Relating SARS-CoV-2 Variants Using Cellular Automata Imaging

Luryane Ferreira de Souza (✉ luryane.souza@ufob.edu.br)

Federal University of Western Bahia

Tarcísio Marciano da Rocha Filho

University of Brasília

Marcelo Albano Moret

Bahia State University

Research Article

Keywords: SARS-CoV-2, COVID-19, Cellular Automata, Protein Sequencing

Posted Date: February 18th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1320867/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Relating SARS-CoV-2 Variants Using Cellular Automata Imaging

Luryane F. Souza^{1,2,*}, Tarcísio M. Rocha Filho^{3,+}, and Marcelo A. Moret^{2,4,+}

¹Universidade Federal do Oeste da Bahia, CCET, Barreiras, 47808-021, Brazil

²SENAI-CIMATEC, Salvador, 41650 -010, Brazil

³Universidade de Brasília, ICCMP & IF, Brasília, 70910-900, Brazil

⁴UNEB, DCET, Salvador, Brazil

*luryane.souza@ufob.edu.br

+marciano@fis.unb.br, mamoret@gmail.com

ABSTRACT

We classify the main variants of the SARS-CoV-2 virus representing a given biological sequence represented as a symbolic sequence of digital codes and its evolution by a cellular automata with a properly chosen rule. The spike protein, common to all variants of the SARS-CoV-2 virus, is then represented by an image of the cellular automaton evolution which represents in a more visible way important features of the protein. We use information theory Hamming distance between different stages of the evolution of the cellular automaton representing seven variants with respect to the original Wuhan/China virus. We show that our approach allows to classify together variants with common ancestors and same mutations. Although being a much simpler method, it can be used as an alternative for building phylogenetic trees.

Keywords: SARS-CoV-2, COVID-19, Cellular Automata, Protein Sequencing.

1 Introduction

The disruption caused during the last two years by the COVID-19 pandemic is hard to be underestimated. From more than five million deaths and 270 million cases world-wide, according to official sources¹, to economic disruption in most countries². In December 31, 2019 the first case was reported in the city of Wuhan, China, and in January 9, 2020, the World Health Organization (WHO) informed that Chinese scientists reported that the disease was caused by a new coronavirus. In February 11, 2020, in order to not associate the disease with any locality or groups of people the new coronavirus was named SARS-CoV-2 and the disease it caused COVID-19. In March 11 of that same year the WHO declared the outbreak a pandemic³.

The SARS-CoV-2 virus is part of the same virus family as the SARS-CoV and MERS-CoV viruses, which were responsible for epidemics in China (2003) and Saudi Arabia (2012)⁴. The last decade or so witnessed important developments in genome sequencing techniques, with a major increase in information gathering (data) on DNA, RNA, and protein sequences, as exemplified by the amount of data in databases such as GenBank⁵ and UniProt⁶ (for a more thorough account on genomic databases see^{7,8}). Genomic information on animals, plants, and significant disease-causing viruses and bacteria are now easily available to researchers worldwide. Even before COVID-19 was declared a pandemic, researchers in China determined the genomic sequencing of the virus⁹. Genomic sequencing is a crucial for designing vaccines, identify variants, determine the virus family and to drugs development¹⁰⁻¹². The SARS-CoV-2 is a single-stranded RNA virus, with a genome size with of 30 Kb, and four structural proteins: Nucleocapsid (N), Matrix (M), Envelope (E) and the Spike (S) proteins^{4,10}. The latter is responsible for recognizing and helping the virus enter the cell, possibly the main reason this protein has been widely studied. Mutations in the SARS-CoV-2 viruses resulting in new variants features mutations in the spike protein increasing replication within cells, and thence an increased transmissibility⁸.

A protein can be represented as a primary structure formed by a sequence of long strings of characters containing all information on the protein: structure, function, hydrophobicity and different motifs. Several researchers have studied how to extract information from the sequence, such as hydrophobicity¹³⁻¹⁶, fractality^{17,18}, geometric and thermodynamic aspects¹⁹⁻²¹. Cellular Automata have been widely used to model complex systems with simple, easy-to-understand rules²², and in recent years many papers were devoted study protein related problems using this approach. ²³ proposed a protein folding model based on cellular automata, with straightforward evolutionary rules based on the hydrophobicity of amino acids. Other works dedicated to the same problem and using cellular automata include²⁴⁻²⁶. Cellular Automata Image (CAI) analysis²⁷ is a powerful tool to classify protein structure²⁸⁻³⁰ and virus taxonomy³¹. These images can contain important information on the

