Computational and Structural Biotechnology Journal 19 (2021) 6263–6271

COMPUTATIONAL AND STRUCTURAL BIOTECHNOLOGY JOURNAL

journal homepage: www.elsevier.com/locate/csbj

Chaos game representation and its applications in bioinformatics

Hannah Franziska Löchel, Dominik Heider *

Department of Mathematics and Computer Science, University of Marburg, Hans-Meerwein-Str. 6, D-35032 Marburg, Germany

ARTICLE INFO

Article history:
Received 26 August 2021
Received in revised form 4 November 2021
Accepted 5 November 2021
Available online 10 November 2021

Keywords:
Chaos game representation
Bioinformatics
Sequence analysis
Alignment-free sequence comparison
DNA and protein encoding
Machine learning

ABSTRACT

Chaos game representation (CGR), a milestone in graphical bioinformatics, has become a powerful tool regarding alignment-free sequence comparison and feature encoding for machine learning. The algorithm maps a sequence to 2-dimensional space, while an extension of the CGR, the so-called frequency matrix representation (FCGR), transforms sequences of different lengths into equal-sized images or matrices. The CGR is a generalized Markov chain and includes various properties, which allow a unique representation of a sequence. Therefore, it has a broad spectrum of applications in bioinformatics, such as sequence comparison and phylogenetic analysis and as an encoding of sequences for machine learning. This review introduces the construction of CGRs and FCGRs, their applications on DNA and proteins, and gives an overview of recent applications and progress in bioinformatics.

Contents

1. Introduction

The chaos game algorithm has been developed by Barnsley [1] to construct fractals based on random input and was later extended to DNA as input by Jeffrey [2]. The results are called chaos game representations (CGR). With the advent of fractal geometry by Mandelbrot in the 1970s [3], multiple algorithms emerged to construct fractals. Fractals are recursive, scale-invariant patterns and their dimension is a fraction [4,5]. Different algorithms exist to construct fractals, such as L-Systems [6], and Kronecker powers [7].

CGR is considered as a milestone in graphical bioinformatics [8,9]. The branch of graphical bioinformatics addresses graphical representations of sequences as a mathematical invariant of the sequences [8]. Thus, multiple approaches exist to visually and numerically represent biological sequences to a broad array of

* Corresponding author.

https://doi.org/10.1016/j.csbj.2021.11.008

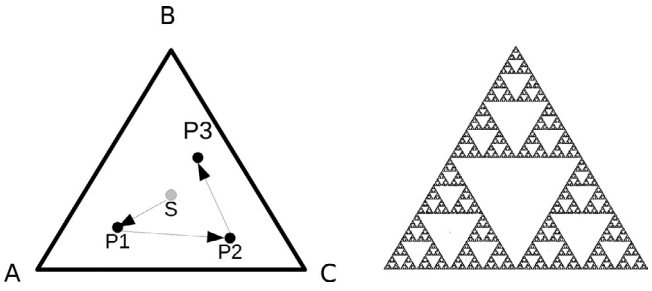

Fig. 1. Chaos Game Algorithm.

applications, such as sequence comparison, visualization, and as an encoding for machine learning. The underlining concept of the CGR algorithm is to map a sequence, i.e., a 1D representation to a higher dimensional space, typically to the 2D space [1]. It was originally developed to construct the Sierpinski triangle. To this end, numbers from one to three are assigned on the vertices of a triangle (see Fig. 1). Based on a randomly selected start point (S), a vertex is randomly chosen (V1), and a point P1 is drawn in half the distance to the vertex V1. This process is repeated, with P1 as the new starting point. The second point (P2) is drawn at half the way to the second randomly selected vertex (V2). By repeating this algorithm, the Sierpinski triangle emerges (see Fig. 1 right). The CGR is an iterative function system (IFS) [1], which derives the process from set theory [10].

In 1990, Jeffrey [2] proposed the application of the CGR algorithm to DNA, leading to broad applications in bioinformatics. Others all extended CGR to proteins, because of its unique properties:

- a sequence is represented as a unique pattern
- a sequence is mapped to unique coordinates
- a CGR maps all possible sequences in all possible lengths to 2D or 3D space
- a single coordinate encodes the complete sequence input
- the starting point does barely influence the outcome

Because of its properties, CGR has already been used, for instance, in alignment-free sequence comparison, phylogeny, and as an encoding for machine learning, and has also a huge potential for future applications in bioinformatics. This review addresses the concept of CGR and its applications and potential in multiple directions in bioinformatics.

2. Chaos game for DNA

Jeffrey [2] was the first to apply CGR on DNA. Instead of using a triangle, the CGR was based on a square, with the four vertices representing the four nucleotides, adenine (A), cytosine (C), guanine (G), and thymine (T). Fig. 2 illustrates the CGR for DNA.

Jeffrey [2] observed that for random sequences no visible patterns emerge (see Fig. 3 upper left). However, for the CGR on a DNA sequence fractal, non-random patterns emerge. Fig. 3 shows three examples of CGR patterns for different genomics sequences of organisms compared to a random sequence. This pattern is the attractor of the sequence [2], i.e., the points or areas to which a dynamical system converges to [11]. CGR was also used to examine visually the quality of random number generators. For random number generators of high quality, no visible patterns in the CGR will emerge [1,12,13]. This feature of CGR has led to two application directions, namely to tools for visual data analytics, e.g., to visualize and analyze pseudorandom number generators [12] and for alignment-free comparisons of sequences [9].

In Fig. 3, we assigned the nucleotides to the CGR coordinates as follows: A is assigned to (−1,1), T is assigned to (1,1), C is assigned to (−1,1), and G is assigned to (1,−1). Jeffrey [2] originally used a different notation with a CGR spanning from (0,0) to (1,1), with a different assignment of the vertices. A diagonal order of purines (and likewise pyrimidines) is widely used in the literature (A-C/G-T). However, a horizontal order of purines and pyrimidines can also be found (A-T/G-C) in some studies. There are $\frac{(n-1)!}{2}$ ways to assign the nucleotides to the vertices, i.e., three different orientations plus rotations and mirroring. Burma et al. [15] used these three orientations to analyze different genomes. They could show that, depending on the orientation of the vertices, different visual patterns emerged for the genomes, which allows a user to visualize different motifs or nucleotide concentrations in the CGR. Barnsley [1] describes the CGR as an iterated function system (IFS), which is based on set theory [10]. In the recent years, a more compact notation based on a geometrical approach has been established [9]:

