Shannon entropy, order and image compression

Author: Marta Grasa Lainez
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.[*]

Advisors: Alberto Fernandez-Nieves, Ramon Planet

Abstract: The identification of nontrivial manifestations of order is of great interest in the field of image analysis. Here, we use image compression as a way to quantify this "order". File compression algorithms (as the ones used in .zip compression) produce smaller files the less disordered the original file is. We apply this principle to various images of dots in disordered dispositions, square lattices and iterations of the Sierpinski carpet, in order to capture a relation between order and image compression. We find that, at least in some cases, we can use compression to quantify order.

I. INTRODUCTION

In condensed matter physics, particles often organize themselves in patterns. However, these are often not easy to detect. In [1], file compression is introduced as a possible method to find the order parameter of a system, particularly as it goes through a phase transition where the order suddenly changes.

Soft condensed matter is the part of condensed matter studying materials that have larger-than-atomic characteristic length scales. We can therefore have direct observations of these systems through optical microscopes and digital cameras. This explains the importance that file compression can have in the field, if any relevant physics can be extracted in the process. An example could be quantifying how ordered the structures observed are.

However, order is oftentimes hard to detect in these images without sophisticated data analysis methodologies. Here, we introduce the concepts of Shannon entropy, computable information density (CID), and the Lempel-Ziv 77 compression algorithm; with the final goal to quantify order in images containing randomly distributed dots, crystalline lattices and fractals.

A. Shannon entropy

In 1948, Claude E. Shannon postulated a way to measure the quantity of information contained in the signals produced by a source. He defined the *Shannon entropy* of a variable X, $H(X)$, as follows [2]:

$$H(X) = -\sum_{\{x_i\}} p(x_i)\log_2(p(x_i)) \ , \tag{1}$$

where $\{x_i\}$ is the set of possible values of X and $p(x_i)$ the probability function of these values. The amount of information is measured in bits, therefore base two logarithms are used.

This entropy can be interpreted as the average number of yes/no questions needed to determine the value x_i of an observation of the variable X, that is, the minimum number of bits required to transmit an observation. It relates to the thermodynamic entropy, if we take each possible value of X to be a microstate and adjust for the Boltzmann constant and the base of the logarithm.

To illustrate this a bit, let's look at the case of a six-sided dice. If the dice is not biased in any way, it has a 1/6 probability of landing on each side, therefore its Shannon entropy is $H = -6 \cdot \frac{1}{6}\log_2\frac{1}{6} = 2.585$. That is, we need 3 bits of information to transmit the result of rolling a dice.

If the dice is biased, it has a higher probability of landing on one side. Therefore, its Shannon entropy decreases, as the average number of yes/no questions needed to know the result is lower. In fact, for a probability p for the dice to land on the biased side, it's Shannon entropy is $H = -p\log_2 p - (1-p)\log_2\frac{1-p}{5}$. By plotting H as a function of p, we can see that H(loaded dice) has a maximum at $p = 1/6$, which corresponds to the unbiased dice; see Fig.1.

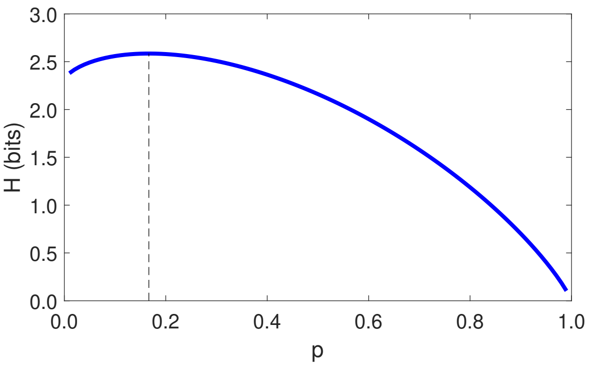

Fig. 1. Shannon entropy H of a biased dice as a function of the probability p of it landing on the biased face. The dashed line indicates the maximum, which corresponds to $p = 1/6$.