modeled system, for example, CAI allows to differentiate similar systems with respect to those significantly different. The identification of functions, structures, location, and common ancestry of a protein sequence can be performed by a comparison with other known proteins in databases, using alignment, similarity, and homology techniques³². In the present paper we propose a protein comparison approach using the corresponding cellular automaton image and the information theoretic Hamming metric for the distance between such images as a measure of similarity and difference, applied to the spike protein. The distance is measured with respect to the S protein in the initial virus as first detected in Wuhan and for the following Variants Of Concern (VOCs) with mutations of the Spike protein: Alpha (first identified in the United Kingdom), Beta (South Africa), Gamma (Brazil), Delta (India), and the more recent Omicron (South Africa), B.1.1.28, and P2 (Brazil). We show that our approach allows a simple and straightforward method to determine evolutionary relationships.

In Section 2 we present the cellular automata image approach for protein classification and the Hamming distance. Our results are presented and discussed in Section 3, and we close the paper with some concluding remarks in Sec. 4.

2 Methods

Cellular automata are discrete dynamical systems with simple local evolution rules and, despite this, can show complex behavior²². The rules take into account the state of neighboring cells, analogous to protein structure since physicochemical characteristics of neighboring amino acids influence the folding or function of the protein. The cellular automata considered here are composed by four components: a grid, the set of states, the neighborhood of each state, and the local transition rule. Several possibilities exist for encoding the sequence of the 20 types of amino acids in a protein: an 8-digit code for each amino acid³³, or codes reflecting physicochemical characteristics and degeneracy, based on rules of similarity, complementarity, and theory of molecule recognition and information theory, with a 5-digit code for each amino acid³⁴. For the present purposes, we implement here the code from Ref.²⁸ representing the amino acid sequences based on the hydrophobicity index of each amino acid, with the corresponding coding for each protein is given in Table 1. For the Spike protein sequence, which has 1273 amino acids, each one coded as a 5 digit sequence, with that the whole protein written using 6365 cells forming the first line of the cellular automata (initial condition), each one having a state of 0 or 1. The protein is then a one-dimensional grid of size $N = 6365$, with $x_i^t = 0, 1$, denoting the state of the cell at the discrete time t and position i , $i = 1, \dots, N$. The neighborhood of the cell at position i is composed by those cells at positions $i - 1$, i and $i + 1$, and periodic boundary conditions, resulting in $2^3 = 8$ different states for the neighborhood. For each possible configuration of the neighborhood, the middle cell can assume two possible states, such that the number of possible evolution rules is given by $2^8 = 256$.³⁵ ³⁵ and ³¹ studied the most appropriate rule for SARS-CoV virus classification and for distinguishing them from other viruses, and concluded that the best suited rule being Wolfram's 184³⁵ (see Fig. 1). A typical feature only seen in SARS-CoV viruses is a V pattern pattern in the cellular automaton image.

Amino Acids	K	N	D	E	P	Q	R
Decimal Code	6	8	9	10	11	12	13
Binary code	00110	01000	01001	01010	01011	01100	01101
Amino Acids	S	T	G	A	H	W	Y
Decimal Code	14	15	16	17	18	20	21
Binary code	01110	01111	10000	10001	10010	10100	10101
Amino Acids	F	L	M	I	V	C	
Decimal Code	23	24	26	27	28	30	
Binary code	10111	11000	11010	11011	11100	11110	

Table 1. Coding for each of the 20 possible amino acids²⁸.

Figure 1. Rule 184 from³⁵ for an elementary cell automaton with three neighbors. The state 0 is represented in white and 1 in black.

In order to implement a numeric metric to distinguish different images we consider here the information theoretic Hamming

distance D_H^{36} , which for a binary sequence can be written as

$$D_H(t) = \frac{1}{N} \sum_{i=1}^N \|x_i^t - \bar{x}_i^t\| \quad (1)$$

with N the size of the grid, x_i^t the state of the cell of the S protein in the initial Wuhan strain and $\bar{x}_i^t$ the Spike protein of the given variant.

2.1 Ethics declarations

We did not use human samples/human data in these studies.

