Algorithm for Coding DNA Sequences into "Spectrum-like" and "Zigzag" Representations

Jure Zupan and Milan Randic

J. Chem. Inf. Model. 2005, 45, 309-313

Summary by Elizabeth Doman

The authors of this paper present an algorithm for representing long strings of building blocks. The building blocks could be, for example, the 4 DNA bases (A- adenine, C- cytosine, T- thymine, G- guanidine), the 20 natural amino acids (from Alanine to Valine), or all 64 base triplets (from AAA to TTT), making up a long sequence of DNA. Their focus is on creating a more visually suitable representation of a given DNA sequence, while at the same time retaining as much of the original information as possible. The algorithm presented here is capable of calculating a 1-D, 2-D, or 3-D representation of a given sequence of DNA. In particular, these representations are reversible, or invertible, which means that the entire representation, and thus the entire DNA sequence, can be reproduced from knowing the coordinates of the last point only. This implies that the information contained in an entire sequence of DNA could be stored as just one single coordinate point (one number in 1-D, two numbers in 2-D, or three in 3-D). The representation is so called because a "spectrum" of numbers is created by assigning each building block to a distinct point in space. When the consecutive numbers of the "spectrum" are plotted, their pathway appears to "zigzag" through space.

This paper extends the work of Jeffery (1990, see reference in article) from 2-D representations to both 1-D and 3-D. Jeffery represented a entire DNA sequence, having 10,000 to 100,000 nucleic acids, in a single planar square. He did this by assigning each of the four corners, (-1,-1), (-1,1), (1,1), and (1,-1), to one of the four nucleic acids, A,T, G and C, respectively. Given a DNA sequence, start in the center, (0,0), and move halfway to the corner assigned to the first nucleic acid in the sequence to be coded. From this point, continue to the halfway point between it and the corner assigned to the second nucleic acid in the sequence. Continuing in this way, "zigzagging" through space, each base in the sequence will be assigned a unique point in space. This assignment contains information describing both the identity and location in the sequence of each base. In the end, the sequence of DNA will be represented by a "spectrum" of points located in the unit square.

Clearly, the assignment of the four bases to one of the corners of the square is arbitrary, and leads to 24 possible different coding schemes. Four of these can be mapped onto other ones by rotation, producing only 6 different possible maps. In turn, three of these can be mapped onto another one by reflection, leaving only 3 distinctly different representations. In particular, these three curves are associated with the diagonal assignments of A-G, A-T, and A-C.

The work of this paper is in extending the code from 2-D to both 3-D and 1-D. The extensions are not difficult to make, and follow the same framework as the above described 2-D representation. The 3-D extension is done by using the four corners of a tetrahedron, (1,-1,-1), (-1,1,-1), (-1,-1,1), and (1,1,1). In this case, only two different variations of the code of the sequence can be obtained.

In 1-D, only the x coordinates are used. For example, $x = -1$ would be assigned to both A and C, while $x = 1$ would be assigned to both T and G. This leaves no way to distinguish between A and C or T and G during the decoding process, creating the inherent loss of very important information. This problem will be addressed in later paragraphs.

The algorithm for coding is really quite simple, and rather beautiful. Denote the j_{th} unit of the N unit long DNA sequence by $seq(j)$. Then, $S(x_j, y_j, z_j)$, the j^{th} point of the 3-D "zigzag" or "spectrum-like" representation, is obtained from its predecessor according to the recursive relationship,

$$S(x_{j+1}, y_{j+1}, z_{j+1}) = \frac{S(x_j, y_j, z_j) + S(x_{seq(j+1)}, y_{seq(j+1)}, z_{seq(j+1)})}{2} \tag{1}$$

in 2-D by,

$$S(x_{j+1}, y_{j+1}) = \frac{S(x_j, y_j) + S(x_{seq(j+1)}, y_{seq(j+1)}, z_{seq(j+1)})}{2} \tag{2}$$

or in 1-D by,

$$S(x_{j+1}) = \frac{S(x_j) + S(x_{seq(j)})}{2} \tag{3}$$

where $S(x_0, y_0, z_0) = (0, 0, 0)$ in 3-D, $S(x_0, y_0) = (0, 0)$ in 2-D, or $S(x_0) = 0$ in 1-D.

