformation. The two strands are arranged in anti-parallel 5'➔3' directions that are implicit in the deoxyribose component. The strands are held together by non-covalent hydrogen bonds between paired A + T or C + G "base pairs". The anti-parallel arrangement and base pairing rules ensures that the alternative strands are complementary to each other. This relationship is the basis of DNA as a self-replicating molecule.

One DNA strand, designated the template strand, serves as a template for 5'➔3' synthesis (transcription) of a complementary messenger RNA (mRNA) molecule, where RNA differs from DNA in being single-stranded and substituting base U for T. The mRNA molecule is translated in the 5'➔3' direction into a polymer comprising a sequence of amino acids (a "polypeptide"), according to a Genetic Code (Fig. 1). In the Code, each of the 64 possible three-letter base sequences ("codons") read 5'➔3' specifies a particular amino acid, except that three codons (UAA, UAG, and UGA) do not specify any amino acid, and therefore serve as terminators ("stops") to polypeptide synthesis. A common Genetic Code is universal for the nuclear genomes of all organisms.

2.2 Bioinformatic data-mining

Because the mRNA sequence is complementary to that of the DNA template strand, it necessarily has the same base sequence in the same 5'➔3' direction as the DNA strand complementary to the template strand, except for the substitution of U for T. This DNA strand, designated the "sense" strand, may therefore be "read" directly from the Genetic Code table, substituting "T" for "U". As a bioinformatic process, it is straightforward to read the polypeptide sequence directly from the DNA sense strand, without the intermediate molecular steps of mRNA transcription and subsequent translation via tRNA (see below).

Any dsDNA molecule may be read from six potential starting points, designated "reading frames." Reading frames are three-base windows that commence at the 1st, 2nd, or 3rd base from the 5' end of one strand, after which each frame repeats, or from the 5' end of the other strand starting at the opposite end of the molecule. Full-length DNA sequences of several hundred to more than a thousand bases that specify protein sequences hundreds of amino acids long are expected to show that only one of these reading frames is an "open" reading frame (ORF), that is, that it does not include a "stop" triplet over the required length of the polypeptide. [By definition, "codons" occur only in mRNA: the equivalent three-letter sequences in the DNA sense strand may be designated "triplets"]. As three out of 64 triplets are stops, the five alternative reading frames are expected to include multiple random stops at expected intervals of ~20 triplets: the first occurrence of a stop closes the reading frame. Commercial DNA software performs this process as a matter of routine, either from novel data or data mined from resources such as GenBank (e.g., Sequencher: Gene Codes, Ann Arbor MI). The sequence data are depicted one strand at a time, in a conventional left-to-right, 5'➔3' screen or text presentation with the inferred polypeptide sequences of one, two, or all three reading frames. The software must also be able to re-orient the data "upside down and backward" as a reverse complement simulacrum of the complementary strand in order to maintain this convention.

2.3. Pedagogic considerations: "Carr's Conjecture"

Introduction to the theory of data mining for ORFs typically begins with the propounding of a short dsDNA sequence of length L=15~25 base pairs. The exemplar sequence is constructed such that five reading frames are closed and only one is open (the "5&1" condition). The task for students is to identify the ORF and infer the correct polypeptide sequence from the Genetic Code. The challenges for instructors include construction of exemplars from scratch, where placement of multiple mutually compatible stop triplets in exactly five reading frames over a short distance is non-trivial. The double-strand nature of the DNA molecule means that specification of letters in one strand to create stops and ORFs mandates complementary changes in the other that may unintentionally create or destroy stop triplets and (or) ORFs.

The senior author therefore asked the second author for a computational algorithm that would provide exemplars that satisfied the "5&1" condition, and evaluate "Carr's Conjecture." By inspection, no solution to the "5&1" condition exists for L=5, which is the minimum-length dsDNA with three (single-triplet, single amino acid) reading frames on either strand. Then, there must exist an upper limit to L for which no solution exists, noting that exemplars of L=15 are common in teaching, and that for L=11 there are three (triple-triplet, or tri-peptide) reading frames on either strand.

