Chapter 6
The Algorithmic Basis of Spatial Complexity

-What is the wisest?
-The Number.
-And what is the second wisest?
-Assigning names to things
“Τί τὸ σοφώτατον; Ἀριθμός.
Τί τὸ δεύτερον σοφώτατον;
Τὸ τοῖς πράγμασι τὰ ὀνόματα τιθέμενον”
(Iamblichus, 245–325 b.C., VP82, 47.17–19)

Abstract Building on previous concepts and measures, two metrics of spatial complexity of small 2d maps are proposed here: C_{P1} and C_{P2}. Both are applicable to square maps (binary or not), so long as each square cell of a map can have a one-to-one correspondence to one symbol (taken from an alphabet of symbols). These symbols can be used to represent as many colors, categories etc. as the map requires, on the basis of a conventional “scanning” run-length encoding sequence. The first metric gives an evaluation of spatial complexity on the basis of blocks of cells with repeating patterns that are identified following an iterative process. The second metric is calculated on the basis of vertical and horizontal pairwise comparisons of successive adjacent strips of the map or surface. Both metrics are easy to compute, assume integer values and their magnitudes are comparable for small maps.

Keywords Spatial complexity · Map Complexity · Geography and Complexity · Edit distance · run-length encoding · Algorithmic complexity · Geocomputation

6.1 The Language of Space

lopadotemahoselahogeleokraniolipsanodimypotrimatosilfio
laravomelitokatakehymenokihlepikosyfofatoperisteralektryonopto
kefalliokiglopeliolagosireovafitraganopterygon
“λοπαδοτεμαχοσελαχογαλεοκρανιολειψανοδιμυποτριμματοσιλφιο
λαραβομελιτοκατακεχυμενοκιχλεπικοσσυφοφαττοπεριστεραλεκτρυονοπτο
κεφαλλιοκιγκλοπελειολαγωσιραιοβαφητραγανοπτερύγων”
(Aristophanes, 427-386 b.C., “Ecclesiazusae”, line 1163)

F. Papadimitriou, *Spatial Complexity*, https://doi.org/10.1007/978-3-030-59671-2_6

Aristophanes, the ancient greek author of the first comedies in the history of theatre wrote this 171 letters-long word that probably used to be the longest known word throughout the ages, supposedly referring to a name of a dish, a kind of a fricassée containing some 15 ingredients: clearly a “complex” word, describing an equally complex set of ailments recommended for the creation of this dish. But it is not human language alone that can create very long words. Metaphorically, space also “speaks” immensely lengthier “words”, so long as one is able to “listen” to them. Geometric, topological and other mathematical aspects of spatial complexity may *indicate* whether a spatial object is endowed with a higher or lower spatial complexity, but they do not constitute measures of spatial complexity themselves. This leads us to a central issue with respect to the measurement of spatial complexity: Since complexity is different than entropy, or any other indicator of complexity, any measure of complexity should be a measure of *complexity* per se, backed by the mathematical literature of complexity. Such a measure should (under certain topological restrictions) be of general applicability to two-(and possibly higher)-dimensional smooth surfaces. A rasterization/digitization significantly facilitates the measurement of spatial complexity, since even maps of points and lines can (under certain conditions) be rasterized and, subsequently, symbols of cover or color can be assigned to each cell of the resulting square grid. The term “raster” that is commonly used in GIS signifies geoprocessing of spatial elements on the basis of squares (pixels). In fact, it is a common practice in geospatial analysis to “rasterize” all spatial elements, that is to attribute to square cells or pixels even linear or point data. The long experience gained from the use of geographic technologies (such as Geographical Information Systems “GIS” and satellite image processing) is particularly useful in this respect. After experimenting with various geo-encoding models, digital arrays of pixels are used to represent either “vector” (0-and-1-dimensional) or “raster” (2-dimensional) spatial entities. The rasterization is easily created by enlarging or reducing the map size as appropriate and assigning a special color to each one of its cells (Fig. 6.1). Whether the term “cell” or “pixel” is used, it

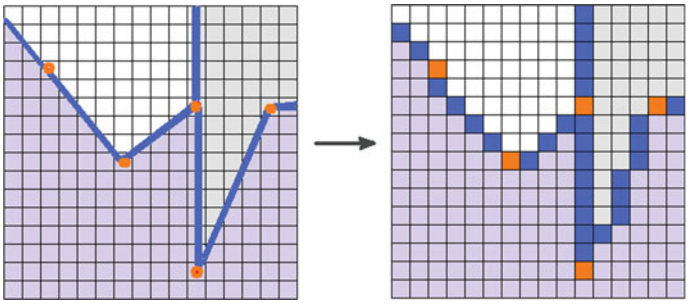


Fig. 6.1 Rasterization” is a widely used process in image analysis, by which points, lines and areas (0,1,2-dimensional spatial elements respectively) are translated to square cells. In this way, all elements of a map, whether points or lines or areas, can be converted to cellular (square) representations, with accuracy depending on the pre-decided resolution of the square map that is to be constructed

invariably concerns discrete square cells on a map of equal square partitions. In the science and practice of image processing, continuous functions are routinely used in image analyses. For instance, the convolution of the map $f(x, y)$ with the map $g(x, y)$ is defined as a function:

$$h(x, y) = \iint f(x_1, y_1)g(x - x_1, y - y_1)dx_1dy_1 \tag{6.1}$$

Thus, analytic expressions of known functions are used in their discrete form, such as, for instance, the discrete Fourier Transform of a 2d array $f(x, y)$ of size mxn:

$$f(u, v) = \frac{1}{\sqrt{mn}} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} f(x, y) e^{-2i\pi\left(\frac{xu}{m} + \frac{vy}{n}\right)} \tag{6.2}$$