$$P_i^j = P_{i-1}^j + sf(V_{i-1}^j - P_{i-1}^j) \quad (1)$$

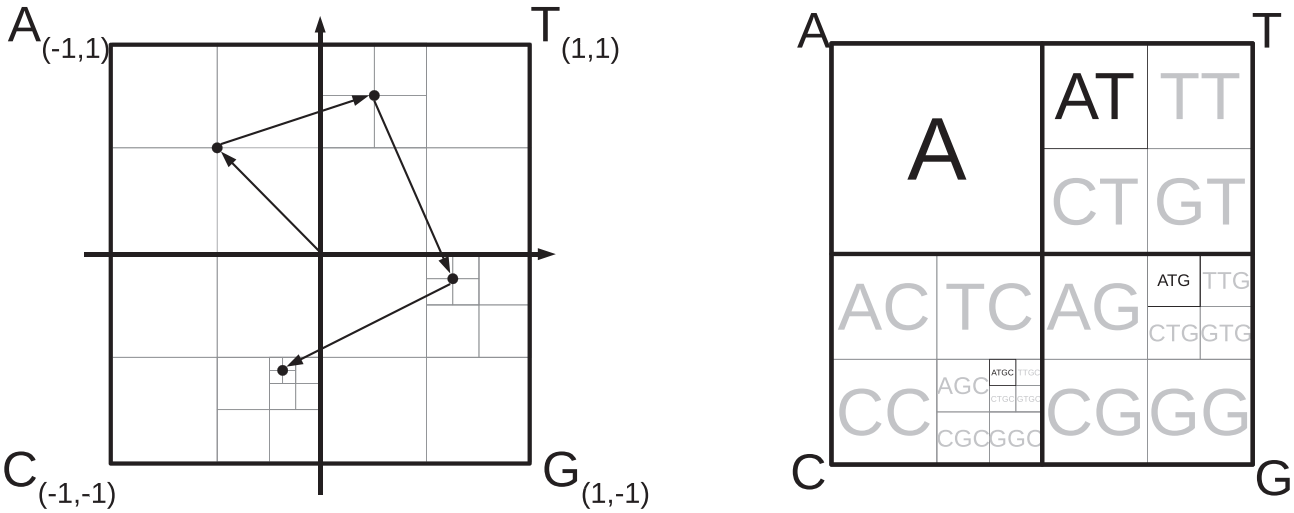

Fig. 2. Chaos Game Representation and algorithm for DNA. Left: CGR algorithm for four vertices with corresponding labels, i.e., A, C, G, and T, as well as corresponding coordinates. In CGR the center has the coordinates (0,0) and the CGR spans from (−1,−1) to (1,1). Right: Division of the CGR space due to the iterative process.

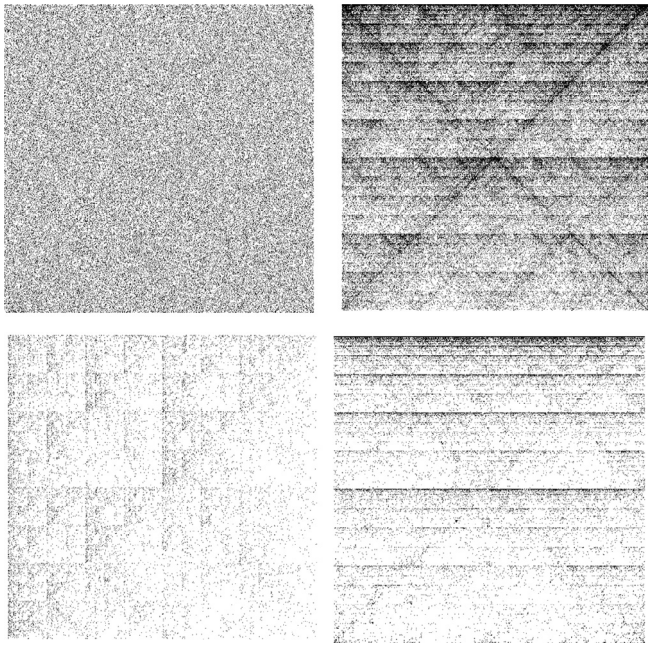

Fig. 3. Chaos Game Representation, order of vertices: A-T/C-G, created with R package kaos [14]. Upper left: random sequence. Upper right: *Hoya carnosa* isolate SZ708 chloroplast, complete genome GenBank: MN781974.1. Lower left: *Homo sapiens neanderthalensis* mitochondrion, complete genome GenBank: KY751400.2. Lower right: *Saccharomyces cerevisiae* S288c mitochondrion, complete genome NCBI Reference Sequence: NC_001224.1.

P_0^j: starting point; either randomly chosen or pre-defined.
j: dimension of the CGR, i.e., 2 for DNA
i: position of the sequence S
sf: scaling factor; 0.5 for DNA
V_i^0 = 1 if S_i T or G, else V_i^0 = −1
V_i^1 = 1 if S_i A or T, else V_i^1 = −1

Different adaptations to adjust the CGR have been developed. For instance, it is possible to change the order and number of the vertices or their coordinates. Moreover, the CGR is not limited to a 2D representation. It is also possible to extend the CGR to a multi-dimensional representation. Furthermore, the scaling factor can be used and adjust the appearance of the CGR.

Jeffrey [2] showed that clear, visible patterns only emerge for vertices $\geqslant 7$ with a scaling factor of 0.5 (see Fig. 4).

Besides those properties mentioned already, the CGR has some additional unique properties. The CGR is a representation of all possible sequences in any length in a continuous space. It can be considered as a generalization of a Markov model (the next state depends on the current state [16]), and the complete sequence can be reconstructed solely from the last coordinates of the CGR [17]. Burma et al. [15] analyzed the CGR for DNA and identified the following additional properties of CGR for DNA:

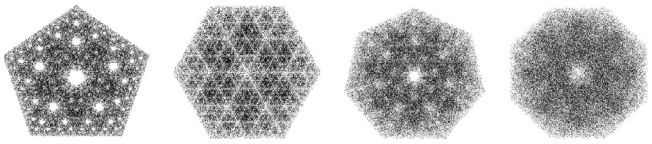

Fig. 4. CGR with a scaling factor of 0.5 for different numbers of vertices (5 to 8 from left to right), created with R package kaos [14]. With a random input sequence the possible space within the CGR is covered. A scaling factor of 0.5 with more than 4 vertices leads to overlapping areas in the CGRs.

- The patterns in the CGRs of long DNA sequences indicate homology between two sequences.
- The CGR allows the identification of repetitive sequences and their frequencies.
- The CGR can be used to identify absent or low frequent subsequences.
- Shorter subsequences show the same characteristic patterns as the full DNA sequence (e.g., genomes).

3. Generalized Chaos game representation

While the original CGR approach by Jeffrey [2] was developed for DNA, different approaches for proteins have been developed later. After the original algorithm based on a Sierpinski triangle was adapted to a square to take the four nucleotides of DNA into account by Jeffrey [2], efforts have been made to extend the algorithm for the 20 amino acids to build CGRs of proteins. For more than four vertices, CGRs get noisy and the points are overlapping (see Fig. 4). To this end Fiser et al. [18] proposed a so-called Sierpinski n-gon, polyflakes, or n-flakes [19] for proteins. In this approach, the scaling factors are adjusted to avoid overlapping. The scaling factor for n-flakes is also known as the kissing number [20]. The coordinates of the vertices for a polyflake can be calculated using the following equation [14]:

$$x[i] = r \cdot sin\left(\frac{2\pi i}{n} + \theta\right)$$
$$y[i] = r \cdot cos\left(\frac{2\pi i}{n} + \theta\right) \quad (2)$$

$x[i]$: x-coordinate for vertex i
$y[i]$: y-coordinate for vertex i
r: radius.
i: vertex.
n: number of vertices.
θ: angel of orientation.