3 Algorithmic & programming considerations

A practical algorithmic generator of ORF exemplars must be able to access the entire space of dsDNA sequences that satisfy the "5&1" condition for a specified L, sample that space in an at least approximately random manner, and be efficient in terms of both CPU runtime and required memory space.

As with a hand calculation, the most direct computational method would be to first generate all possible DNA sequences of length L, and then sample randomly from this generated set. Given 4^L possible sequences, this remains computationally impracticable in terms of memory and (or) runtime. Even if such a process were made more space-efficient by implementing enumeration in a recursive process that terminates as soon as an ORF exemplar is found, on inspection the small proportion of ORF exemplars relative to 4^L suggests that the time required to encounter such a sequence would be prohibitive.

```
function GenerateSkeleton(S, ORFNum, rfn)
  if rfn == 7 then
    return CompleteSequence(ORFNum, S)
  elif rfn == ORFNum
    return GenerateSkeleton(S, ORFNum, rfn + 1)
  else
    res = Null
    for (pos, stopCodon) pair in a randomization of the
        list of all possible such pairs for reading frame rfn
        of S do
      St = place stop codon stopCodon at position pos relative
          to reading frame rfn of S
      if that stop placement is possible then
        res = GenerateSkeleton(St, ORFNum, rfn + 1)
        if res is not equal to Null then
          exit for-loop
    return res

function CompleteSequence(ORFNum, S)
  if S has no more unfilled bases then
    return S
  else
    res = Null
    pos = position of unfilled base in S
    for base in randomization of list ["A", "G", "C", "T"] do
      St = S
      St[pos] = base
      if the number of stops in open reading frame of St equals 0 then
        res = CompleteSequence(ORFNum, St)
        if res is not equal to Null then
          exit for-loop
    return res

function generateRandomORF(seqLen)
  let S be a sequence of length seqLen of unfilled bases
  ORFNum = random selection from reading-frame numbers 1 .. 6
  return GenerateSkeleton(S, ORFNum, 1)
```

Figure 2: Pseudocode for the two-part recursive search algorithm

Instead, we developed a method that invokes a two-level recursive search that first generates a dsDNA “skeleton” with at least one stop codon in each of five frames, and then completes the remainder of the dsDNA sequence by adding bases at random to the skeleton so as to produce an ORF exemplar in which the “5&1” condition is maintained, i.e., the sixth frame remains open. The two levels of this search are described in the algorithms *GenerateSkeleton* and *CompleteSequence*, respectively, for which pseudo-code are given in Fig. 2. As required, access to the entire space of dsDNA sequences satisfying the "5&1" condition for a specified L is complete and random by virtue of the randomization of the lists of stop triplets and stop triplet positions at each stage of the recursion in *GenerateSkeleton* and the randomization of the DNA base to be added to the skeleton completion at each stage of the recursion in *CompleteSequence*. The ORF exemplars are then produced by *GenerateRandomORF*. Note that these are not fully random, as certain sequences may be generated multiple times through appropriate completions relative to different skeletons, and will hence be over-represented in the sampled sequence space. They are, however, sufficiently random for heuristic purposes. The algorithms were implemented in the Python 2.7 programming language.

4 Mathematical considerations

We would like to derive an accurate estimate of the total number of ORF exemplars of length L ($NORF_L$), and (or) an efficient enumeration of that number.