[*]Electronic address: mgrasala53@alumnes.ub.edu

B. Computable information density

The Shannon entropy is a very useful concept, but unless we consider a very large numbers of signals from a given source, which would allow knowing the probability functions, it may be hard to obtain. We will therefore approximate it via the *computable information density* (CID) instead, defined as [1]:

$$\mathrm{CID} \equiv \frac{\mathcal{L}(x)}{L} \ , \tag{2}$$

where L is the length of a sequence of events and $\mathcal{L}(x)$ is the length of the binary compressed sequence using a certain compression algorithm. The CID of a sequence can never be smaller than the Shannon entropy of the source. For the algorithm used in .zip compression, the CID becomes the Shannon entropy in the thermodynamic limit: for very long sequences the CID can be considered a good approximation of H and therefore of the amount of information encoded in the sequence [5].

As it is a measure of how much information per character a sequence contains, a random sequence of 0 and 1 will have a much greater CID than a repeating sequence like 01010101... of the same length, as the second one requires much less information to be transmitted.

When an alphabet of length $|\alpha|$ of a sequence contains more than two elements, we must also take $|\alpha|$ into account when studying the amount of information per character. However, here we will only work with binary sequences, and thus we will use the CID as a measure of how compressible a sequence is. Note that when CID ≥ 1, the sequence is not worth compressing.The smaller the CID, the more compressible the sequence is and the less amount of information we need to give in order to losslessly recover it.

C. The Lempel-Ziv 77 compression algorithm

The algorithm commonly used for .zip files is the Lempel-Ziv 77 (LZ77) compression algorithm [3]. This is a lossless compression algorithm, which means that the compressed file can be decoded into the original file without any errors or missing parts. It is also universal in that it compresses any file even without previous knowledge.

It works by searching for repeating patterns in a sequence. It uses a cursor which starts before the first character of the sequence, and searches for the longest sequence beginning with the character to its right which has already occurred. This information is saved as a pair (i, ℓ) where i is the position of the first character of the first time the sequence occurred (or the character itself when it's observed for the first time) and ℓ is the length of the repeated sequence.

As an example, let us compress step by step the sequence $x_1 = baababaa$ using the LZ77 algorithm. The cursor starts on the first character. As this is a new character, it will save $(b, 0)$ and move the cursor to the second character. This is also a new character, therefore it saves $(a, 0)$ and moves the cursor to the third character. This is a repeated character, a (character 2). But the sequence ab has not occurred before, so it will save $(2, 1)$ and move the cursor to the fourth character, where it finds the already appeared sequence ba. This sequence of two characters has appeared starting in character 1, so the algorithm saves $(1, 2)$ and moves the cursor on to the sixth character. There it finds another repeating sequence of three characters, encoded as $(1, 3)$. Sequence x_1 is therefore encoded as:

$$\mathrm{LZ77}(x_1) = \{(b, 0), (a, 0), (2, 1), (1, 2), (1, 3)\} \ . \tag{3}$$

The binary code length of the compressed file $\mathcal{L}(x)$ can be deduced from the number C of pairs (i, ℓ) used to compress the original sequence. Let α be the alphabet of the original sequence, it takes $\log_2(|\alpha| + L)$ bits to save i (a position in the sequence or a character of the alphabet) and $\log_2 \ell_i + O(\log_2(\log_2 \ell_i))$ to specify the length of the sequence starting on character i. Hence, assuming $L \gg |\alpha|$ and using Jensen's inequality for convex functions applied to the case of the logarithm [5], we get:

$$\begin{aligned}\mathcal{L}_{LZ77}(x) \leq C \log_2(|\alpha| + L) + \sum_{i=1}^{C} \log_2 \ell_i + \\ + O\left(\sum_{i=1}^{C} \log_2(\log_2 \ell_i)\right) \\ \leq C \log_2 C + 2C \log_2 \frac{L}{C} + O\left(C \log_2 \log_2 \frac{L}{C}\right) \ .\end{aligned} \tag{4}$$