Fiser et al. [18] proposed an equation, to calculate the kissing number for polyflakes. However, this equation is not suitable for four vertices. In this case, the equation generated a looser packing with a scaling factor of 0.54 for four vertices [21]. Fig. 5 shows the impact of the different scaling factors on a squared CGR. Almeida and Vinga [21] proposed a different equation that can produce a scaling factor of 0.5 for four vertices. An alternative version with a compacter notion can be found in Strichartz [22], applied for CGR in Lochel et al. [14]:

$$sf = 1 - \frac{sin\left(\frac{\pi}{n}\right)}{sin\left(\frac{\pi}{n}\right) + sin\left(\frac{\pi}{n} + \frac{2\pi m}{n}\right)}$$
$$m = \left\lfloor \frac{n}{4} \right\rfloor \quad (3)$$
$\lfloor\rfloor floorfunction$

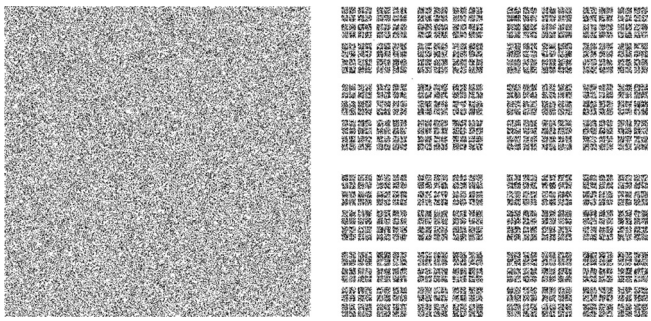

Fig. 5. Impact of the scaling factor for a CGR (with four vertices). Created with the R package kaos [14]. Left: $sf = 0.5$ based on the original CGR [1,2]. Right: $sf = 0.54$ as an result of the equation for the kissing number in Fiser et al. [18].

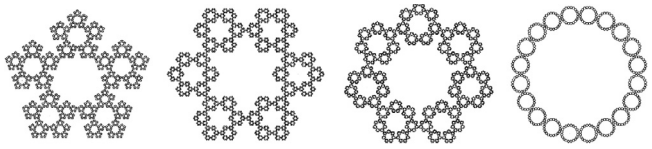

Fig. 6. CGR polyflakes with the kissing number as scaling factor and different numbers of vertices (from left to right: 5, 6, 7, and 20). Created with the R package kaos [14].

with n the number of vertices.

Fig. 6 shows the polyflakes based on the CGR with random numbers as input and different number of vertices (i.e., 5, 6, 7, and 20). The CGR with 20 vertices can, for instance, be used for proteins with the twenty proteinogenic amino acids as vertices and with the kissing number as the scaling factor. The use of polyflakes enabled the encoding of protein sequences into the CGR and has been applied in different studies [18,21,14].

Similar to CGRs for DNA, there are also $\frac{(n-1)!}{2}$ ways to assign amino acids to the edges of the 20 vertices of the polyflakes. The algorithm maps the sequence to 2D space, and a loss-less reversal of the algorithm to the original sequence is possible. From a computing perspective, there is no difference in the CGRs with respect to the order of the edges. However, to compare different CGRs with each other in one classification task or phylogeny analysis, the order of the vertices have to be the same for all samples in the dataset.

However, alternative strategies have also been developed. For instance, Basu et al. [23] separated the twenty amino acids based on their properties into 12 groups and used a CGR with 12 vertices that has further been separated into a grid of 24 squares, as a representation of protein sequences.

4. Frequency matrix chaos game representation

While the original CGR uses exact coordinates for each point, an discretization called the frequency chaos game representation (FCGR) enabled a coarse-grained and less noisy CGR abstraction for sequences. FCGR is based on counting the points of the CGR based on a pre-defined grid. In Fig. 7 the CGR is separated by a grid (here 8×8 cells), and the number of points in each cell is counted. This procedure results in a matrix representing the frequency of k-mers (here 3-mers), and thus a visualization in, for instance, grayscale.

The FCGR, similar to the original CGR, enables the identification of motifs or missing motifs in a given sequence [15,13]. Additionally, FCGR allows the visualization of homology between different genomes as a coarse-grained grayscale visualization Burma et al. [15]. The plotting of the CGR on a computer screen inevitably leads to a compression of the image. Thus, Burma et al. [15] used this circumstance on purpose, by separating the CGR with a grid in resolutions of $2^k \times 2^k$, in order to count the frequencies of different k-mers. They also assigned different colors to the frequencies and could demonstrate that FCGR can be used to visualize homology between genomes. Moreover, they found that patterns in CGR are repeated (i.e., that CGR are fractals), and that underrepresented sequences are shown as "white holes" in the image. Additionally, they proposed to use FCGR to identify repetitive elements in the genome or gene duplications. In the same year, Hill et al. [24] and Huynen et al. [25] used CGR to compare frequencies of dinucleotides for human globin and alcohol dehydrogenase genes [24] and GC content in histones [25]. Huynen et al. [26] further analyzed FCGRs and demonstrated that the CGR can be considered as a first-order-Markov chain. Oliver et al. [27] generated entropic profiles by increasing the resolution of the CGR and calculating the entropy of the histograms (CGR as frequency plots), which can be used to clearly show differences between random sequences and genomes.

The above-described approaches refer to FCGRs based on grids in $2^k \times 2^k$ and, therefore, are based on representations of k-mers.

In contrast, Almeida et al. [17] evaluated different non-integer resolutions. Their approach results in a division of the CGR to non-integer oligonucleotides (i.e., by division of the grid, where k is not an integer). The number of quadrants in a grid of an FCGR can be calculated by Almeida et al. [17]:

$$q = 2^{2 \cdot k} \tag{4}$$

q: number of quadrants
k: k-mer size

For example, to calculate the length of a k-mer of a 10 x 10 grid, the equation can be rearranged, resulting in a k-mer size of 3.32 [17]:

$$k = \frac{log_2(q)}{2} \tag{5}$$

Almeida et al. [17] introduced the usage of non-integer resolution to address redundancy in genomic structures, which is a major feature in genomic structures, e.g., for amino acid encoding. Almeida et al. [17] could show that CGR is a generalized Markov-Chain and that Markov-models are particular cases of CGR models. The extension of a CGR to FCGR enabled new methods for sequence comparison and phylogeny. Additionally, Almeida et al. [17] sug-

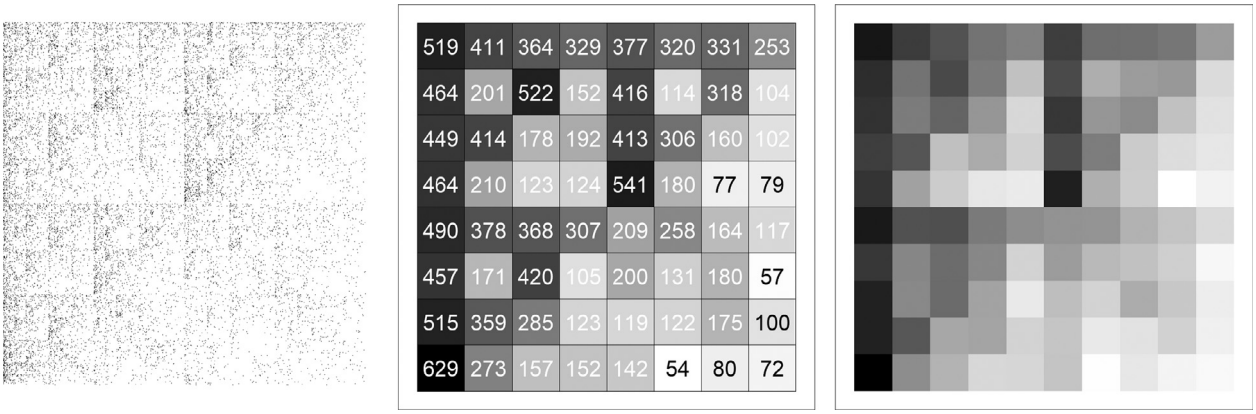

Fig. 7. Complete mitochondrial genome sequence of the Tyrolean Iceman (GenBank:EU810403.1). From left to right: CGR; FCGR with count matrix; FCGR as grayscale visualization. Created with R package kaos [14].