3 Results and discussion

Following the considerations above we implemented the cellular automaton for the SARS-CoV-2 spike protein from available genomic data (data for Alpha, Beta, Gamma, Delta, B.1.1.28, P2 and the original strain are available at ⁵ and the Omicron data at ³⁷), with the coding in Table 1, and evolved using the rule in Fig. 1 for 1000 time steps. Figure 2 shows the resulting image representing the evolution of the automaton for each variant, where the V shaped patterns characteristic of the SARS-CoV viruses³¹ are clearly visible. Figure 3 shows the time evolution of the Hamming distance D_H for each variant with respect to the original Wuhan strain. The initial distance has a small value, as expect for variants of the same virus, and increases with t up to an asymptotic constant value after approximately $t = 400$ steps. The small number of mutations, if compared to the number of amino-acids in the protein and measured by the small Hamming distance at $t = 0$, is amplified by the evolution of the cellular automata and results in quite different asymptotic values of D_H , after an initial irregular transient of roughly 200 time steps. This allows us to classify the cellular automata as Wolfram Class IV, with an intermediate behavior between Classes II (periodical) and III (chaotic). Although the Omicron variant has more mutations (and therefore a higher value of D_H) than other known VOCs, with 33 amino acid changes in the spike protein³⁸, its distance plot remains closed to the variants sharing the N501Y mutation (see Table 2 for the mutations in each variant). This large number of modifications seems to be linked to increased rates of transmissibility and possibly smaller efficiency of vaccines³⁹.

Figure 2. Imaging of the evolution of protein cellular automata from coding in from Table 1 and the Wolfram’s rule in Fig. 1, for the different variants.

Table 2 shows the different mutations present in each main variant of the SARS-CoV-2 virus. We then see from Fig. 3 that the present approach groups the variants carrying the N501Y mutation, the sense that final stationary Hamming distance between these variants and the original are more closer and with higher values. The Gamma and P2 variants are also closer as they have the same clade B.1.1.28 (note that the distance for P2 and B.1.1.28 are practically the same in the Figure), while the Delta variant, which carries the P681R mutation unfamiliar to the other variants, is the one with smallest distance. We believe that the present approach is a straightforward way to measure evolutionary distances between SARS-CoV-2 variants, much simpler than other techniques as in⁴⁰ and⁴¹ where a normalized Laplacian pyramid is employed to measure pairwise similarities in cellular automata image wavelet images in order to build tree phylogenetics.

Figure 3. Left: Hamming distance for the time evolution of the cellular automata associated to the spike protein between the different variants of SARS-CoV-2 and the initial Wuhan variant. Right: Zoom over the initial period.

Variant	Mutations
Alpha	HV69-70del, Y145del, N501Y, A570D, D614G, P681H, T716I, S982A, D1118H
Beta	L18F, D80A, D215G, R246I, K417N, E484K, N501Y, D614G, A701V
Gamma	L18F, T20N, P26S, HV69-70del, D138Y, Y145H, R190S, K417T, E484K, N501Y, D614G, H655Y, T1027I, V1176F
Delta	T95I, G142D, E154K, L452R, E484Q, D614G, P681R, Q1071H
B.1.1.28	HV69-70del, Y145del, D614G, V1176F
P2	HV69-70del, Y145del, E484K, D614G, V1176F
Omicron	A67V, HV69-70del, T95I, G142D, VYY143-145del, N211I, L212del, G339D, S371L, S373P, S375F, K417N, N440K, G446S, S477N, T478K, E484A, Q493R, G496S, Q498R, N501Y, Y505H, T547K, D614G, H655Y, N679K, P681H, N764K, D796Y, N856K, Q954H, N969K, L981F

Table 2. Mutations the Spike protein of the SARS-CoV-2 variants from³⁸.

The variants with smallest values of D_H are those with the smallest number of mutations as given in Table 2 are the Delta, B.1.1.28 and P2, which are also the variants without the N501Y mutation. Despite the differences in the images of each variant, resulting from the mutations, the chosen rule also results in the V-shaped pattern for SARS-CoV-2 type coronaviruses. During the COVID-19 pandemic many mutations occurred in the virus sequence, but without a functional change in the Spike protein, although some mutations may bring some advantages. However, since different sequences perform the same function, mutations in proteins are degenerate, a behavior fundamental for natural selection to occur. Without degeneracy, there is no genetic variability, and this hinders natural selection from acting⁴².