We then take the following as a reasonable approximation for the binary compressed image using LZ77:

$$\mathcal{L}_{LZ77}(x) \approx C \log_2 C + 2C \log_2 \frac{L}{C} \ . \tag{5}$$

For sequence x_1, we have $L(x_1) = 8$, $C(x_1) = 5$ and therefore:

$$\begin{aligned}\mathcal{L}_{LZ77}(x_1) &= 5 \log_2 5 + 10 \log_2 \frac{8}{5} = 18.390 \\ \mathrm{CID}(x_1) &= \frac{\mathcal{L}_{LZ77}(x_1)}{L(x_1)} = \frac{18.390}{8} = 2.299 \ .\end{aligned} \tag{6}$$

We then conclude that the sequence x_1 is not worth compressing. However, if we duplicate the sequence, obtaining $x_2 = x_1 x_1 = baababaabaababaa$, the length of the sequence $L(x_2) = 16$ is doubled, but only one extra pair is needed to recover the sequence. In this case $\mathrm{LZ77}(x_2) = \{(b, 0), (a, 0), (2, 1), (1, 2), (1, 3), (1, 8)\}$ and $C = 6$. Therefore:

$$\mathcal{L}_{LZ77}(x_2) = 16.245 \qquad \text{and} \qquad \text{CID}(x_2) = 2.031\ . \quad (7)$$

This new sequence requires less information per character than x_1, but still is not worth compressing. By repeating the process of adding the same sequence at the end, we can decrease the CID (Fig. 2) until we obtain a sequence worth compressing with CID ≤ 1.

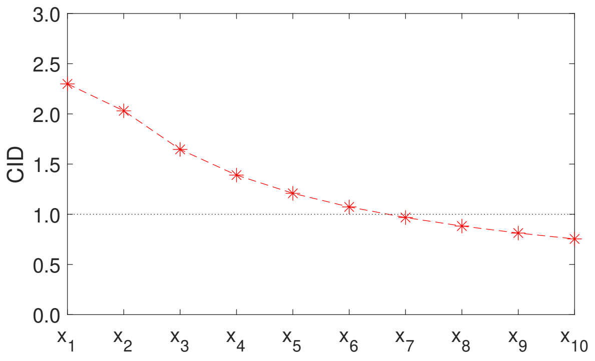

Fig. 2. CID of a sequence x_i consisting on repeating i times the sequence $x_1 = baababaa$.

This first happens for $x_7 = \overbrace{x_1 \ldots x_1}^{7\text{ times}}$, which has $L = 56$, $LZ77(x_7) = \{(b,0),(a,0),(2,1),(1,2),(1,3),(1,48)\}$, $C = 6$, therefore:

$$\mathcal{L}_{LZ77}(x_7) = 7.744 \qquad \text{and} \qquad \text{CID}(x_7) = 0.968\ . \quad (8)$$

This serves as an example of the fact that the more regularity (repeating sequences) in a file, the better the LZ77 compression algorithm works. Therefore, the more ordered an image is, the better the algorithm will be able to compress it.

This same process is used to compress and compute the CID of 2D images. However, in this case, the 2D image of pixels is converted to a 1D sequence. This can be achieved in various ways. Here, we will use the so called Hilbert scan, which preserves locality [4].

II. IMAGES WITH A FIXED NUMBER OF DOTS

As a first case, we measure the CID of black images with the same number of white dots, in different stages of progression from a square lattice to a random spatial distribution of dots. We expect the CID to be minimum in the lattice (maximum order) and maximum in the image of random dots (minimum order).