gested to calculate the global distance (d) between two FCGRs based on a weighted Pearson correlation coefficient (rw), using the following equation:

$$
\begin{aligned}
nw &= \sum_{i=1}^{k} x_i \cdot y_i \\
\bar{x}w &= \frac{\sum_{i=1}^{k} x_i^2 \cdot y_i}{nw} \\
\bar{y}w &= \frac{\sum_{i=1}^{k} x_i \cdot y_i^2}{nw} \\
sx &= \frac{\sum_{i=1}^{k} (x_i - \bar{x}w)^2 \cdot x_i \cdot y_i}{nw} \\
sy &= \frac{\sum_{i=1}^{k} (y_i - \bar{y}w)^2 \cdot x_i \cdot y_i}{nw} \\
rw_{x,y} &= \frac{\sum_{i=1}^{k} \frac{x_i - \bar{x}w}{\sqrt{sx}} \frac{y_i - \bar{y}w}{\sqrt{sy}} \cdot x_i \cdot y_i}{nw} \\
d &= 1 - rw
\end{aligned}
\tag{6}
$$

nw: compounded frequency.
x_i, y_i: FCGR quadrant.
rw_{xy}: weighted correlation coefficient.
d: distance.

For the comparison of two sequences based on their global distance d, their FCGRs have to be in the same resolution. d is a number between 0 and 2, while values greater than 1 indicate a negative correlation coefficient and 0 indicates similarities of the sequences [17].

For FCGRs of DNA, the subdivision into squares leads to a subdivision of k-mers. While for proteins, the subdivision of the 20-gons into squares does not. To this end, methods applied on DNA might perform better than on proteins. Moreover, while CGR is lossless, FCGR as a coarse-grained approach is considered to be lossy.

5. Extensions of CGR

Besides the generalized CGR and the FCGR, multiple other extensions of the CGR algorithm exist.

While the aforementioned approaches use CGR as a 2D representation of sequences (i.e., 1D representations), other approaches also use 3D or multi-dimensional CGR. While 2D and 3D representations can be plotted, higher dimensional representations, are hard to visualize. Nevertheless, multidimensional CGRs could also be used as a high-dimensional data representation for subsequent applications, e.g., machine learning, similar to other methods such as the PCA. Albeit FCGR is typically represented in grayscale (or colors), 3D representations have been proposed as well, e.g., by Korolev et al. [29], Solovyev [30], or Deschavanne et al. [31]. In these 3D CGRs, the number of k-mers is represented in a third dimension instead of a gray value (see Fig. 8). This approach can be used to visualize difference between different scales, however, it comes with the typical limitations of 3D representations, e.g., the possibility of perspective distortion.

Sun et al. [32] developed a three-dimensional CGR for proteins, in a regular dodecahedron, with promising results regarding protein classification and phylogenetic analysis.

Hao [33] introduced an alternative approach to construct and describe the patterns in CGR, based on Kronecker powers. Kronecker powers are matrix operations and in connection with DNA they can be applied as follows:

$$
\begin{bmatrix} A & T \\ G & C \end{bmatrix} \otimes \begin{bmatrix} A & T \\ G & C \end{bmatrix} = \begin{bmatrix} AA & TA & AT & TT \\ GA & CA & GT & CT \\ AG & TG & AC & TC \\ GG & CG & GC & CC \end{bmatrix} \tag{7}
$$

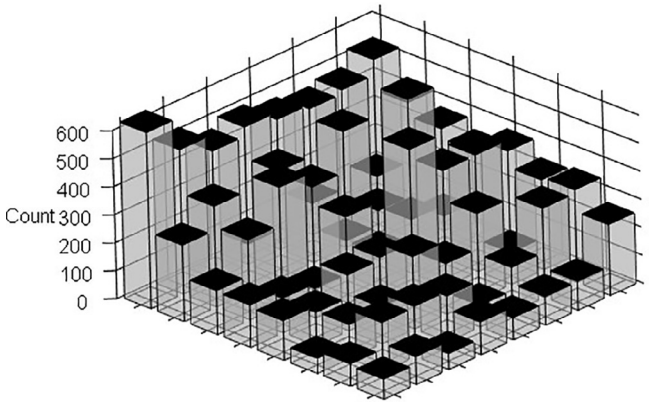

Fig. 8. Complete mitochondrial genome sequence of the Tyrolean Iceman (GenBank:EU810403.1), as 3-D FCGR created with R package kaos [14] and barplot3d [28].

This approach allows further investigations of the DNA sequences, e.g., regarding missing subsequences [34]. It can also be extended to a hypercube for any alphabet [35].

6. Applications in bioinformatics

Within the different extensions of the CGR algorithm, several applications in bioinformatics have been developed. The extension from DNA to larger alphabets also allowed the application of CGR for proteins as described above. By using the FCGR of DNA, sequences can be compared based on their image representation with different metrics. Thus, the CGR can be used for alignment-free sequence comparisons (see Zielezinski et al. [36] for a detailed review on alignment-free sequence comparisons), and therefore, for phylogenetic analyses, but also as an encoding method for machine learning applications. Alignment-free sequences comparisons have some advantages, for instance, the time complexity and computing power. For alignments, there are multiple underlying assumptions that all genes have the same arrangement in a genome, which is, for instance, not the case for viruses [36]. The advantage of FCGR-based methods for phylogeny is not so much accuracy but rather the speed and size of data possible to be handled. Thus, one can use an FCGR-based method on a relatively large data set to get a crude baseline, then use alignment-based methods on subsets of interest. While massive computational power is now commonly available, alignment-based methods eventually require too much computing power to handle, e.g., tens of thousands of genomes that would need a pairwise comparison. Also, CGR-based methods do not require that sequences have equal or similar lengths, which is necessary for certain analyses, e.g., machine learning.

Depending on the task, CGR/FCGR can be applied in different ways (shown in Fig. 9). In the first step, the algorithm produces x- and y- coordinates for each sign (here shown for DNA). These coordinates can either be plotted or further processed to an FCGR as a numerical matrix encoding. Additionally, a visualization of the matrix by gray-values is possible. For CGRs, the numerical encoding results in two numbers for each sign, so the length of the encoding depends on the input. While for FCGRs, the transformation to a matrix enables a fixed input dimension. For instance, this fixed input dimension makes FCGR attractive as a numerical encoding for machine learning and sequence comparison. These images can be used for classification or visualization. Numerical encodings have the advantage of avoiding information loss and

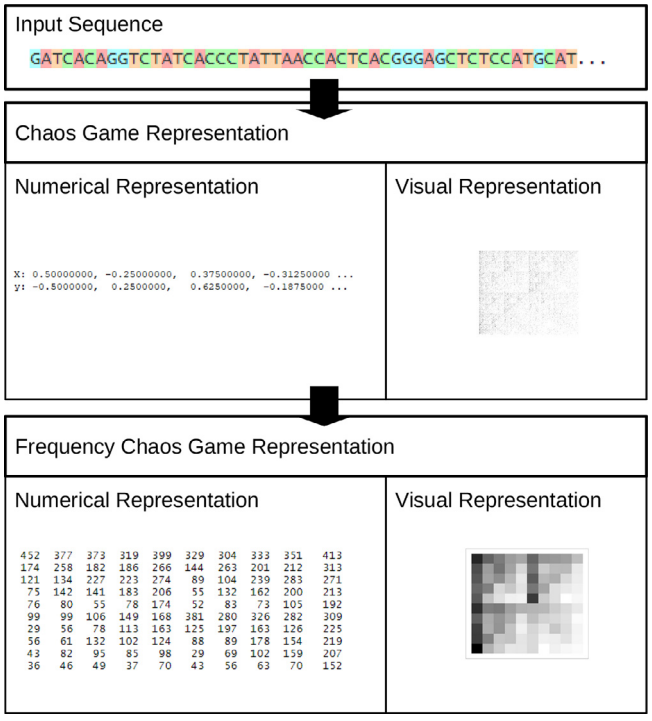

Fig. 9. Workflow of CGR based analysis. The input sequence (DNA, proteins, or any one-dimensional sequence) is encoded with the CGR algorithm to x- and y-coordinates (here shown for DNA). The coordinates can be transformed into an FCGR matrix as numerical encoding. Depending on the task and the chosen method, the numerical encoding can be used for applications in bioinformatics or visualized and then be further processed.

overhead of reading, writing, and storing images. The images provide a visualization and the adoption of image-processing methods. Nevertheless, the visualizations are not easy to interpret without prior knowledge of CGR. The visualization can be applied to show the absence or the accumulated occurrence of particular motifs and for visual sequence comparison.