4 Concluding remarks

The approach presented in this work allowed to cluster variants with common ancestors by using the asymptotic Hamming distance to the cellular automaton images of each variant, as shown in Fig. 2, and yields a more straightforward and simpler evolutionary classification of variants. Indeed, it avoids the use of more complicated approaches such as alignment technique, similarity analysis and image processing to relate the variants evolutionarily. It allowed to properly discern the deviation of Omicron with respect to other variants using the Hamming distance, although still preserving the V shaped pattern characteristic of the SARS-CoV viruses and having the highest number of mutations among known variants. Along the current COVID-19 pandemic variants evolved to become more infectious and to partially avoid immunity, one of the being that the system is degenerate. Our method also approximates the variants that have the N501Y mutation. Furthermore, after three iterations of the Wuhan protein, amino acid changes N to Y at position 501. This rapid convergence of these amino acids at this position suggests an alternative explanation for the emergence of Alpha, Beta, and Gamma on three continents simultaneously, showing evolutionary convergence. Also, without degeneration, mutations could lead to unfavorable structures for the virus, making it easier to control the spread of the disease⁴².

Cellular automata is a simple tool that allows to extract meaningful information from proteins sequences, with a very low computational cost. We hope that the present work will contribute as an useful tool to build protein phylogenetic trees.

5 Data Availability

The datasets used during the current study are the sequences of the Spike proteins of the virus initially found in Wuhan [YP_009724390.1] and its variants Alpha [QWP89177.1], Beta [UAL50115.1], Gamma [QXF22923.1], Delta [QXP08802.1], Omicron [UGO97992.1], B.1.1.28 [QQK84800.1] and P2 [QXF22396.1] which are available on the website <https://www.ncbi.nlm.nih.gov/genbank/>. The Spike protein sequences generate the cellular automata used to calculate the Hamming distance. The codes used in this paper can be found in the Additional information section.

References

1. John Hopkins University. John Hopkins Coronavirus Resource Center (2021). Available on-line: <https://coronavirus.jhu.edu/map.html> (accessed on 4 September 2021).
2. Tooze, A. *Shutdown – How Covid Shook the World's Economy* (Penguin Random House, New York City, 2021).
3. World Health Organization. WHO timeline-COVID-19. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/interactive-timeline?gclid=CjwKCAiA7dKMBhBCEiwAO_crFAhknuq4kc_PZRW1qx3v_bMHTvAmmEewQ2vyKtZ47HyUy7DLGIZxoCkC4QAvD_BwE#event-115 (2020). Accessed 17/11/2021.
4. Machhi, J. *et al.* The natural history, pathobiology, and clinical manifestations of SARS-CoV-2 infections. *J. Neuroimmune Pharmacol.* **15**, 359–386, DOI: [10.1007/s11481-020-09944-5](https://doi.org/10.1007/s11481-020-09944-5) (2020).
5. GenBank. National Center for Biotechnology Information (2021).