As a starting point, we generate 1000×1000 pixel images with 10000 dots in a square lattice (plus an image of only the background). We then generate a series of images where sections of the lattice are substituted by the same number of random dots; see Fig. 3 for three representative examples. This images are identified by it's *order parameter* χ_4, obtained as:

$$\chi_4 = \frac{1}{N}\sum_{i=1}^{N} e^{4i\theta_i}\ , \qquad (9)$$

where $N = 10000$ is the total number of dots and θ_i is the angle formed between the lines joining a dot and its two nearest neighbouring dots. We see that a square lattice has $\chi_4 = 1$, while a set of random dots has $\chi_4 = 0$.

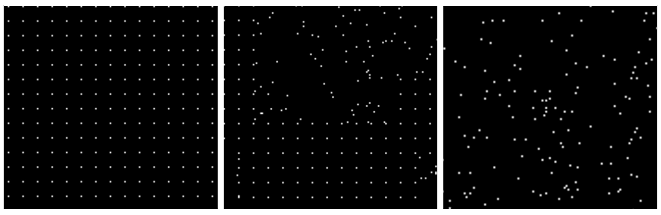

Fig. 3. Sections of the images corresponding to order parameter (from left to right) $\chi_4 = 1$, 0.5035 and 0

We then check that all images have the same size (5266 KB), and compress them to .zip format (which uses the LZ77 compressing algorithm). We then subtract the size of the compressed background from the size of the compressed images to compute the CID of only the dots. The resulting CID in terms of the order parameter is plotted in Fig. 4.

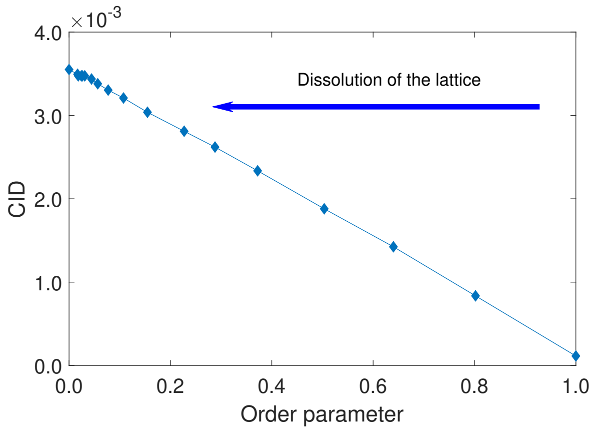

Fig. 4. Representation of the CID of a dissolving square lattice in terms of its order parameter.

We see that, at least in this case, the CID is closely related to the order parameter of a set of dots; in fact the CID decreases linearly as the number of dots in the ordered phase increases. Therefore the CID can be used to monitor the order in a given image.

III. IMAGES WITH A VARYING NUMBER OF DOTS

In this section we will study cases in which the number of dots in the image does not remain constant. We will compare how the CID of images with square lattices of increasing density and multiple iterations of a 2D fractal compare to images containing the same amount of randomly distributed dots. In all three cases we generate square images with 243 pixels on each side, with an increasing number of dots. We expect images with random organization of dots to have the maximum CID, and that the CID increases with the number of dots up to a maximum corresponding to half the pixels in the image.

A. Square lattice

For this case, we generate images with an increasing number of dots N in a square lattice. For each image we also generate an image with the same number of random dots. We then check that they have the same size, compress them, and subtract the size of the compressed background to compute the CID of the sets of dots.

We expect the evolution of the CID to follow a more or less flat line, as the compressed size of a regular sequence such as a square lattice should not depend on the density of the lattice. We can however expect some irregularities due to the fact that the compression algorithm has to convert a 2D image to a 1D sequence.

We see in Fig. 5 that the CID of dots in a square lattice increases for small N, but reaches a point where it does stabilize into a more or less flat line, decreasing slightly due to the fact that adding dots increases L in Eq. (2) while maintaining $\mathcal{L}$.