6.1. Alignment free sequence comparison

Sequence comparisons based on alignments are widely used in bioinformatics, but have some disadvantages. For instance, they are computationally complex (typically quadratic runtime, i.e., $O(nm)$ with n and m the lengths of the two sequences to be compared), particularly for long sequences, and depend on multiple assumptions (e.g., gap penalties and substitution matrices) [36].

Jeffrey [2] showed that different and unique patterns in the CGR emerge from different sequences. In combination with the FCGR approach, the resulting matrices can be used to calculate these differences, as demonstrated by Almeida et al. [17] with Eq. 6. Other strategies exist, for instance, using the euclidean distance [31]. Deschavanne et al. [31] utilized the euclidean distance between two images based on k-mer FCGR to calculate phylogenetic distances between genomes. Accordingly, there are two possible ways to compare two sequences based on CGR, namely (i) the comparison of FCGRs in equal resolutions, and (ii) the direct comparison of the coordinates of the CGRs [37].

The comparison based on FCGRs is therefore based on the frequency of k-mers (or non-integer k-mers for particular resolutions of the grid), and has been used by, e.g., Hill et al. [24], Huynen et al. [25], and Deschavanne et al. [31]. Fig. 10 shows the FCGR of three mitochondrial genomes (human, Tyrolean Iceman, Neanderthals and chimpanzee) and the difference between these genomes to illustrate the genome comparison based on FCGR. We calculated the differences by subtracting the matrices from each other. While the higher frequencies in human are indicated in green, the iceman/neanderthals/chimpanzee are shown in pink. The FCGRs of the genomes look very similar at the first glance, however, a simple subtraction reveals significant differences. The difference-plot of the human and iceman genomes (both *homo sapiens*) has less differences compared to the difference-plot of the human and neanderthals and the interval of the differences is also smaller

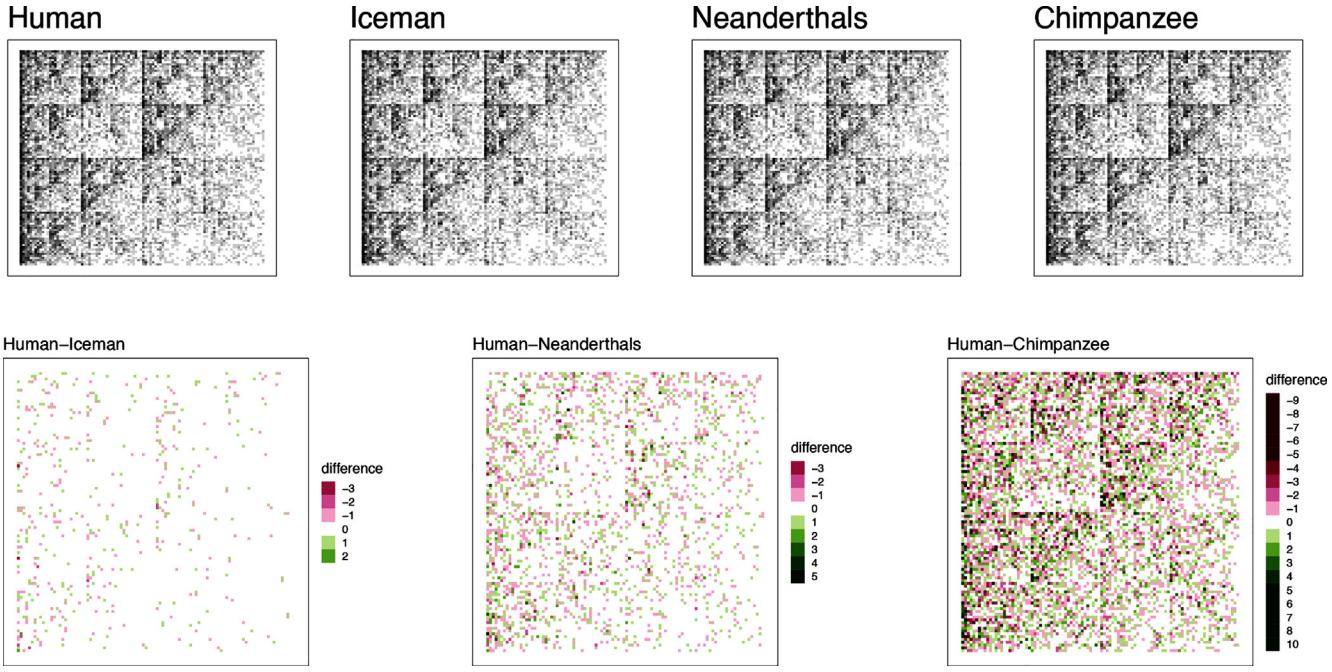

Fig. 10. Comparison of the FCGRs of complete mitochondrial genomes of Human (GenBank:FJ986465.1), the Tyrolean Iceman (GenBank:EU810403.1), Neanderthals (GenBank: KY751400.2) and chimpanzee (NCBI Reference Sequence: NC_001643.1). Top: FCGRs of the mitochondrial genomes. Bottom: Differences between the FCGRs.

([−3, 2] versus [−3, 5]). Both aspects clearly show that the mitochondrial genome of the Tyrolean Iceman is closer related to the human genome than the neanderthals' genome. The difference between human and chimpanzee has even a greater interval and more differences. Besides this visual approach, other methods exist to calculate the distance between two FCGRs, e.g., the Euclidean distance. Karamichalis et al. [38] compared six different distance metrics with respect to the construction of phylogenetic trees based on FCGRs, namely (i) the Structural Similarity Index (SSI), (ii) the descriptor distance, (iii) the Euclidean distance, (iv) the Manhattan distance, (v) the Pearson distance, and (vi) the approximated information distance. They demonstrated that the SSI and the descriptor distance show a better performance compared to the other metrics. Other alternatives have been developed in recent years for genome comparison based on the FCGR images [39], for instance, the multifractal analysis [40] or the discrete cosine transformation [41].

The sequence comparison based directly on the coordinates was proposed later by Joseph and Sasikumar [37]. To compare two sequences A and B, they used a $2^n \times 2^n$ FCGR of sequence B and aligned the coordinates of the CGR of sequence A. To this end, the alignment algorithm starts with the last coordinate of the last nucleotide of sequence A and selects the corresponding square in the grid of sequence B. Based on that principle, they calculated the distance of the points on a pre-defined metric and could deploy an algorithm based on CGR for local sequence alignments. Hoang et al. [42] developed a method for coordinate-based sequence comparison based on discrete Fourier-transformation (DFT). The underlining idea of this method is based on the generation of the CGRs of the set of sequences. Therefore, a DFT is performed on each CGR. The power spectra are then computed based on the DFT. The length of the spectra depends on the input sequence length, thus, Hoang et al. [42] applied an even scaling method. In the final step, they used the Unweighted Pair Group Method (UPGM) with the arithmetic mean to construct phylogenetic trees.