6. UniProt. The Universal Protein Resource (2021).
7. Chen, C., Huang, H. & Wu, C. H. Protein bioinformatics databases and resources. *Methods Mol. Biol.* **1558**, 3–39, DOI: [10.1007/978-1-4939-6783-4_1](https://doi.org/10.1007/978-1-4939-6783-4_1) (2017).
8. NIH – National Library of Medicine. NCBI SARs-CoV-2 Resources (2021).
9. Wu, F. *et al.* A new coronavirus associated with human respiratory disease in China. *Nature* **579**, 1–8, DOI: [10.1038/s41586-020-2008-3](https://doi.org/10.1038/s41586-020-2008-3) (2020).
10. Khan, M. T. *et al.* Structures of SARS-CoV-2 RNA-Binding proteins and therapeutic targets. *Intervirology* **64**, 1–14, DOI: [10.1159/000513686](https://doi.org/10.1159/000513686) (2021).
11. Chou, K. C., Wei, D. Q. & Zhong, W. Z. Binding mechanism of coronavirus main proteinase with ligands and its implication to drug design against SARS. *Biochem. Biophys. Res. Commun.* **308**, 148–151, DOI: [10.1016/S0006-291X\(03\)01342-1](https://doi.org/10.1016/S0006-291X(03)01342-1) (2003).
12. Chou, K. C., Wei, D. Q., Du, Q. S., Sirois, S. & Zhong, W. Z. Progress in computational approach to drug development against SARS. *Curr. Medicinal Chem.* **13**, 3263–3670, DOI: [10.2174/092986706778773077](https://doi.org/10.2174/092986706778773077) (2006).
13. Moret, M. A. & Zebende, G. F. Amino acid hydrophobicity and accessible surface area. *Phys Rev E Stat Nonlin Soft Matter Phys* **75**, 011920, DOI: [10.1103/PhysRevE.75.011920](https://doi.org/10.1103/PhysRevE.75.011920) (2007).
14. Phillips, J. C. Scaling and self-organized criticality in proteins I. *Proc. Natl. Acad. Sci.* **106**, 3107–3112, DOI: [10.1073/pnas.0811262106](https://doi.org/10.1073/pnas.0811262106) (2009).
15. Phillips, J. C. Synchronized attachment and the Darwinian evolution of coronaviruses CoV-1 and CoV-2. *Phys. A: Stat. Mech. its Appl.* **581**, 126202, DOI: <https://doi.org/10.1016/j.physa.2021.126202> (2021).
16. Li, S., Cai, C., Gong, J., Liu, X. & Li, H. A fast protein binding site comparison algorithm for proteome-wide protein function prediction and drug repurposing. *Proteins: Struct. Funct. Bioinforma.* **89**, 1541–1556, DOI: [10.1002/prot.26176](https://doi.org/10.1002/prot.26176) (2021).
17. Moret, M. A., Miranda, J. G. V., Nogueira, E., Santana, M. C. & Zebende, G. F. Self-similarity and protein chains. *Phys. Rev. E, Stat. Nonlinear, Soft Matter Phys.* **71**, 012901, DOI: [10.1103/PhysRevE.71.012901](https://doi.org/10.1103/PhysRevE.71.012901) (2005).
18. Moret, M. A., Santana, M. C., Nogueira, E. & Zebende, G. F. Protein chain packing and percolation threshold. *Phys. A-Statistical Mech. Its Appl.* **361**, 250–254 (2006).
19. Moret, M. A. Self-organized critical model for protein folding. *Phys. A: Stat. Mech. its Appl.* **390**, 3055–3059, DOI: <https://doi.org/10.1016/j.physa.2011.04.008> (2011).
20. Xu, X. L., Shi, J. X., Wang, J. & Li, W. Long-range correlation and critical fluctuations in coevolution networks of protein sequences. *Phys. A: Stat. Mech. its Appl.* **562**, 125339, DOI: <https://doi.org/10.1016/j.physa.2020.125339> (2021).