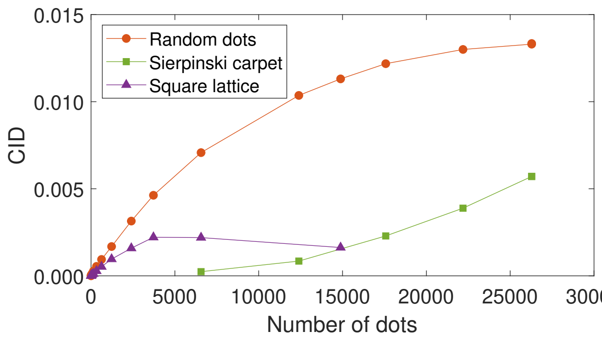

Fig. 5. CID of dots organized at random, in a square lattice and in a Sierpinski carpet, as a function of the number of dots.

We can better see how the square lattice compares to the random set in terms of the CID by plotting the ratio (CID of square lattice)/(CID of random dots) for sets with the same number of dots, as seen in Fig. 6. Here we see that after a maximum at small N, the ratio decreases. This maximum is most likely due to the conversion from the 2D image to a 1D sequence before applying the LZ77 algorithm, as it disappears when dealing with 1D square lattices (section V, appendix). This fact illustrates that the step of converting 2D images to 1D sequences is critical to optimally compress them.

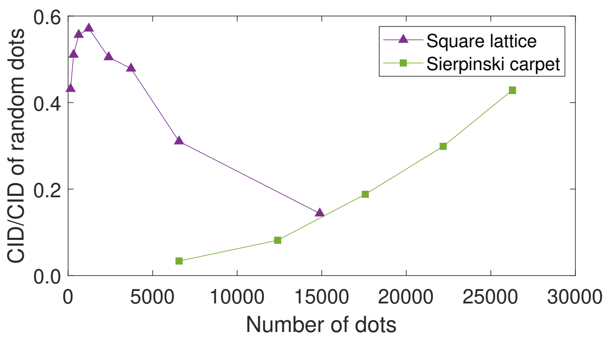

Fig. 6. Ratio CID/(CID of random dots) for sets with the same number of dots, for square lattices and the first five iterations of the Sierpinski carpet.

B. Fractal: Sierpinski carpet

Once checked how the CID varies in terms of N for a regular square lattice, we ask weather this behavior remains for more complex patterns, such as fractals, which are also “ordered” in that they posses scale invariance. In particular, we now study the case of the Sierpinski carpet, whose first three iterations are represented in Fig. 7.

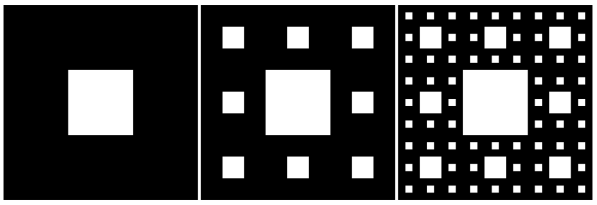

Fig. 7. First three iterations of the Sierpinski carpet.

We generate for each iteration an image of the carpet and another image with the same number of random dots, plus one background image. We then check that all images have the same size, compress them to .zip format and subtract the size of the compressed background to compute the CID.

We first notice that the CID is lower for the dots in the fractal than for the dots distributed at random (see Fig. 5). However, unlike for the square lattice, the CID for the fractal set increases with N, and this seems to happen in a potential manner. One can hypothesize that this is due to the fact that, even if the dots are “ordered”, this order becomes more complex with each iteration due to scale invariance, and that therefore, the amount of information per character increases in each iteration.

To better see how the CID of the fractal compares to the CID of random dots, we plot their ratio for sets with the same N. We find that the potential-like increase in the CID for successive iterations of the Sierpinski carpet is maintained; see Fig. 6. This suggests that the Hilbert scan together with the LZ77 compression algorithm is not effective for detecting scale invariance.