The upper bound on $NORF_L$ is simply 4^L, the total number of sequences of length L. To derive a lower bound, we note that a subset of the ORF exemplars for any L>10 must include the ORF examples for L=11 (96: see Fig. 4), supplemented by a completion of the open frame that does not contain any of the three stop triplets. There are of course 61 such completions. Note that it is immaterial for present purposes if such a completion generates additional stop codons in the closed reading frames. The size of this subset, which sets a lower bound on $NORF_L$, is

$$\begin{aligned}
&= 96 * 61^{\{((L - 11)/3) - 1)\}} \\
&= 2^{6.585} * (2(5.93)^{\{((L - 11)/3) - 1)\}} \\
&= 2^{\{(5.93 * (((L - 11)/3) - 1)) + 6.585\}} \\
&= 2^{\{((1.976 * ((L - 11)) - 5.93) + 6.585\}} \\
&= 2^{(1.976L - 21.08)}
\end{aligned} \tag{1}$$

The lower bound thus increases exponentially in L, which validates our use of a smart random ORF generator as described, rather than a naive all-strings random ORF generator. The recursive search algorithm, modified to an all-exemplars search that stores results and removes duplicate exemplars prior to enumeration, would therefore be expected to succumb to memory limitations. Alternatively, an exhaustive "brute force" algorithm that enumerates exemplars as they arise, without storing them, would be expected to succumb to CPU limitations, even when optimized to run on multiple machines. We implemented both alternative algorithms.

5 Results

5.1 Behavior of search algorithms with respect to L

The recursive and exhaustive algorithms show that there are in fact no solutions for L = 5~10, and 96 for L=11. Enumerations from the two methods agree for 11≤L≤19, at which point the recursive algorithm succumbs to memory limitations. For L<22, CPU usage for the exhaustive algorithm was measured on a single quad-core PC. For L≥22, CPU usage was measured over a network of such machines, and by L=25 exact CPU usage is obscured by competing demands from other users on the same network. Calculation of $NORF_L$ for L>25 with the resources available to us would require several days (Fig. 3).

5.2 Properties of $NORF_L$ with respect to 4^L

$NORF_L$ increases exponentially, and appears to converge on 4^L (Figs. 3 and 4). Although power curves calculated for values of L over the range L=19~N appear to provide close approximations of $NORF_L$ up to the (N-1) (r^2 > 0.999), as in Fig. 3 all such curves begin to diverge from 4^L rather than approach it, and all perform poorly as *a posteriori* predictors for L > (N-1).

The number of solutions for RFs 1-3

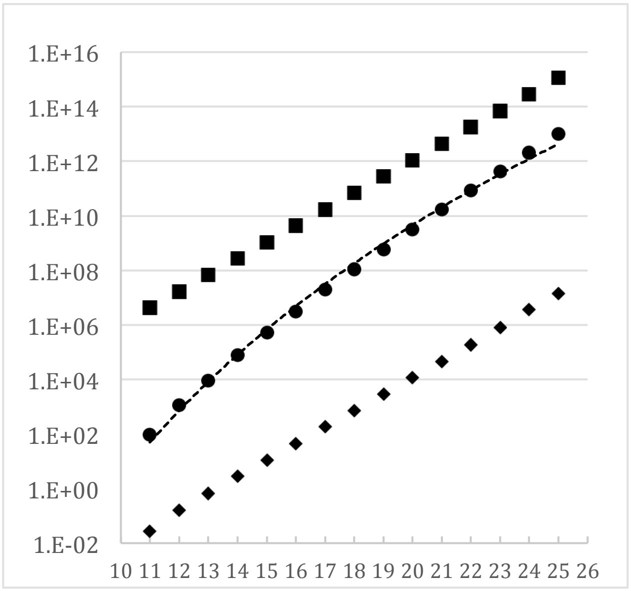

Figure 3: Semi-logarithmic plot of the enumerated number of ORF exemplars of length ($NORF_L$) for L = 11 ~ 25 (●). The power curve of best fit is $NORF_L = 6x10^{-31} x (L)^{30.659}$ (dotted line). The upper limit on $NORF_L$ is the total number of possible dsDNA sequences of length L, 4^L (■). Required CPU time for the exhaustive algorithm is given in seconds (◆); CPU is log-linear with respect to L, as CPU = $0.613(\log_{10} L) - 11.736$ (r^2 = 0.99978).