21. Nelson, E. D. & Onuchic, J. N. Proposed mechanism for stability of proteins to evolutionary mutations. *Proc. Natl. Acad. Sci.* **95**, 10682–10686, DOI: [10.1073/pnas.95.18.10682](https://doi.org/10.1073/pnas.95.18.10682) (1998).
22. Toffoli, T. & Margolus, N. *Cellular Automata Machines: A New Environment for Modeling* (MIT press in scientific computation, Cambridge, 1987).
23. Sleit, A. & Madain, A. Protein folding in the two-dimensional hydrophobic polar model based on cellular automata and local rules. *Int. J. Comput. Netw. Inf. Secur.* **16** (2016).
24. Varela, D. & Santos, J. Protein folding modeling with neural cellular automata using Rosetta. In *GECCO '16 Companion: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, 1307–1312 (Association for Computing Machinery, New York, NY, USA, 2016).
25. Varela, D. & Santos, J. Protein folding modeling with neural cellular automata using the Face-Centered Cubic model (2017). Published in IWINAC 19 June 2017.
26. Varela, D. & Santos, J. Automatically obtaining a cellular automaton scheme for modeling protein folding using the FCC model. *Nat. Comput.* **18**, DOI: [10.1007/s11047-018-9705-y](https://doi.org/10.1007/s11047-018-9705-y) (2019).
27. Wolfram, S. Cellular automata as models of complexity. *Nature* **311**, 419–424 (1984).
28. Xiao, X. & Chou, K. Digital coding of amino acids based on hydrophobic index. *Protein Pept. Lett.* **14**, 871–5 (2007).
29. Xiao, X., Wang, P. & Chou, K. C. Predicting protein structural classes with pseudo amino acid composition: An approach using geometric moments of cellular automaton image. *J. Theor. Biol.* **254**, 691–6, DOI: [10.1016/j.jtbi.2008.06.016](https://doi.org/10.1016/j.jtbi.2008.06.016) (2008).
30. Kavianpour, H. & Vasighi, M. Structural classification of proteins using texture descriptors extracted from the cellular automata image. *Amino Acids* **49**, 261–271, DOI: [10.1007/s00726-016-2354-5](https://doi.org/10.1007/s00726-016-2354-5) (2017).
31. Wang, M. *et al.* A new nucleotide-composition based fingerprint of SARS-CoV with visualization analysis. *Medicinal Chem.* 39–47, DOI: [10.2174/1573406053402505](https://doi.org/10.2174/1573406053402505) (2005).
32. Gabler, F. *et al.* Protein sequence analysis using the MPI bioinformatics toolkit. *Curr. Protoc. Bioinforma.* **72**, e108, DOI: [10.1002/cpbi.108](https://doi.org/10.1002/cpbi.108) (2020).
33. Ghosh, S. & Chaudhuri, P. P. Cellular automata model for proteomics and its application in cancer immunotherapy. In *Cellular Automata. ACRI 2018. Lecture Notes in Computer Science*, 3–15 (Springer International Publishing, 2018).
34. Xiao, X., Shao, S., Ding, Y. & Chen, X. Digital coding for amino acid based on cellular automata. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, vol. 5, 4593–4598, DOI: [10.1109/ICSMC.2004.1401256](https://doi.org/10.1109/ICSMC.2004.1401256) (2004).
35. Xiao, X. *et al.* Using cellular automata to generate image representation for biological sequences. *Amino Acids* **28**, 29–35, DOI: [10.1007/s00726-004-0154-9](https://doi.org/10.1007/s00726-004-0154-9) (2005).
36. Hamming, R. W. Error detecting and error correcting codes. *The Bell Syst. Tech. J.* **29**, 147–160, DOI: [10.1002/j.1538-7305.1950.tb00463.x](https://doi.org/10.1002/j.1538-7305.1950.tb00463.x) (1950).
37. Mullen, J. L. *et al.* Outbreak.Info (2021). Accessed 17 December 2021.
38. European Centre for Disease Prevention and Control. Implications of the emergence and spread of the SARS-CoV-2 b.1.1. 529 variant of concern (Omicron) for the EU/EEA. <https://www.ecdc.europa.eu/en/publications-data/threat-assessment-brief-emergence-sars-cov-2-variant-b.1.1.529> (2021). Accessed 17/12/2021.
39. World Health Organization. Enhancing Readiness for Omicron (b.1.1.529): Technical brief and priority actions for member states. [https://www.who.int/publications/m/item/enhancing-readiness-for-omicron-\(b.1.1.529\)-technical-brief-and-priority-actions-for-member-states](https://www.who.int/publications/m/item/enhancing-readiness-for-omicron-(b.1.1.529)-technical-brief-and-priority-actions-for-member-states) (2021). Accessed 17/12/2021.
40. Wu, Z. C., Xiao, X. & Chou, K. C. 2D-MH: A web-server for generating graphic representation of protein sequences based on the physicochemical properties of their constituent amino acids. *J. Theor. Biol.* **267**, 29–34, DOI: [10.1016/j.jtbi.2010.08.007](https://doi.org/10.1016/j.jtbi.2010.08.007) (2010).
41. Rahman, M. M., Biswas, B. A. & Bhuiyan, M. I. H. Protein similarity analysis by wavelet decomposition of cellular automata images. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 1–6 (IEEE, Cox's Bazar, Bangladesh, 2019).
42. Edelman, G. M. & Gally, J. A. Degeneracy and complexity in biological systems. *Proc. Natl. Acad. Sci.* **98**, 13763–13768, DOI: [10.1073/pnas.231499798](https://doi.org/10.1073/pnas.231499798) (2001).