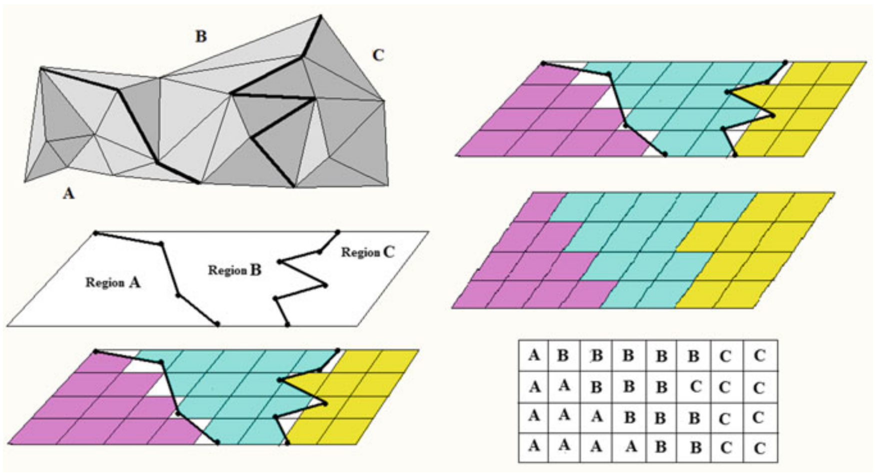
Equivalently, the form of a closed curve on the plain in polar coordinates (with distance D from the barycentre and angle θ with respect to the horizontal axis) is given in terms of a typical Fourier expression:

$$D = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(n\theta) + b_n \sin(n\theta)] \tag{6.3}$$

The analytic approach applies when the curve is “not too rough”. When the curve’s shape is not so, the analytic approach is of limited applicability, because it renders more than one values for the same angle θ. But when the map is examined in discrete cells, there is always one single value attributed to each cell. Further, the numerical application of such functions practically presupposes discrete spatial domains, so some “discretization” or “rasterization” process is eventually unavoidable for all kinds of computerized image processing.

A *square “map”* is a binary or multicolored planar surface, which has been constructed by any means (satellite or aerial photography, plain photography, field observations, cartographic processing etc.), and can be described mathematically by a function on the discrete plain Z^2 defined as $f{:}Z^2 \rightarrow \{0, 1, 2, ..., V\}$, where V is the number of colors (“covers”) appearing on the map. In example, a “binary map” is defined by a function $f{:}Z^2 \rightarrow \{0, 1\}$, where $f(0) = \text{white}$ and $f(1) = \text{black}$. In the same way, for higher spatial dimensions, the mapping function f is defined in Z^3 (for three dimensions, in which case, the equivalent of the 2d pixel is a 3d a “voxel”), in Z^4 (for four dimensions, in which case, the equivalent is a 4d a “doxel”), or in Z^n in general.

Obviously, an alphabet of symbols may be chosen so as to adequately represent (under certain conventions) spatial allocations (i.e. colors) on a rasterized map (Fig. 6.2). Yet, we should not lose sight from the fact that both spatial and semantic



A	B	B	B	B	B	C	C
A	A	B	B	B	C	C	C
A	A	A	B	B	B	C	C
A	A	A	A	B	B	C	C

Fig. 6.2 If a three-dimensional surface that is covered by three types of cover (A, B, C) is projected on a 2-dimensional space and then encoded by using an alphabet of three symbols (A, B, C), it eventually yields a map, or, alternatively, a 2d linguistic description of this surface

generalizations made on the initial image, map or surface, affect the accuracy, precision and effectiveness of our assessments of spatial complexity. It is easy to visualize how a map configuration can eventually be converted into a linguistic or numerical description with a "run length encoding", where the spatial sequence of a map's allocation of covers (i.e.landscape or soil types, kinds of land use, geomorphic features) corresponds to symbols of an alphabet. Similarly, three-dimensional spaces can be converted to linguistic descriptions by using some (appropriate for that purpose) alphabet.

Even maps of points can be converted to maps of square cells. This process involves the use of *Voronoi polygons* (or Voronoi cells, or Thiessen polygons) that make a one-to- one correspondence of spatial regions to points. Indeed, areas covered by Voronoi polygons can be converted to a map of square cells (Fig. 6.3). Following similar procedures, lines and curves may be made equivalent to areas, which are eventually displayed as raster maps.

Practically, re-arranging a two-dimensional image as an one-dimensional line has long proven useful in computational analyses, particularly for image classification purposes (Seiffert and Michaelis 1997). It is now natural to ask "how can we measure the complexity of spatial forms of a binary map (or landscape with two land cover types), *without* considering spatial distributions of *particular* physical entities in it?".

An index of spatial complexity (I_S) was initially proposed by Papadimitriou (2002), measuring the number of common boundaries between different kinds of map cover per unit area. Later, a different index was proposed (Papadimitriou 2009), based on Levenshtein distances (Levenshtein 1966) which measure the minimum distance between two strings of symbols as the *minimum* number of the three basic operations necessary to convert one string into another (deletions d, substitutions s,