6.2. Encoding for machine learning

Within the scope of bioinformatics, multiple machine learning techniques, such as artificial neural networks, support-vector machines, or random forests, have been applied to sequence data (such as proteins and DNA) to predict, for instance, functional properties of the sequences [43]. The most common classification problem is binary classification, where a set of sequences is separated into two groups, e.g., positive and negative. For instance, for the prediction of resistance in pathogens [44–46] or peptide classification [47,48].

Almost all machine learning techniques have in common that they need a fixed input dimension. To this end, preprocessing methods have been developed [49], e.g., sparse encoding or interpolation [50].

CGR, especially FCGR, can also be used to create input data with a fixed input dimension. To this end, different strategies can be applied. The CGR or FCGR can be directly used as an input image for machine learning [51–53], as an extended natural vector of the image [39,32]), or in combination with singular value decomposition [54]. Moreover, the FCGR can also be used directly as an input matrix or vector for the machine learning model [55,14,56]). While the use of the images can lead to lossy compression, the representation by grayscale only allows values between 0 and 255. The choice of FCGR as an image encoding makes it mandatory to decide if the highest gray value is assigned to the highest frequency in the sequence or to the highest frequency in the dataset. By using the first approach, the FCGR of a sequence gets normalized to its sequence length. With the second approach, the differences between the frequencies of different samples is preserved. However, prior knowledge of the complete dataset is needed. Since the CGR can be applied for DNA and proteins, both approaches have been used in recent years. For instance, CGR-images as encoding has been used to predict coding/non-coding regions in mammals. Emam et al. [57] compared five different machine learning techniques: Naive Bayes, Logistic Regression, K-Nearest Neighbor, Perceptron, and support vector machines. The CGR coordinates themselves can also serve as an encoding for sequences with equal length (e.g., as applied in Hoang et al. [58] to predict splice sites). But the most frequent strategy is FCGR as encoding. Rizzo et al. [51] used FCGRs for taxonomic predictions of 16S ribosomal RNA. They trained a convolutional neural network on the FCGR images and reached very high accuracy. Löchel et al. [14] developed a resistance prediction model based on protein sequences. In this study, we compared the performance of the original scaling factor of 0.5 and the kissing number to produce polyflakes within FCGRs in different resolutions. Additionally, we compared the performance of different machine learning models (e.g., neural networks, random forests, and support vector machines), on FCGRs. The neural networks, trained on the FCGR polyflakes, showed a superior accuracy compared to the state-of-the-art models and encodings. Han et al. [55] used vectorized FCGRs for nucleosome positioning prediction. Zhou et al. [53] used convolutional neural networks to predict essential genes based on FCGR images. Dick and Green [52] used FCGRs of proteins of different organisms, either as polyflakes and the corresponding DNA sequences as FCGRs, to predict the source organism of the protein with deep convolutional neural networks. The results of these studies have in common that CGR/FCGR performed very well as encoding compared to existing alternatives, in most of the cases. However, the chosen resolution in FCGRs plays a significant role on subsequent prediction performance. At the moment, there are multiple open questions regarding FCGR as an encoding. For instance, Dick and Green [52] investigated the impact of the order of the vertices and showed that by changing the order of the vertices data augmentation might be feasible. They also compared the performance of FCGRs as encoding based on proteins and back-translated amino acids and found that proteins performed better within their task.The arrangement of the vertices could theoretically affect subsequent machine learning. Especially for FCGRs, the order might impact the result. For FCGRs in DNA, the CGR is a square, divided into smaller subsquares. An FCGR for proteins divides a 20-gon into smaller squares. The arrangement of the vertices might therefore have an impact on the encoding, which could, however, also be compensated by the resolution of the FCGR. Additionally, there are other questions regarding the resolution for the FCGRs. Overall, FCGR/CGR can be applied in different ways as encodings, either as an image or as a matrix/vector. Therefore it can be used with different machine learning techniques. The usage of the images enables image classifications. For this procedure, the FCGRs/CGRs have to be stored as images and re-read, which leads to additional computing time.

6.3. Comparison with other methods

The CGR/FCGR encodes sequences into a numerical encoding that can be visualized. These encodings can serve as input for bioinformatics analyses. From a visual perspective, CGRs/FCGRs are hard to read and prior knowledge is mandatory to understand the images. To this end, other graphical alternatives might be considered for visualization. The value of CGR/FCGR is the underlying numerical encoding. This numerical representation, as a mathematical invariant, transfers sequences into data structures for computational tasks. To this end, there are multiple layers to consider the performance and value of CGR/FCGR, namely (i) the visualiza-

tion, (ii) the performance on sequence comparisons, (iii) applications in phylogeny, (iv) as a numerical encoding for machine learning, and (v) as an image encoding for machine learning, e.g., deep learning.

As aforementioned, the visual aspects (i) of CGR/FCGR are only readable with prior knowledge. Other alternative fractal visualizations exist, for instance, the Hilbert curve[59]. A visual comparison based on FCGR (like in Fig. 10) is of some value and easier to understand without prior knowledge. FCGRs as visualization of k-mers can give some insights into the sequence. While the visualization on proteins is even harder to read, but can also be of some value for comparative tasks. Beyond the visual aspects, the true value of CGR/FCGR is the numerical representation that allows alignment-free sequence comparison (ii) and, as a result, phylogenetic analysis (iii). CGR/FCGR offers the opportunity for alignment-free sequence comparison. Most analyses are based on sequence alignments under the assumption of a linear arrangement of genes. Additionally, alignment-free comparisons have a lower run-time and some other advantages [36]. As we do not have a ground truth for phylogenetic analysis, the performance of different strategies is hard to validate. Additionally, the numerical representation for proteins/DNA/ or any sequence can be used for machine learning (iv), as well as the images(v). There are multiple encodings for Sequences (DNA, proteins, or text). For instance, one-hot encoding for DNA [60] or structure and sequence-based encodings for proteins [47]. While the performance of the chosen encoding depends on the dataset and the machine learning technique [48]. To this end, CGR and FCGR have many use cases and might be the better choice in some applications, but at this point, more research is needed.

7. Conclusion and future perspectives

It has been demonstrated that CGR is a powerful method in bioinformatics in many different applications. It can be applied for alignment-free sequence comparisons, and it has become a novel encoding technique for DNA and proteins for machine learning problems. For instance, multiple questions were addressed with this algorithm during the SARS-CoV-2 pandemic. For instance CGR and FCGR were used by Touati et al. [61] and [62], respectively, to build phylogenetic trees of coronaviruses. Sengupta et al. [63] applied machine learning for taxonomic alignment-free classification of viruses, based on CGR. Additionally, different directions can be identified within the CGR-research area, e.g., biometric analyses [64,65]. However, the CGR algorithm has also applications beyond bioinformatics or biometry, e.g., in the analysis of music and audio signals [66,67], authorship identification [68], economy [69], or optimization [70]. Due to the fact that the CGR can map sequences to 2D space, other applications are possible. For instance, Simplified Molecular Input Line Entry Specification (SMILES) or phonetic signals. In both cases, these encodings can be used either for machine learning or for similarity comparison.