Author contributions statement

L.F.S. conducted the computer modelling, M.A.M. separated the protein sequence data from the variants, T.M.R.F. made the images of the automata. All analyzed the results and reviewed the manuscript

Additional information

Competing Interests

The author(s) declare no competing interests.

Computer Code

Here I present the programs used in this paper. The first program called protein.cpp is a C++ code that transforms the protein sequence of size 1273 into a binary sequence of size 6365.

```
// protein.cpp
#include <iostream>
#include <iomanip>
#include <stdio.h>
using namespace std;

int main(int argc, char** argv)
{
    int n,i,l,r,s,f,e, seq[10000], vetor[10000];

    FILE *arquivo, *arquivo1;

    n=1273; //protein size

    //READ A FILE WITH THE PROTEIN SEQUENCE
    arquivo1 = fopen("B1128.txt", "rt");

    // This command lists each typed letter at a position in a vector
    for(i=0; i<n; i++)
    {
        char palavra[n];
        fscanf(arquivo1, "%c \n", &palavra[i]);
        char ascChar = toupper(palavra[i]);
        seq[i]= ascChar-0; // turns each letter in the sequence into an ASCII table

    }
    fclose(arquivo1);

    //Changing ASCII table numbers to decimal hydrophobicity
    for(f=0 ; f<n ; f++)
    {
        if(seq[f]==65)//A
            seq[f]=17;
        if(seq[f]==67)//C
            seq[f]=30;
        if(seq[f]==68)//D
            seq[f]=9;
        if(seq[f]==69)//E
            seq[f]=10;
        if(seq[f]==70)//F
            seq[f]=23;
        if(seq[f]==71)//G
            seq[f]=16;
```

```

        if (seq[f]==72)//H
        seq[f]=18;
        if (seq[f]==73)//I
        seq[f]=27;
        if (seq[f]==75)//K
        seq[f]=6;
        if (seq[f]==76)//L
        seq[f]=24;
        if (seq[f]==77)//M
        seq[f]=26;
        if (seq[f]==78)//N
        seq[f]=8;
        if (seq[f]==80)//P
        seq[f]=11;
        if (seq[f]==81)//Q
        seq[f]=12;
        if (seq[f]==82)//R
        seq[f]=13;
        if (seq[f]==83)//S
        seq[f]=14;
        if (seq[f]==84)//T
        seq[f]=15;
        if (seq[f]==86)//V
        seq[f]=28;
        if (seq[f]==89)//Y
        seq[f]=21;
        if (seq[f]==87)//W
        seq[f]=20;
        if (seq[f]==45)// –
        seq[f]=0;
    }

//CREATE A FILE WITH THE PROTEIN SEQUENCE IN BINARY
arquivo = fopen("B1128_inicial.txt", "wt");

        if (arquivo==NULL){
            cout << "Could not open program" << endl;
            exit(0);
        }

// Makes the change from decimal to 5-digit binary.

for(e=0; e<n ;e++){
    for(l = 4; l >= 0; l--) {
        s=seq[e];
        r = s >> l;
        if(r & 1) {
            vetor[4-l]=1;
        }
        else {
            vetor[4-l]=0;
        }
        fprintf(arquivo,"%d \n", vetor[4-l] );
    }
}

```

```

        fclose(arquivo);

return 0;
}

```

```
//-----
```

The second program called automata.cpp is a C++ code. The program automata.cpp receives the protein sequence from protein.cpp placing it in the first line of the automaton and evolves the automaton with rule 184

```

// automata.cpp
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip>
using namespace std;

int main(int argc, char** argv)
{
    int N,linhas, regra,l,r,m,a,lin, bin[8], anterior[10000],atual[10000];
    FILE *arquivo, *arquivo2;

    /* Input */
    N=6365; //column numbers
    linhas=1000; //number of lines
    regra=184; // automaton rule

    //automaton rule in binary
    for(l = 7; l >= 0; l--) {
        r = regra >> l;
        if(r & 1) {
            bin[7-l]=1;
        } else {
            bin[7-l]=0;
        }
        cout << bin[7-l];
    }

    //READ A FILE WITH THE PROTEIN SEQUENCE IN BINARY
    arquivo = fopen(" B1128_inicial.txt", "rt");
    for(m=0; m<N;m++){
        fscanf(arquivo," %d \n", &atual[m]);
    }
    fclose(arquivo);

    //CREATE A FILE FOR THE CELLULAR AUTOMATA
    arquivo2 = fopen(" automatoB1128.txt", "wt");

    // First line of the automaton is the protein sequence
    for(m=0; m<N;m++){
        fprintf(arquivo2,"%d \t", atual[m]);
    }
    fprintf(arquivo2, "\n");

    // Change the current line to the old one
    for(m=0; m<N;m++){
        anterior[m]=atual[m];
    }
}