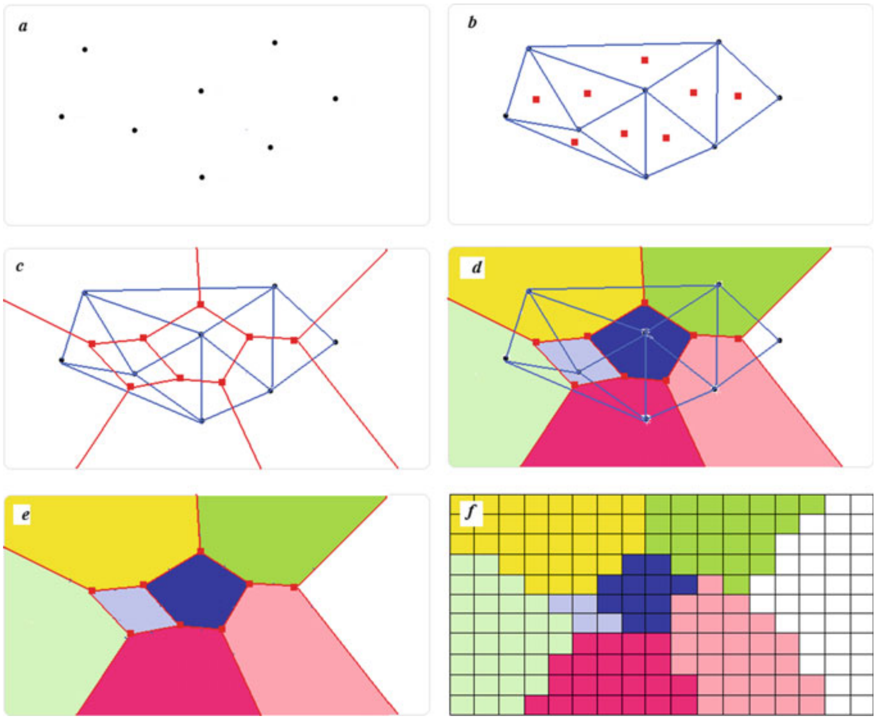


Fig. 6.3 Even spatial distributions of points (**a**) can be converted to areas of square cells (*f*). Voronoi polygons can be used to convert point data to maps of square cells. Beginning with a map (**a**) with point data only, a triangulation of the space is created by connecting the points (**b**). Next, the barycentres of all triangles are defined (**c**). Consequently, the barycentres are joined by lines and these lines are extended as appropriate (**d**) up to the map's boundaries. The number of the resulting Voronoi polygons (**e**) is equal to the number of initial points (8 in this example) and, eventually, the map of Voronoi polygons is rasterized (**f**)

additions t): $a = min\{d + s + t\}$. It was shown (Papadimitriou 2009) that this algorithm can be spatialized and that it can be used to calculate the spatial complexity of a map by means of the formula:

$$C_L = \frac{1}{uv} \sum_{\substack{i=1 \\ j=1}}^{\substack{j=v-1 \\ i=u-1}} \alpha_{ij} \tag{6.4}$$

where u = rows, v = columns of cells of the map, and α_{ij} are the entries of the matrix of the edit distances which are derived by comparing parallel and adjacent strips of landscapes. This formula converts the one-dimensional edit algorithm from a measure of minimum description of string differences to a measure of minimal length description of spatial differences and it is for this reason that it serve as a measure of spatial complexity (Papadimitriou 2009).

Another index (K_S) was later suggested (Papadimitriou 2012), based on the concept of Kolmogorov complexity (Kolmogorov 1965, 1986), defined as

$$K_s = \min\big((U(x),\ U(y), U\big(x'\big), U\big(y'\big)\big) \tag{6.5}$$

where $x =$ the original horizontal string, $y =$ the compressed horizontal string, $x' =$ the vertical string, $y' =$ the vertical compressed string.

The notion of "Kolmogorov complexity" was used to evaluate the algorithmic complexity of a finite string of symbols (e.g. ABDDCBFGAAG… or 0,111,001,010…), which is defined on a finite alphabet, such as {A, B, C, D … Z} or {0, 1, 2, …}, or {black, white}, or {0, 1} or any other.

The problem with Kolmogorov complexity is that, in the general case, it is non-computable, so alternative approaches have appeared in the literature. Fortnow et al. (2011) sought possible answers to the question of extracting the Kolmogorov-randomness from a string and showed that we can extract Kolmogorov complexity with only a constant number of bits of additional information in a polynomial-time computable procedure. Furthermore, it is possible to apply a modified notion of Kolmogorov-complexity to "short" strings, as will be explained in the next section.

6.2 Metrics of Spatial Complexity

"Tutor: What is twice three?

Scout: What's a rice tree?

Sub-scout: When is ice free?

Sub-Sub-scout: What's a nice fee?"

(Lewis Carroll, 1832–1898)

With these premises from previous research, two new and more effective metrics of spatial complexity will be defined here. "*Fivos Papadimitriou spatial complexity metric 1*" (referred to as C_{P1} hereafter) and the explicit procedure for its computation will be described next.

This metric makes use of an initial concept by Papentin (1973, 1980, 1982, 1983a, b), who suggested that it is possible to evaluate the complexity (not spatial) of a small string by identifying patterns in it and then the string's complexity is defined as the length of the compressed string. However, he did not present any algorithmic procedure for identifying such patterns and his method was not spatial either. Also, the measure previously defined on the basis of Kolmogorov complexity (K_s) that was described by Papadimitriou (2012) did not comprise any algorithmic procedure for finding patterns in a string. And one further difference between the C_{P1} and the procedures developed by both Papentin (1973) and Papadimitriou (2012) mentioned previously, is that C_{PI} is much more restrictive, because it is defined as the minimum algorithmic complexity taking into account *all possible orientation changes* of the square map under consideration.

Consequently C_{PI} differs:

(I) from the Papadimitriou (2012) metric, because: (a) it follows an exact algorithmic procedure for finding patterns, (b) it is based on Papentin complexity rather than on Kolmogorov complexity and (c) it is calculated as the minimum from all possible symmetric transformations (rotations, inversions etc.) of the original map.
(II) from the Papentin (1973) metric, because: (a) it is spatial (Papentin's measure was not) and (b) it is associated with an exact procedure for the identification of patterns.

In computing the string length, the following conventions are used:

(a) If a symbol is repeated twice or more consecutively, then it can be written with a power of the times it appears consecutively, i.e. the string AAAAAAABA can be shortened to A^7BA.
(b) If a block of symbols repeats itself consecutively, then parentheses can be used and powers indicating the number the block of symbols is repeated, i.e. the string ABCABCABCABCABCABC can be represented as $(ABC)^6$.
(c) Lambda (λ) symbolizes a block of consecutive symbols, used to indicate compression. In the compressed string, the lambda block of symbols is separated by a comma (,) from the λ-compressed string. The comma is also counted as a separate symbol in the final string length calculation. For instance, the initial 30 symbols-long string:

BWBBWBBWBBWBBWBBWBBWBBWBBWBWWW
can be compressed by using the compression $\lambda = BWB$ that has 5 symbols to create the following short string: λ^9W^3.