The notable aspect of the relation is that, in both 2-D and 3-D, the recursive relationship is invertible. In other words, if the coordinate of the last point in the representation is known, the entire representation can be reproduced, and hence the DNA sequence recovered. Therefore, an entire DNA sequence can be described by just the very last point in the representation.

Given a point in the representation, the base which is describes is determined by the location of the point with respect to all the corners (or end points in 1-D). The closest corner to which the point is located determines what base that point represents. For example, in 2-D, if the the bottom left corner (-1,-1) is assigned to A, and the point of interest is located in the bottom left quadrant of the plane, then that point represents the base A.

In the forward coding process, the current coordinate value is halved to define the next coordinate point. Therefore, given a current point, the previous point can be determined by taking the distance between that point and the closest corner (or end point in 1-D) and doubling it symmetrically over that current point. In this way, given the last single point in the representation, we can work backwards to obtain the entire representation, and hence the DNA sequence which it describes.

However, the information loss in the 1-D problem makes it impossible to invert. One way to resolve this problem is by further assigning each point a black/white label (or a binary digit ”0” or ”1”). In this way, one could label the A and T bases white (or ”1”), and the C and G bases black (or ”0”). This ensures that the 1-D representation is reversible. Another way to overcome the loss of information in the 1-D case is to expand the positions of the bases to different numbers on the x-axis. For example x= -2, -1, 1, and 2 could be assigned to A, C, G, and T respectively, with $x = 0$ as a starting point. In this case, the representation would still be described by the same recursive definition as before, but would now be invertible.

In general, the number of basic units of a sequence in not limited to four, but any finite number of building blocks could be used. The only requirement is that the 1-D, 2-D, or 3-D coordinates denoting the positions of the building blocks be known and fixed. For example, the 64 base triplets (AAA, AAC, ..., TTT) could be used as building blocks to define a 1-D representation of a DNA sequence. The triplets positions on the x axis could be defined by assigning $x = -32$ to AAA, and $x = -31$ to AAC, and so on, at last assigning $x = +32$ to TTT. This is just one example that might work nicely, and there are many other distributions that might be interesting for different reasons.

In the paper, the algorithm is illustrated for the first 10 bases of the first exon in the human β-globine gene. Along with a table of the coordinates of consecutive bases, a visual representation of the sequence is shown in 1-D, 2-D, and 3-D. The visual graphs of the ”spectrum-like” representation do indeed follow a "zigzag" pathway through space (see figures in article).

If a bunch of sequences are represented in this "spectrum-like" way, the coordinates can be compared to one another (for example, using the Euclidean distance as a measure of similarity). The representations can then be hierarchically clustered into a dendrogram. This technique is illustrated in the paper, using both the 2 and 4 point 1-D coding scheme, on the first exon of the β-globine gene for 10 different species. The results are similar in both cases, and accurately organize the genes into clusters of similarity.

The algorithm presented in this paper has many benefits. It transforms any DNA sequence into a 1-D, 2-D, or 3-D visual graph that seems to "zigzag" through space. It assigns each base (or whatever building block you choose) in the sequence to a distinct point in space, giving the sequence a "spectrum-like" representation. An entire representation can be reproduced by knowing only the single last point of the representation (assuming the assignments of the bases to given corner points are known). This implies that huge amounts of data (DNA sequences of 100,000 base pairs) can be represented by a single point, assuming that enough digits are used to accurately reproduce the data. Furthermore, the algorithm is not restricted to using only 4 building blocks, and is capable of transforming sequences composed of any number of units. Along with such applications as hierarchical clustering, the consequences of this type of representation could be widespread. It is a very clever and simple way to store the complicated information contained in a long sequence of DNA- pretty neat!