```

```

}

// Neighborhood analysis
for (lin=2; lin <= linhas; lin++){
for (a=0; a<N; a++){
    if (a==0){
        if ((anterior[N-1]==1 && anterior[a]==1) && anterior[a+1]==1){
            atual[a]=bin[0];
        }
        else if ((anterior[N-1]==1 && anterior[a]==1) && anterior[a+1]==0){
            atual[a]=bin[1];
        }
        else if ((anterior[N-1]==1 && anterior[a]==0) && anterior[a+1]==1){
            atual[a]=bin[2];
        }
        else if ((anterior[N-1]==1 && anterior[a]==0) && anterior[a+1]==0){
            atual[a]=bin[3];
        }
        else if ((anterior[N-1]==0 && anterior[a]==1) && anterior[a+1]==1){
            atual[a]=bin[4];
        }
        else if ((anterior[N-1]==0 && anterior[a]==1) && anterior[a+1]==0){
            atual[a]=bin[5];
        }
        else if ((anterior[N-1]==0 && anterior[a]==0) && anterior[a+1]==1){
            atual[a]=bin[6];
        }
        else if ((anterior[N-1]==0 && anterior[a]==0) && anterior[a+1]==0){
            atual[a]=bin[7];
        }
    }
    else if (a==N-1){
        if ((anterior[a-1]==1 && anterior[a]==1) && anterior[0]==1){
            atual[a]=bin[0];
        }
        else if ((anterior[a-1]==1 && anterior[a]==1) && anterior[0]==0){
            atual[a]=bin[1];
        }
        else if ((anterior[a-1]==1 && anterior[a]==0) && anterior[0]==1){
            atual[a]=bin[2];
        }
        else if ((anterior[a-1]==1 && anterior[a]==0) && anterior[0]==0){
            atual[a]=bin[3];
        }
        else if ((anterior[a-1]==0 && anterior[a]==1) && anterior[0]==1){
            atual[a]=bin[4];
        }
        else if ((anterior[a-1]==0 && anterior[a]==1) && anterior[0]==0){
            atual[a]=bin[5];
        }
        else if ((anterior[a-1]==0 && anterior[a]==0) && anterior[0]==1){
            atual[a]=bin[6];
        }
        else if ((anterior[a-1]==0 && anterior[a]==0) && anterior[0]==0){
            atual[a]=bin[7];
        }
    }
}
}

```



```

    }
}
else {
    if((anterior[a-1]==1 && anterior[a]==1) && anterior[a+1]==1){
        atual[a]=bin[0];
    }
    else if((anterior[a-1]==1 && anterior[a]==1) && anterior[a+1]==0){
        atual[a]=bin[1];
    }
    else if((anterior[a-1]==1 && anterior[a]==0) && anterior[a+1]==1){
        atual[a]=bin[2];
    }
    else if((anterior[a-1]==1 && anterior[a]==0) && anterior[a+1]==0){
        atual[a]=bin[3];
    }
    else if((anterior[a-1]==0 && anterior[a]==1) && anterior[a+1]==1){
        atual[a]=bin[4];
    }
    else if((anterior[a-1]==0 && anterior[a]==1) && anterior[a+1]==0){
        atual[a]=bin[5];
    }
    else if((anterior[a-1]==0 && anterior[a]==0) && anterior[a+1]==1){
        atual[a]=bin[6];
    }
    else if(anterior[a-1]==0 && anterior[a]==0 && anterior[a+1]==0){
        atual[a]=bin[7];
    }
}
}

// Saves the current row that was obtained with the rule
for(m=0; m<N;m++){
    fprintf(arquivo2,"%d \t", atual[m]);
}
fprintf(arquivo2, "\n");

// Replace the current line with the previous one
for(m=0; m<N;m++){
    anterior[m]=atual[m];
}
}
fclose(arquivo2);
return 0;
}
//_____

```

The third program, written in Python language, calculates the Hamming distance of the variant automata for the Wuhan automaton.

```

#distancia.ipynb
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import odeint
import pylab as pl
import math
import matplotlib.ticker as tck

#Input

```

```

L=1000 #number of lines
C=6365 #column numbers

a=0
s=int(a)
fname='teste_6.txt' #Wuhan automata
A=np.loadtxt(fname)

for i in range(1,9):
    fname_S='teste_{i}.txt' #Variants automata
    anterior_S=np.loadtxt(fname_S.format(i=i))
    b=0
    soma=int(b)
    soma=0
    D=[]
    for l in range(L):
        for m in range(C):
            if A[l][m]==anterior_S[l][m]:
                s=0
            else:
                s=1
            soma=soma+s
        D.append(soma)
        soma=0
    fname_S = "DistanciaHamming_{i}.txt" # Hamming distance from each variant
    #automaton to the Wuhan automaton
    atual_S=np.savetxt(fname_S.format(i=i),D,fmt = '%.0f', newline='\n')

```