At first glance, CGRs are fractal images of sequences. But the value of this representation is the numerical encoding to coordinates or as a matrix. These data structures offer further processing of the data. Multiple alternative graphical representations exist, which can also be used as numerical encodings [8].

The unique properties of the CGR algorithm, which allow recovery of the sequence from its last coordinates, have also found some applications regarding data encoding and compression. While data compression is not possible by using only the CGR algorithm, the combination of Huffman encoding and CGR allows compression of DNA [71,72]. Moreover, applications for encryption based on CGR exist [73]. The CGR is a powerful method with multiple applications. In bioinformatics for instance, it offers the opportunity for alignment-free sequence comparisons, phylogeny, and encoding for machine learning.

Funding

This work has been financially supported by the LOEWE program of the State of Hesse (Germany) in the MOSLA research cluster. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Authors contribution

HFL and DH developed the concept and wrote the manuscript. HFL generated the figures. DH supervised the study. All authors read and approved the final manuscript.

CRediT authorship contribution statement

Hannah Franziska Löchel: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft. **Dominik Heider:** Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Barnsley Michael F. Fractals Everywhere: New Edition. Dover Publications; 2012.
[2] Joel Jeffrey H. Chaos game representation of gene structure. Nucl Acids Res 1990;18(8):2163–70.
[3] Jones Huw. Fractals before mandelbrot a selective history. Fractals Chaos 1991:7–33.
[4] Mandelbrot Benoit B. The fractal geometry of nature/Revised and enlarged edition. whf; 1983.
[5] Jin Yi, Wu Ying, Li Hui, Zhao Mengyu, Pan Jienan. Definition of fractal topography to essential understanding of scale-invariance. Scientific Rep 2017;7(1):1–8.
[6] Prusinkiewicz Przemyslaw, Hanan James. Lindenmayer systems, fractals, and plants, vol. 79. Springer Science & Business Media; 2013.
[7] Shallit Jeffrey, Stolfi Jorge. Two methods for generating fractals. Comput Graph 1989;13(2):185–91.
[8] Randić Milan, Novič Marjana, Plavšić Dejan. Milestones in graphical bioinformatics. Int J Quantum Chem 2013;113(22):2413–46.
[9] Almeida Jonas S. Sequence analysis by iterated maps, a review. Briefings Bioinf 2014;15(3):369–75.
[10] George Winston Zobrist, Chaman Sabharwal. Progress in Computer Graphics. Intellect Books; 1992.
[11] Jopp Fred, Breckling Broder, Reuter Hauke. Modelling complex ecological dynamics. London: Springer; 2010.
[12] Mata-Toledo Ramón A, Willis Matthew A. Visualization of random sequences using the chaos game algorithm. J Syst Software 1997;39(1):3–6.
[13] Dutta Chitra, Das Jyotirmoy. Mathematical characterization of chaos game representation: New algorithms for nucleotide sequence analysis. J Mol Biol 1992;228(3):715–9.
[14] Löchel Hannah F, Eger Dominic, Sperlea Theodor, Heider Dominik. Deep learning on chaos game representation for proteins. Bioinformatics 2020;36(1):272–9.
[15] Burma Pradeep Kumar, Raj Alok, Deb Jayant K, Brahmachari Samir K. Genome analysis: a new approach for visualization of sequence organization in genomes. J Biosci 1992;17(4):395–411.
[16] Eddy Sean R. What is a hidden markov model? Nat Biotechnol 2004;22(10):1315–6.
[17] Almeida Jonas S, Carrico Joao A, Maretzek Antonio, Noble Peter A, Fletcher Madilyn. Analysis of genomic sequences by chaos game representation. Bioinformatics 2001;17(5):429–37.
[18] Fiser Andras, Tusnady Gabor E, Simon Istvan. Chaos game representation of protein structures. J Mol Graphics 1994;12(4):302–4.