The above notation would be incomplete without citing the compression $\lambda = BWB$, because λ could symbolize just any block of symbols. Hence, the compression of the initial string should be written as: $\lambda = BWB$, λ^9W^3 which consists in 10 symbols (5 symbols for the $\lambda = BWB$ plus one symbol for the comma, and 4 symbols additionally for the short string).

Thus, without any loss of information, it can be said that this compressed 10-symbols long string can fully encode the initial 30-symbols long string. Hence, the formula for the calculation of C_{PI} is simply:

$$C_{PI} = \min_{D_4}\{S_b\} \tag{6.6}$$

where $b = 1, 2, 3, \ldots L$ and S_b are all the compressed strings generated by blocks (b) of symbols, from $b = 1$ symbol up to $b = L$ symbols, and L is the initial length of the string and *for all possible map orientations* derived from the symmetry group of the square (D_4).

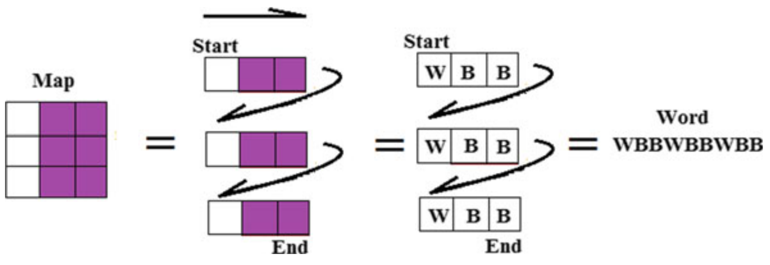


Fig. 6.4 Converting a map to a "word" (a string of symbols) for the calculation of its spatial complexity

Let us now see exactly how the C_{PI}-complexity of a small map can be calculated. Beginning by scanning cells from top to bottom and from left to right (as one reads the lines of a book), consider a 3 × 3 binary map (Fig. 6.4) with initial string (with the position of each symbol numbered below): W B B W B B W B B.

The procedure begins by first trying possible compressions using blocks of two symbols, beginning from the two first symbols on the left of the string and ending with the two last symbols (the abbreviation CSL stands for compressed string length):

(a) Blocks of 2 symbols

Symbols' numbers	λ	Compression	Short string	CSL
1 + 2	WB	λ = WB	λ = WB, λBλBλB	11
2 + 3	BB	λ = BB	λ = BB, WλWλWλ	11
3 + 4	BW	λ = BW	λ = BW, WBλBλB^2	12
4 + 5	WB	λ = WB	λ = WB, λBλBλB	11
5 + 6	BB	λ = BB	λ = BB, WλWλWλ	11
6 + 7	BW	λ = BW	λ = BW, WBλBλB^2	12
7 + 8	WB	λ = WB	λ = WB, λBλBλB	11
8 + 9	BB	λ = BB	λ = BB, WλWλWλ	11

and continues with block sizes increasing by one, up to $L - 1$, as follows:

(b) Blocks of 3 symbols

Symbols'numbers	λ	Compression	Short string	CSL
1 + 2 + 3	WBB	λ = WBB	λ = WBB, $λ^3$	8
2 + 3 + 4	BBW	λ = BBW	λ = BBW, W$λ^2B^2$	11
3 + 4 + 5	BWB	λ = BWB	λ = BWB, WB$λ^2$B	11
4 + 5 + 6	WBB	λ = WBB	λ = WBB, $λ^3$	8

(continued)

(continued)

Symbols'numbers	λ	Compression	Short string	CSL
$5+6+7$	BBW	λ = BBW	λ = BBW, Wλ^{2}B^2	11
$6+7+8$	BWB	λ = BWB	λ = BWB, WBλ^{2}B	11
$7+8+9$	WBB	λ = WBB	λ = WBB, λ^3	8

(c) Blocks of 4 symbols

Symbols' numbers	λ	Compression	Short string	CSL
$1+2+3+4$	WBBW	λ = WBBW	λ = WBBW, λB^{2}WB^2	13
$2+3+4+5$	BBWB	λ = BBWB	λ = BBWB, WλBWB^2	13
$3+4+5+6$	BWBB	λ = BWBB	λ = BWBB, WBλWB^2	13
$4+5+6+7$	WBBW	λ = WBBW	λ = WBBW, λB^{2}WB^2	13
$5+6+7+8$	BBWB	λ = BBWB	λ = BBWB, WλBWB^2	13
$6+7+8+9$	BWBB	λ = BWBB	λ = BWBB, WBλWB^2	13

(d) Blocks of 5 symbols

Symbols' numbers	λ	Compression	Short string	CSL
$1+2+3+4+5$	WBBWB	λ = WBBWB	λ = WBBWB, λBWB^2	13
$2+3+4+5+6$	BBWBB	λ = BBWBB	λ = BBWBB, WλWB^2	13
$3+4+5+6+7$	BWBBW	λ = BWBBW	λ = BWBBW, WBλB^2	13
$4+5+6+7+8$	WBBWB	λ = WBBWB	λ = WBBWB, λBWB^2	13
$5+6+7+8+9$	BBWBB	λ = BBWBB	λ = BBWBB, WλWB^2	13

(e) Blocks of 6 symbols

Symbols' numbers	λ	Compression	Short string	CSL
$1+2+3+4+5+6$	WBBWBB	λ = WBBWBB	λ = WBBWBB, λWB^2	13
$2+3+4+5+6+7$	BBWBBW	λ = BBWBBW	λ = BBWBBW, WλB^2	13
$3+4+5+6+7+8$	BWBBWB	λ = BWBBWB	λ = BWBBWB, WBλB	13
$4+5+6+7+8+9$	WBBWBB	λ = WBBWBB	λ = WBBWBB, λWB^2	13

(f) Blocks of 7 symbols

Symbols' numbers	λ	Compression	Short string	CSL
$1+2+3+4+5+6+7$	WBBWBBW	λ = WBBWBBW	λ = WBBWBBW, λB^2	13
$2+3+4+5+6+7+8$	BBWBBWB	λ = BBWBBWB	λ = BBWBBWB, WλB	13
$3+4+5+6+7+8+9$	BWBBWBB	λ = BWBBWBB	λ = BWBBWBB, WBλ	13