IV. CONCLUSIONS

- We have shown that the CID can be a good indicator of the order parameter in a process in which a set of randomly distributed dots orders itself into a square lattice. In the case of a 2D square lattice, the CID increases with the number of dots until it reaches a point where it starts to decrease slowly.
- The observed increase is related to how we transform the 2D image into a 1D sequence before compression, highlighting the significance of this step.
- In the case of the Sierpinski carpet, which we have taken as an example of a scale invariant fractal, the CID increases with the number of dots, but never reaching the CID of the same number of randomly distributed dots. This distinguishes this case from the periodic (such as 111111...), quasiperiodic (such as the *Fibonacci word*) and pseudorandom (such as the digits of π) sequences studied in the supplementary information of [1]. Compression of fractals is thus interesting and will be the subject of further study in the soft condensed matter lab led by Prof. Fernandez-Nieves.

V. APPENDIX: 1D SQUARE LATTICE

To check weather the initial increase with N in the case of the square lattice are due to the conversion of the 2D image to a 1D sequence, we repeat the process outlined in section III A with images only one pixel wide. The results are plotted in Figs. 8 and 9. In Fig. 8 we see that the CID behaves similarly to the way it did for the case of the 2D image, starting with a small increase followed by a stabilization with a slight downward slope. In Fig. 9 we observe that the trend is similar to that series corresponding to a square lattice in Fig. 6, but the maximum observed there does not appear. This suggests that the maximum may have been caused by the scanning of the image to convert it to a 1D sequence before compression.

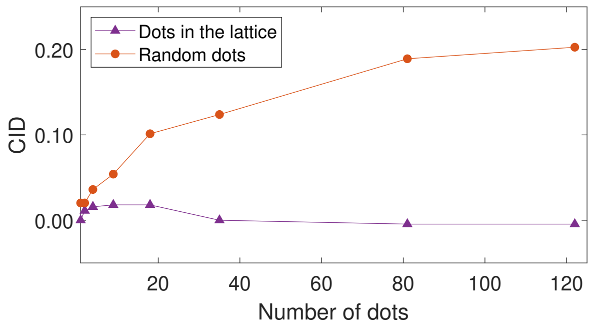

Fig. 8. CID of sets of dots in a 1 pixel wide image, ordered at random and in a square lattice, as a function of the number of dots in the set.

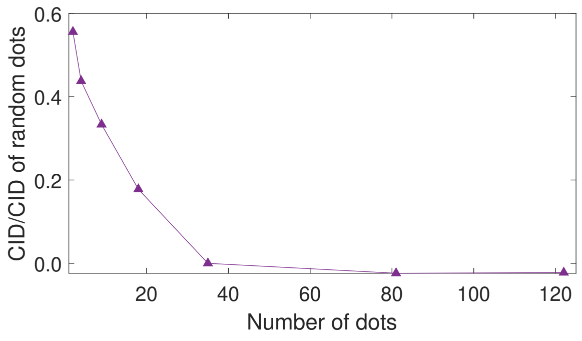

Fig. 9. Ratio CID/(CID of random dots) for sets with the same number of dots, for 1D square lattices.

Acknowledgments

I would like to thank my advisors, Alberto Fernandez-Nieves and Ramon Planet, for their guidance and input. I am also grateful to my friends for keeping me sane in this times of covid restrictions, and to my family for their unconditional support.

[1] S. Martiniani, P. M. Chaikin, D. Levine: Quantifying Hidden Order out of Equilibrium, *Phyisical Review X* **9**, 01131 (2019).
[2] C. E. Shannon:A Mathematical Theory of Communication, *Bell System Technical Journal* **27**, 379 (1948).
[3] A. Lempel, J. Ziv: A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory* **23**, 337 (1977).
[4] A. Lempel, J. Ziv: Compression of Two-Dimensional Data, *IEEE Transactions on Information Theory* **32**, 2 (1986).
[5] P. C. Shields: Performance of LZ Algorithms on Individual Sequences, *IEEE Transactions on Information Theory* **45**, 1283 (1999).