[19] Jones Huw. Dürer, gaskets and barnsley's chaos game. Comput Graphics Forum 1990;9(4):327–32.
[20] Bates Tom et al. A generalization of the chaos game. In: Bridges 2019 Conference Proceedings. Tessellations Publishing; 2019. p. 139–46.
[21] Almeida Jonas S, Vinga Susana. Biological sequences as pictures–a generic two dimensional solution for iterated maps. BMC Bioinf 2009;10(1):1–7.
[22] Strichartz Robert S. Evaluating integrals using self-similarity. Am Math Monthly 2000;107(4):316–26.
[23] Basu Soumalee, Pan Archana, Dutta Chitra, Das Jyotirmoy. Chaos game representation of proteins. J Mol Graphics Model. 1997;15(5):279–89.
[24] Hill Kathleen A, Schisler Nicholas J, Singh Shiva M. Chaos game representation of coding regions of human globin genes and alcohol dehydrogenase genes of phylogenetically divergent species. J Mol Evol 1992;35(3):261–9.
[25] Huynen Martijn A, Konings Danielle AM, Hogeweg Pauline. Equal g and c contents in histone genes indicate selection pressures on mrna secondary structure. J Mol Evol 1992;34(4):280–91.
[26] Goldman Nick. Nucleotide, dinucleotide and trinucleotide frequencies explain patterns observed in chaos game representations of dna sequences. Nucl Acids Res 1993;21(10):2487–91.
[27] Oliver JL, Bernaola-Galvan P, Guerrero-Garcıa J, Roman-Roldan R. Entropic profiles of dna sequences through chaos-game-derived images. J Theor Biol 1993;160(4):457–70.
[28] Christopher Wardell. barplot3d: Create 3D Barplots, 2019. URL URL: https://CRAN.R-project.org/package=barplot3d. R package version 1.0.1.
[29] Korolev Sergey V, Tumanyan Vladimir G. Fractal dimensions of oligonucleotide compositions of dna sequences. In: Bioinformatics, Supercomputing and Complex Genome Analysis. World Scientific; 1993. p. 635–8.
[30] Solovyev Victor V, Lim Hwa A, Milanesi Luciano, Lawrence Charles. Application of fractal representation of genetic texts for recognition of genome functional and coding regions. In: Bioinformatics, Supercomputing and Complex Genome Analysis. World Scientific; 1993. p. 609–22.
[31] Deschavanne Patrick J, Giron Alain, Vilain Joseph, Fagot Guillaume, Fertil Bernard. Genomic signature: characterization and classification of species assessed by chaos game representation of sequences. Mol Biol Evol 1999;16(10):1391–9.
[32] Sun Zeju, Pei Shaojun, He Rong Lucy, Yau Stephen S-T. A novel numerical representation for proteins: Three-dimensional chaos game representation and its extended natural vector. Comput Struct Biotechnol J 2020;18:1904–13.
[33] Hao Bai-Lin. Fractals from genomes–exact solutions of a biology-inspired problem. Physica A 2000;282(1–2):225–46.
[34] Anitas Eugen Mircea. Small-angle scattering and multifractal analysis of dna sequences. Int J Mol Sci 2020;21(13):4651.
[35] Tiňo Peter. Multifractal properties of hao's geometric representations of dna sequences. Physica A 2002;304(3–4):480–94.
[36] Zielezinski Andrzej, Vinga Susana, Almeida Jonas, Karlowski Wojciech M. Alignment-free sequence comparison: benefits, applications, and tools. Genome Biol 2017;18(1):1–17.
[37] Joseph Jijoy, Sasikumar Roschen. Chaos game representation for comparison of whole genomes. BMC Bioinf 2006;7(1):1–10.
[38] Karamichalis Rallis, Kari Lila, Konstantinidis Stavros, Kopecki Steffen. An investigation into inter-and intragenomic variations of graphic genomic signatures. BMC Bioinf 2015;16(1):1–22.
[39] Pei Shaojun, Dong Wenhui, Chen Xiuqiong, He Rong Lucy, Yau Stephen S-T. Fast and accurate genome comparison using genome images: the extended natural vector method. Mol Phylogenetics Evol 2019;141:106633.
[40] Swain Martin T. Fast comparison of microbial genomes using the chaos games representation for metagenomic applications. Proc Comput Sci 2013;18:1372–81.
[41] Lichtblau Daniel. Alignment-free genomic sequence comparison using fcgr and signal processing. BMC Bioinf 2019;20(1):1–17.
[42] Hoang Tung, Yin Changchuan, Yau Stephen S-T. Numerical encoding of dna sequences by chaos game representation with application in similarity comparison. Genomics 2016;108(3–4):134–42.
[43] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, et al. Machine learning in bioinformatics. Briefings Bioinf 7(1); 2006: 86–112.
[44] Dominik Heider, Jan Nikolaj Dybowski, Christoph Wilms, and Daniel Hoffmann. A simple structure-based model for the prediction of hiv-1 co-receptor tropism. BioData Mining 7; 2014. ISSN 1756–0381. doi:10.1186/1756-0381-7-14.
[45] Löchel Hannah F, Riemenschneider Mona, Frishman Dmitrij, Heider Dominik. SCOTCH: subtype A coreceptor tropism classification in HIV-1. Bioinformatics 2018;34(15):2575–80. https://doi.org/10.1093/bioinformatics/bty170.
[46] Löchel Hannah F, Dominik Heider. Comparative analyses of error handling strategies for next-generation sequencing in precision medicine. Scientific Rep 10(1); 2020: 5750. ISSN 2045–2322. doi:10.1038/s41598-020-62675-8.
[47] Spänig Sebastian, Heider Dominik. Encodings and models for antimicrobial peptide classification for multi-resistant pathogens. BioData Mining 2019;12(1):29. https://doi.org/10.1186/s13040-019-0196-x.
[48] Sebastian Spänig, Siba Mohsen, Georges Hattab, Anne-Christin Hauschild, Dominik Heider. A large-scale comparative study on peptide encodings for biomedical classification. NAR Genomics Bioinf 3(2): lqab039; 2021. ISSN 2631–9268. doi:10.1093/nargab/lqab039.
[49] Liu Bin. Bioseq-analysis: a platform for dna, rna and protein sequence analysis based on machine learning approaches. Briefings Bioinf 2019;20(4):1280–94.
[50] Heider Dominik, Hoffmann Daniel. Interpol: An R package for preprocessing of protein sequences. BioData Mining 2011;4:16. https://doi.org/10.1186/1756-0381-4-16.
[51] Riccardo Rizzo, Antonino Fiannaca, Massimo La Rosa, Alfonso Urso. Classification experiments of dna sequences by using a deep neural network and chaos game representation. In Proceedings of the 17th International Conference on Computer Systems and Technologies 2016, pp. 222–228.
[52] Dick Kevin, Green James R. Chaos game representations & deep learning for proteome-wide protein prediction. In: 2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE), IEEE. p. 115–21.
[53] Zhou Qian, Qi Saibing, Ren Cong. Gene essentiality prediction based on chaos game representation and spiking neural networks. Chaos Solitons Fractals 2021;144:110649.
[54] Tanchotsrinon Watcharaporn, Lursinsap Chidchanok, Poovorawan Yong. A high performance prediction of hpv genotypes by chaos game representation and singular value decomposition. BMC Bioinf 2015;16(1):1–13.
[55] Han Guo-Sheng, Li Qi, Li Ying. Comparative analysis and prediction of nucleosome positioning using integrative feature representation and machine learning algorithms. BMC Bioinf 2021;22(6):1–24.
[56] Zheng Kai, You Zhu-Hong, Li Jian-Qiang, Wang Lei, Guo Zhen-Hao, Huang Yu-An. icda-cgr: Identification of circrna-disease associations based on chaos game representation. PLoS Comput Biol 2020;16(5):e1007872.
[57] Emam Mohamed, Ali Amna, Abdelrazik Eman, Elattar Mustafa, El-Hadidi Mohamed. Detection of mammalian coding sequences using a hybrid approach of chaos game representation and machine learning. In: 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE. p. 2949–51.
[58] Hoang Tung, Yin Changchuan, Yau Stephen S-T. Splice sites detection using chaos game representation and neural network. Genomics 2020;112(2):1847–52.
[59] Anders Simon. Visualization of genomic data with the hilbert curve. Bioinformatics 2009;25(10):1231–5.
[60] Bartoszewicz Jakub M, Seidel Anja, Renard Bernhard Y. Interpretable detection of novel human viruses from genome sequencing data. NAR Genomics Bioinf 2021;3(1):lqab004.
[61] Rabeb Touati, Sondes Haddad-Boubaker, Imen Ferchichi, Imen Messaoudi, Afef Elloumi Ouesleti, Henda Triki, Zied Lachiri, and Maher Kharrat. Comparative genomic signature representations of the emerging covid-19 coronavirus and other coronaviruses: High identity and possible recombination between bat and pangolin coronaviruses. Genomics 112(6): 2020; 4189–4202.
[62] Sengupta Dipendra C, Hill Matthew D, Benton Kevin R, Banerjee Hirendra N. Similarity studies of corona viruses through chaos game representation. Comput Mol Biosci 2020;10(3):61.
[63] Randhawa Gurjit S, Soltysiak Maximillian PM, El Roz Hadi, de Souza Camila PE, Hill Kathleen A, Kari Lila. Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: Covid-19 case study. Plos one 2020;15(4):e0232391.
[64] Jampour Mahdi, Yaghoobi Mahdi, Ashourzadeh Maryam, Soleimani Adel. A new fast technique for fingerprint identification with fractal and chaos game theory. Fractals 2010;18(03):293–300.
[65] Jampour Mahdi, Ebrahimzadeh Reza, Yaghoobi Mahdi, Soleimani-Nezhad Adel. Towards a fast method for iris identification with fractal and chaos game theory. Int J Pattern Recogn Artif Intell 2012;26(04):1256011.
[66] Meloon Brian, Sprott Julien C. Quantification of determinism in music using iterated function systems. Empirical Stud Arts 1997;15(1):3–13.
[67] Cohen-McFarlane Madison, Dick Kevin, Green James R, Goubran Rafik. Chaos game representation of audio signals. In: 2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE. p. 1–6.
[68] Stoean Catalin, Lichtblau Daniel. Author identification using chaos game representation and deep learning. Mathematics 1933;8(11):2020.
[69] Cristescu Constantin P, Stan Cristina, Scarlat Eugen I. Modeling with the chaos game (i). simulating some features of real time series. UPB Sci Bull Ser A 2009;71:95–100.
[70] Talatahari Siamak, Azizi Mahdi. Chaos game optimization: a novel metaheuristic algorithm. Artif Intell Rev 2021;54(2):917–1004.
[71] Dixon J, Karlsson C. Chaos game for data compression and encoding. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE), pages 7–13. The Steering Committee of The World Congress in Computer Science, Computer..., 2018.
[72] Yaghoobi Mahdi et al. A new approach in dna sequence compression: Fast dna sequence compression using parallel chaos game representation. Expert Syst Appl 2019;116:487–93.
[73] Ayubi Peyman, Setayeshi Saeed, Rahmani Amir Masoud. Deterministic chaos game: a new fractal based pseudo-random number generator and its cryptographic application. J Inf Secur Appl 2020;52:102472.