(g) Blocks of 8 symbols

Symbols' numbers	λ	Compression	Short string	CSL
$1+2+3+4+5+6+7+8$	WBBWBBWB	λ = WBBW, BBWB	λ = WBBWBBWB, λB	13
$2+3+4+5+6+7+8+9$	BBWBBWBB	λ = BBWB, BWBB	λ = BBWBBWBB, Wλ	13

From these calculations, it follows that a string of 8 symbols is the shortest one. This shortest string faithfully represents the original string of 9 symbols without loss of information. But 8 is *not* the value of C_{P1}. From all possible rotations of the initial map, a rotation by 90° to the right produces a compressed string shorter than 8 symbols (Fig. 6.5), so the resulting string consists in 4 symbols only: W^3B^6 and hence we eventually conclude that $C_{P1} = 4$ for this map.

Besides C_{P1}, yet another algorithmic metric will be defined here, what can be called the "*Fivos Papadimitriou spatial complexity metric 2*", symbolized by C_{P2}, based on modifications of the formula that was derived by Papadimitriou (2009), so as to make it much easier to calculate and more appropriate for small square maps. The division by the product of rows and columns will be omitted here so it will be only the sum of the total Levenshtein distances; not their average as in Papadimitriou (2009) and is thus calculated as:

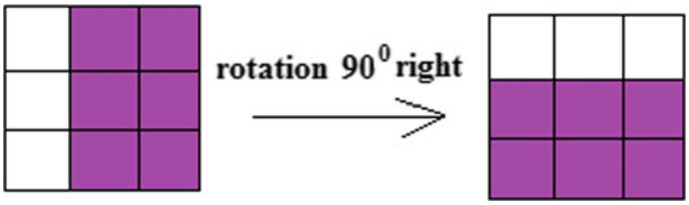


Fig. 6.5 Orientation matters in the calculation of spatial C_{P1} -complexity: rotating the initial binary map (left, with string $WB^2WB^2WB^2$) by 90° to the right results in the string W^3B^6, of which the complexity is readily calculated to be equal to 4 (in contrast to the original map, for which a lengthy calculation process was required, resulting to a complexity value equal to 8)

$$C_{P2} = \sum_{\substack{i=1\\j=1}}^{\substack{j=v-1\\i=u-1}} \alpha_{ij} \tag{6.7}$$

where a_{ij} are the in-between i-rows ($i = 1, 2, \ldots, u$) and in-between j-columns ($j = 1, 2, \ldots, v$) Levenshtein distances which result from pairwise comparisons of strips of the surface considered, successively, and for all the strips covering the surface. Thus, the C_{P2} metric differs from the C_L metric in the following:

(a) it requires less calculations to compute compared to C_L for square maps since no averaging for columns and rows is required;
(b) it assumes only positive integers as values;
(c) it can be used to derive values that are comparable to C_{P1} for *small* maps (notice, for instance, that, for 3×3 binary maps the maximum value of both C_{P1} and C_{P2} is 8).

However, instead of seeking a minimum length of string number, C_{P2} seeks a minimum operations number.

Another difference is that while C_{P1} is sensitive to symmetric transformations (rotations, inversions), C_{P2} is not (for an example calculation of C_{P2} see Fig. 6.6).

Complexity calculations of either C_{P2} or C_{P1} may also be carried out (as the case may be) according to the "*entropy encoding*" technique that is widely used in image processing for many electronics applications. This is a procedure of lossless data compression, derived by encoding the image components in a "zigzag" fashion, as shown in Fig. 6.7. It begins from one of the corner cells of the image and then proceeds in a zigzag manner up to the opposite corner of the image, by recording the cell category of each cell encountered in this zigzag process. Yet, there are several other scanning methods that are used in optoelectronics and other technologies.

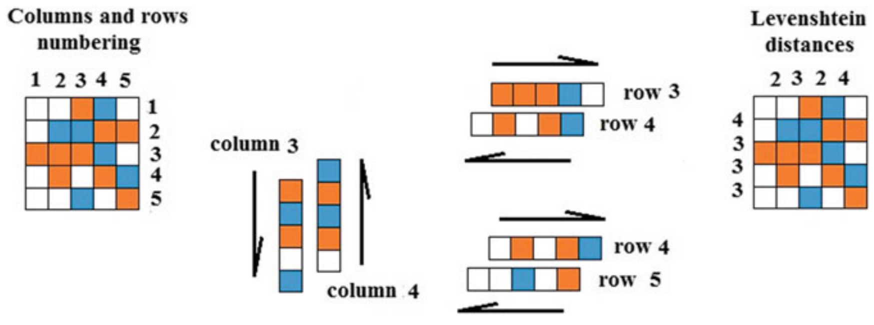


Fig. 6.6 A 5 × 5 square map with three cover types showing how C_{P2} and C_{P1} values are comparable for small square maps. First, notice that C_{P1}=22 because the map's irreducible string is: $A^2CBA^2B^2C^5$ BA^2CACBA^2BAC. Some characteristic column and row slides are depicted to illustrate the calculations of C_{P2} per pair of rows and columns. Eventually, the calculations lead to C_{P2} = 24 (that is an agreement by more than 90% of the two metrics for this map)

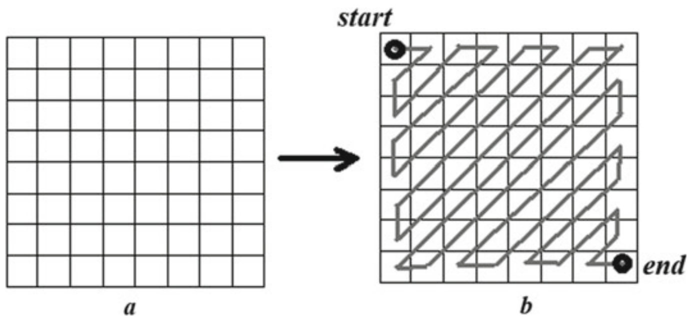


Fig. 6.7 The "entropy encoding" technique applied on a rasterized 8 × 8 initial image (**a**): beginning with the top left corner cell (**b**), continuing with the second upper right cell, then carrying on diagonally until the entire image is "encoded", down to the bottom right cell

Following such other methods will inevitably result in different strings of symbols than the raster scanning (Fig. 6.8).