are symmetrical to those for RFs 4-6 respectively in all enumerations up to L=25 (Fig. 4). All RFs for L=12, 13, and 14 can encode tetra-peptides, but L=12 only for the 1st, L=13 only the 1st and 2nd, and L=14 for all three. Ordered alternatively, L=11 can encode tri-peptides in all three frames, whereas L=12 encodes a tetra-peptide only in the first, and L=13 in the first and second. Stated formally, if (L-2) and the number of amino acids are congruent modulo 3, equal numbers of amino acid residues are generated in the polypeptides from all reading frames. It is therefore counterintuitive that for L=11,14,17 there are more ORF exemplars in RFs 1 & 4, and that for L=12,15,18 and L = 13,16,19 there are more ORF exemplars in RFs 2 & 5 and 3 & 6, respectively. We suspect this is due to irregularities in strand-specific base composition imposed by the asymmetric base composition of TAA, TAG, & TGA stops.

L	RFs 1&4	RFs 2&5	RFs 3&6	Total
11	48	0	0	96
12	128	320	128	1,152
13	1,024	1,024	2,560	9,216
14	20,768	8,912	8,480	76,320
15	67,072	118,528	69,952	511,104
16	422,912	410,624	727,808	3,122,688
17	4,330,112	2,707,232	2,605,712	19,286,112
18	15,141,696	22,959,360	16,147,968	108,498,048
19	85,099,776	83,415,552	125,783,808	588,598,272
20	654,746,480	480,181,328	467,512,496	3,204,880,608
21	2.386139E+	3.311441E+0	2.565512E+0	1.652618E+1
22	1.249009E+	1.235346E+1	1.699009E+1	8.366729E+1
23	8.200810E+	6.561595E+1	6.438249E+1	4.240131E+1
24	3.058337E+	4.008699E+1	3.297290E+1	2.072865E+1
25	1.525479E+	1.518219E+1	1.969322E+1	1.002604E+1

Figure 4: Distribution of ORFs over reading frames

Similarly, the asymmetry of ORFs between RFs 1 and 3, and 4 and 6, is also unexpected. We suspect that this is influenced by asymmetries in the stop sequences, such that the [G+C] content of the two DNA strands will differ from each other. Although bases used in the skeleton completions in the webapp algorithm are selected at random, the fraction of completions with [G+C] < 0.4 for the dsDNA fluctuates over smaller values of L.

5.2 Web Application

A webapp that generates dsDNA sequence exemplars that satisfy the "5&1" condition for L ≤ 100 is available at [http://www.ucs.mun.ca/~donald/orf/biocomp/]
The Python implementation was converted into a webapp with the use of HTML/CSS and JavaScript so as to allow the random ORF generator to run as a self-contained, client-side application inside a web browser. Once the page is downloaded, no further communication with a web server is necessary. Although the web application does not require a separate plug-in, JavaScript must be enabled in the web browser used to run the application.

The webapp (Fig. 5) displays a color-coded dsDNA sequence, with the top strand oriented left–to-right in the 5'➔3' direction, and reading frames (RFs) 1-3 commencing at the 1st, 2nd, and 3rd bases, respectively. The lower DNA strand is oriented right-to-left in the anti-parallel, 5'➔3' direction, and RFs 4-6 commence at the 1st, 2nd, and 3rd read from the right. The ORF is highlighted. The conventional IUPAC single-letter abbreviations for amino acids are centered over the middle base of the triplet; stop codons are indicated by asterisks (*).