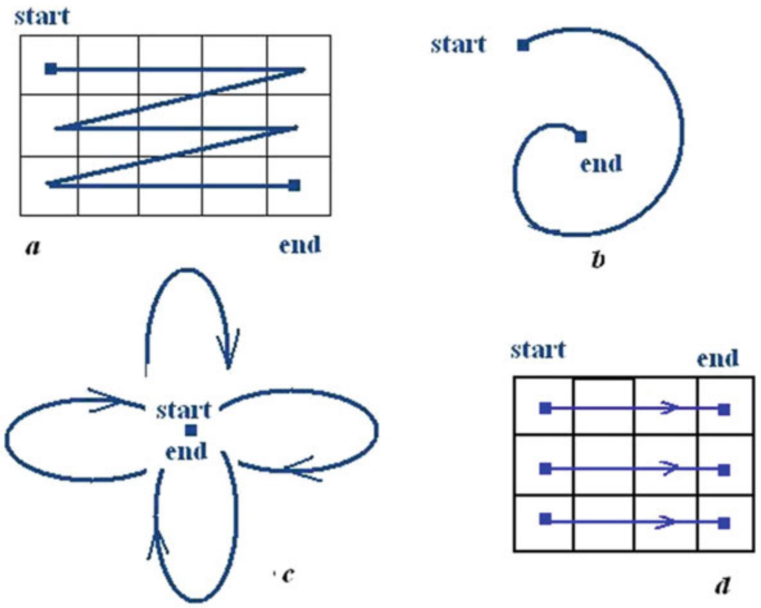


Fig. 6.8 Alternative scanning methods used in various technologies: raster scanning (**a**) that is most commonly used, rosette scanning (**b**), spiral scanning (**c**), linear array scanning (**d**)

6.3 Extrema of Spatial Complexities C_{P1} and C_{P2}

Since things may differ from one another,
to a higher or a lesser degree of difference,
there is also a maximal difference,
and that one I call contrariety
"ἐπεὶ δὲ διαφέρειν ἐνδέχεται ἀλλήλων τὰ διαφέροντα πλεῖον καὶ ἔλαττον,
ἔστι τις καὶ μεγίστη διαφορά, καὶ ταύτην λέγω ἐναντίωσιν"
(Aristotle, 384-322 b.C., "Metaphysics", 10.1055a)

It is easy to verify that the C_{P2} of any multicolored square map of size n ranges in between 2 and $2n - 2\sqrt{n}$. Indeed, the spatial complexity of any multicolored map is lowest when there is only one different cell at any one of the map's corners. In this case, the map's C_{P2} is at least 2, because the corner cell has only two borders with two white cells. So the pairwise comparisons of the 1st row with the 2nd row and the 1st column with the 2nd column will record this difference in C_{P2}. As the size of a multicolored map increases, all cells will have different colors, so the maximum C_{P2} produces $\sqrt{n}$ by pair of rows, as well as by per pair of columns. Hence there are $\sqrt{n} - 1$ comparisons of map rows, plus $\sqrt{n} - 1$ comparisons of map columns, with $\sqrt{n}$ differences per comparison. Consequently, there are $2\sqrt{n}(\sqrt{n} - 1)$ differences between cells, compared by columns and by rows pairwise and thus,

$$2 \leq C_{P2} \leq 2n - 2\sqrt{n} \tag{6.8}$$

Expectedly, the max C_{P2} grows with increasing n.

As concerns C_{P1}, if a square binary map of size n has only one black cell and that particular cell is located at anyone of the map's corners, then its C_{P1} is minimum and equal to:

$$C_{P1\,\mathrm{min}} = 3 + \lfloor \log_{10}(n) \rfloor \tag{6.9}$$

In fact, for any binary map, the lower complexity is attained when there is only one black cell at any one of the map's corners. In this case, the map's C_{P1} is

$$C_{P1}\{\mathrm{BW}^{\mathrm{u}}\} = 3 \tag{6.10}$$

where u is the number of white cells that follow in the string.

If there are up to u = 9 white cells (that is a map with $n = 10$ cells) then the string still has the same minimum:

$$C_{P1}\{\mathrm{BW}^{9}\} = 3 \tag{6.11}$$

and thus, in this case, the minimum C_{P1} is also $C_{P1} = 3$.

If there are u $=$ 10 white cells following the corner black cell, then the string has $n = 11$ so that:

$$C_{P1}\left\{\mathrm{BW}^{10}\right\} = 4 \tag{6.12}$$

In fact, C_{PI} will remain equal to 4 for all strings of length up to $n = 99$:

$$C_{P1}\left\{\mathrm{BW}^{99}\right\} = 4 \tag{6.13}$$

However, with one more cell ($n = 100$), the string will need three digits to be described (of which two characters only to represent the power of W), so C_{PI}-complexity becomes:

$$C_{P1}\left\{\mathrm{BW}^{100}\right\} = 5 \tag{6.14}$$

Thus, as the map size increases, string codes are determined by a power of 10 (hence by the logarithm of 10) of the exponent u. As $\log_{10}n$ also assumes non-integer values, the floor function $\lfloor\rfloor$ of the logarithm of n applies, so C_{PI} has a bound of its minimum value, depending on the map size n, defined as:

$$C_{P1\,\mathrm{min}} = 3 + \left\lfloor \log_{10}(n) \right\rfloor \tag{6.15}$$

or, equivalently,

$$C_{P1} = 2 + \left\lceil \log_{10}(n) \right\rceil \tag{6.16}$$

where $\lceil\rceil$ is the ceiling function.