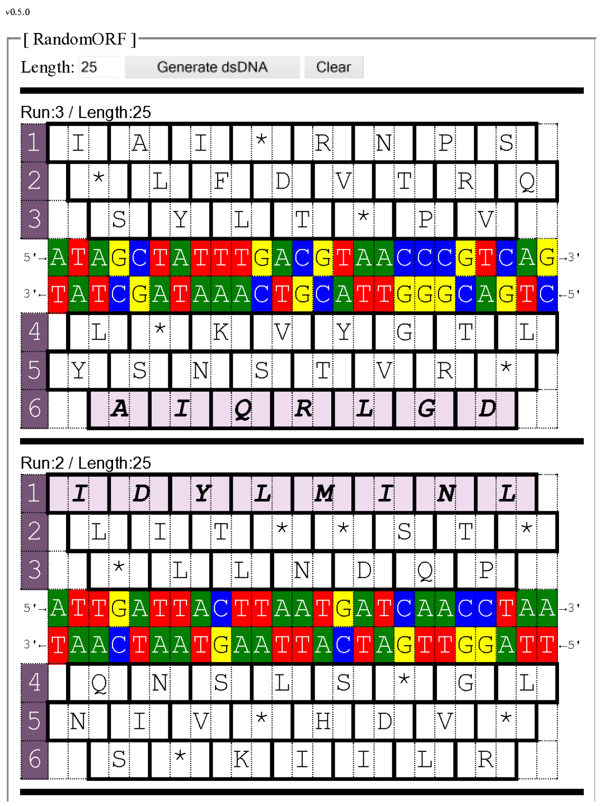

Figure 5: Screen capture of webapp. 5'➔3' ORFs in frames 1 (bottom) and 6 (top) run left-to-right and right-to-left, and encode polypeptides ***N***-IDYLMINL-***C*** and ***N***-DGLRQIA-***C***, respectively, where **N** and ***C*** are the amino and carboxyl termini, respectively.

6 ORF exemplars in genetic and bioinformatic education

The standard “Central Dogma” rubric for extraction of information from DNA (DNA makes RNA makes Protein) requires identification of a 5’➔3’ ORF in one strand of a dsDNA molecule, use of the antiparallel, complementary DNA strand as a template to transcribe an antiparallel, complementary 5’➔3’ mRNA, and “translation” of the mRNA codons 5’➔3’ into an amino acid polypeptide, by means of the Genetic Code.

We have found it more useful to demonstrate and emphasize the equivalence of the DNA sense strand to the mRNA, which are oriented in the same and have identical sequences except for substitution of U in the mRNA for T in the DNA. It is then more obvious that the polypeptide may be inferred logically from the DNA sense strand, with mental substitution of U for T in reading the code table. This is of course the same logic used by standard computer programs (e.g., Sequencher) to display an amino acid sequence aligned left-to-right from the input of a co-linear, single-stranded DNA sequence. The webapp presented here also introduces the information content of the dsDNA molecule, and reinforces the bioinformatic logic of data mining in a way that the standard rubric does not.

We provide a more complete discussion of the pedagogical applications of the web application in [5]. The webapp may also be a useful research tool. We are, for example, exploring the occurrence of randomly-generated medium-length exemplars in real-life data as periodic ‘punctuation’ that could ensure that closed frames stay closed.

7 Acknowledgements

SM Carr and HT Wareham were supported by NSERC Discovery Grants during the preparation of this MS. We thank Terra Nova Genomics, Inc., for hosting the web application. SMC dedicates this paper to Prof. William D. Stansfield of Cal Poly, San Luis Obispo, in recognition of his long service in genetics education.

8 References

[1] H. Judson, The Eighth Day of Creation, 2nd ed. Cold Spring Laboratories, Cold Spring Harbor, New York, 1996.

[2] F. H. C. Crick, The genetic code, yesterday, today, and tomorrow. Cold Spring Harbor Symposia in Quantitative Biology 31 (1966) 3-9.

[3] M. Nirenberg, P. Leder, M. Bernfield, R. Brimacombe, J. Trupin, F. Rottman, C. O’Neal, RNA codewords and protein synthesis VII. On the general nature of the RNA code. Proceedings of the National Academy of Science (USA) 53 (1965) 1161-1168.

[4] M. Nirenberg, T. Caskey, R. Marshall, R. Brimacombe, D. Kellogg, et al., The RNA code and protein synthesis. Cold Spring Harbor Symposia in Quantitative Biology 31 (1966) 11-24.

[5] S. M. Carr, D. Craig, and H. T. Wareham. A web application for generation of DNA sequence exemplars with open and closed reading frames in genetics and bioinformatics education. CBE – Life Sciences Education, in press (2014)