Notice that the floor function is a stepwise function, meaning that the maximum values of the minima of $C_{PI\ min}$ assume integer values only, whose ranges increase slowly with n. To verify, consider the following examples, with increasing mapsize n:

$$\text{For}\, n = 81,\, C_{P1\ \mathrm{min}} = 3 + \left\lfloor \log_{10}(81) \right\rfloor = 3 + \lfloor 1.81 \rfloor = 4 \tag{6.17}$$

$$\text{For}\, n = 121,\, C_{P1\ \mathrm{min}} = 3 + \left\lfloor \log_{10}(121) \right\rfloor = 3 + 2 = 5 \tag{6.18}$$

$$\text{For}\, n = 625,\, C_{P1\ \mathrm{min}} = 3 + \left\lfloor \log_{10}(625) \right\rfloor = 3 + \lfloor 2.25 \rfloor = 5 \tag{6.19}$$

Thus if any one of the four corner cells of a binary square map is black, whatever the map size, the minimum C_{PI} complexity can not be higher than $3 + \left\lfloor \log_{10}(n) \right\rfloor$.

More generally, for the entropy class 1, the minimum C_{PI} complexity is attained when there is only one black cell, located at anyone of the map's corners, so for all binary maps

$$C_{P1\ \min} = 3 + \lfloor \log_{10}(n) \rfloor = 2 + \lceil \log_{10}(n) \rceil \tag{6.20}$$

holds, whatever the value of n is.

Given these, it can be proven that if all the black cells of a binary square map appear as a single block of ω-consecutive black cells located after m-consecutive white cells anywhere in the map except for its corners, then the C_{P1} is:

$$C_{P1} = 3 + \lceil \log_{10}(m) \rceil + \lceil \log_{10}(\omega) \rceil + \lceil \log_{10}(n - m - \omega) \rceil \tag{6.21}$$

Indeed, for any black cell located anywhere in the binary map but the corner, the string has the general form:

$$\mathrm{W}^m \mathrm{B} \mathrm{W}^{n-m-1} \tag{6.22}$$

where m is the number of cells preceding the black cell and therefore the C_{P1} of such strings is:

$$C_{P1} = 3 + \lceil \log_{10}(m) \rceil + \lceil \log_{10}(n - m - 1) \rceil \tag{6.23}$$

For three black consecutive cells located anywhere in the map except for the corner, the string is:

$$\mathrm{W}^m \mathrm{B}^3 \mathrm{W}^{n-m-3} \tag{6.24}$$

which has a C_{P1} equal to:

$$C_{P1} = 5 + \lceil \log_{10}(m) \rceil + \lceil \log_{10}(n - m - 3) \rceil \tag{6.25}$$

Consequently, for a block of ω consecutive black cells located anywhere in the map but the corner, the string is:

$$\mathrm{W}^m \mathrm{B}^{\omega} \mathrm{W}^{n-m-\omega} \tag{6.26}$$

and hence:

$$C_{P1} = 3 + \lceil \log_{10}(m) \rceil + \lceil \log_{10}(\omega) \rceil + \lceil \log_{10}(n - m - \omega) \rceil \tag{6.27}$$

This formula gives the complexity of all binary maps with entropy class 1 with black cells appearing as a single block anywhere in the binary map except for the map's four corner cells.

As an example, consider a binary 10×10 map (Fig. 6.9), with the following incompressible string of 9 symbols:$\mathrm{W}^{36}\mathrm{B}^{25}\mathrm{W}^{39}$. This map has a single block of 25

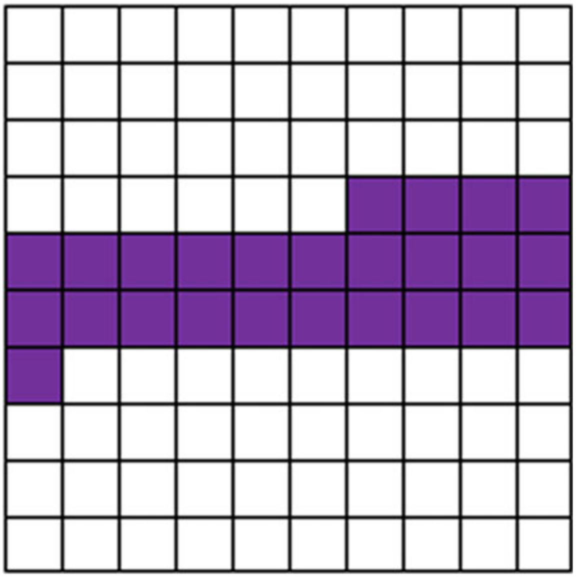


Fig. 6.9 A 10 × 10 binary map, with a single block of 25 consecutive black cells located after 36 initial white cells and 39 white cells that follow immediately after the block of black cells

consecutive black cells located after 36 initial white cells and 39 white cells that follow immediately after the block of black cells.

Applying the previous formula, the C_{P1} complexity of this map is:

$$\begin{aligned} C_{P1} &= 3 + \lceil \log_{10}(36) \rceil + \lceil \log_{10}(25) \rceil + \lceil \log_{10}(39) \rceil \\ &= 3 + 2 + 2 + 2 = 9 \end{aligned} \tag{6.28}$$

Due to to the periodic nature of the floor and ceiling functions, the following proposition is interesting to consider, which holds when the logarithm of n with base 10 is non-integer (as most often the case is).

Also, it if and only if $\lfloor \log_{10}(n) \rfloor \notin Z$**,** then a lower bound of the minimum C_{P1} of a square binary map is $2 + \log_{10} n$.

This follows immediately by considering the Fourier series expansion (sawtooth function) of the floor function for non-integer real numbers x:

$$\lfloor x \rfloor = x - \frac{1}{2} + \frac{1}{\pi} \sum_{k=1}^{\infty} \left[\frac{\sin(2\pi k x)}{k} \right] \tag{6.29}$$

Substituting the value $x = \log_{10}(n)$ to

$$C_{P1\ \min} = 3 + \lfloor \log_{10}(n) \rfloor \tag{6.30}$$

yields:

$$C_{P1\ \min} = \log_{10} n + \frac{5}{2} + \frac{1}{\pi}\left[\sum_{k=1}^{\infty} \frac{\sin\left[2\pi k(\log_{10}(n))\right]}{k}\right] \tag{6.31}$$

and given the bounded variation of the Fourier series, one lower bound is:

$$2 + \log_{10} n \tag{6.32}$$

In fact, it is easy to verify that

$$\frac{5}{2} + \log_{10} n > 3 \tag{6.33}$$

holds for every $n > 3.16228$ (this is one lower bound; what is the infimum?).

As an example, consider $n = 121$. Then, $\log_{10}(121) = 2.082$, so a lower bound of $C_{PI\ min}$ is $2 + 2.082 = 4.082$. Indeed,

$$C_{P1\ \min} = 3 + \left\lfloor \log_{10}(121) \right\rfloor = 5 > 4.082 \tag{6.34}$$

Interestingly, the minimum C_{PI} of binary maps can also be related to the transcendental number π. An alternative expression is derived by expanding the Fourier series by complex numbers. In the previous example, the same result is derived either way:

$$\begin{aligned} C_{P1\,\min} &= \frac{5}{2} + \log_{10}(121) \\ &- \frac{i\left[\log(1 - e^{-2i\pi \log_{10}(121)}) - \log(1 - e^{2i\pi \log_{10}(121)})\right]}{2\pi} = 5 \end{aligned} \tag{6.35}$$

If and only if all black cells of a binary square map appear as a single block of ω-consecutive black cells located after m-consecutive white cells anywhere in the map except for its corner, and

$$\left\lfloor \log_{10}(m) \right\rfloor \wedge \left\lfloor \log_{10}(\omega) \right\rfloor \wedge \left\lfloor \log_{10}(n - m - \omega) \right\rfloor \notin Z,$$

then a lower bound of the minimumC_{PI} of the map is:

$$3 + \log_{10} m\omega + \log_{10}(n - m - \omega) \tag{6.36}$$

This can be proven by considering that

$$C_{P1\,\min} = 6 + \log_{10} m\omega + \log_{10}(n - m - \omega) - \frac{3}{2} +$$
$$+ \frac{1}{\pi}\left[\sum_{k=1}^{\infty} \frac{1}{k}\left(\sin\left[2\pi k(\log_{10} m)\right] + \sin\left[2\pi k(\log_{10} \omega)\right] + \sin\left[2\pi k(\log_{10} n - m - \omega)\right]\right)\right] \tag{6.37}$$

and as

$$-\frac{3}{2} \leq \frac{1}{\pi}\left[\sum_{k=1}^{\infty} \frac{1}{k}\left(\sin\left[2\pi k(\log_{10} m)\right] + \sin\left[2\pi k(\log_{10} \omega)\right] + \sin\left[2\pi k(\log_{10} n - m - \omega)\right]\right)\right] \leq \frac{3}{2} \tag{6.38}$$

it follows that a lower bound is

$$3 + \log_{10} m\omega + \log_{10}(n - m - \omega) \tag{6.39}$$

Finally, as the map size n of a multicolored square map increases with $n \to \infty$, the ratio of the maximum C_{PI} to the number of the cells' boundaries converges to 1 and the minimum C_{PI} tends to zero.

The proof follows easily by counting the number of boundaries from left to right and converting them into a string of symbols. The first row has $\sqrt{n}$ boundaries, but the last cell of the 1st row is the boundary of the second row. Continuing until the penultimate cell, the number of boundaries in a multicolored square map is $U = n - 1$. All the first column cells have no boundary (they share the same boundary with the last column's right boundaries) and the last cell (down right) has no boundary to any other cell. So, for instance, a 4×4 map has 15 boundaries between cells if the entire map is converted into an one-dimensional map. Consequently, it easily follows that

$$\lim_{n\to\infty}\left(\frac{C_{P1\,\max}}{U}\right) = \lim_{n\to\infty} \frac{n}{n-1} = 1 \tag{6.40}$$

and

$$\lim_{n\to\infty}\left(\frac{C_{P1\,\min}}{U}\right) = \lim_{n\to\infty} \frac{3 + \lfloor \log_{10} n \rfloor}{n-1} = 0 \tag{6.41}$$

References

Fortnow, L., Hitchcock, J. M., Pavan, A., Vinodchandran, N. V., & Wang, F. (2011). Extracting Kolmogorov complexity with applications to dimension zero-one laws. *Information and Computation, 209,* 627–636.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission, 1,* 1–17.

Kolmogorov, A. (1986). On the notion of infinite pseudorandom sequences. *Theoretical Computer Science, 39,* 9–33.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady, 10,* 707–710.

Papadimitriou, F. (2002). Modelling indicators and indices of landscape complexity: An approach using GIS. *Ecological Indicators, 2,* 17–25.

Papadimitriou, F. (2009). Modelling spatial landscape complexity using the Levenshtein algorithm. *Ecological Informatics, 4*(1), 51–58.

Papadimitriou, F. (2012). The algorithmic complexity of landscapes. *Landscape Research, 37*(5), 599–611.

Papentin, F. (1973). A Darwinian evolutionary system III. Experiments on the evolution of feeding patterns. *Journal of Theoretical Biology, 39,* 431–445.

Papentin, F. (1980). On order and complexity. I. General considerations. *Journal of Theoretical Biology, 87,* 421–456.

Papentin, F. (1982). On order and complexity. II. Application to chemical and biological structures. *Journal of Theoretical Biology, 95,* 225–245.

Papentin, F. (1983). Binary sequences I. Complexity. *Information Sciences, 31*(1), 1–14.

Papentin, F. (1983a). Binary sequences III. Complexity versus homogeneity and symmetry. *Information Sciences, 31*(1), 33–39.

Seiffert, U., & Michaelis, B. (1997). Growing a 3D-SOMs with 2D-input layer as a classification tool in a motion detection system. *International Journal of Neural Systems, 8*(1), 81–89.