Algorithmic Complexity in Textile Patterns

MSc Thesis *(Afstudeerscriptie)*

written by

Heidi Metzler
(born August 3rd, 1986 in Danbury, United States of America)

under the supervision of **Dr. Leen Torenvliet**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
anticipated July 4th, 2018	Dr. Floris Roelofsen
	Dr. Peter van Emde Boas
	Dr. Jakub Szymanik

Institute for Logic, Language and Computation

Abstract

Algorithmic complexity, also called Kolmogorov complexity and Kolmogorov-Chaitin complexity, motivates the use of techniques to approximate the complexity of objects and measure similarity between them. This thesis explores the application of these methods to patterns in textiles. A brief history of the relevance of textile production to computability is given and it is shown that Turing Machines can be simulated by knitting. Approximations of algorithmic complexity indicate that there may be a way to distinguish meaningful information from arbitrarily populated matrices. The Turkmen tribes were nomadic people with a social structure that allowed woven ornaments to change independently of one another over time. Turkmen textiles present a difficult classification puzzle. The Normalized Compression Distance is a parameter-free, feature-free metric used to measure similarity between objects given approximations of their algorithmic complexity. This technique generates an evolutionary tree consistent with historical information on Turkmen tribes. This demonstrates how algorithmic complexity can be usefully employed in the areas of material culture, archeology, and art history.

Acknowledgments

Special thanks are due to Leen Torenvliet for his encouragement, his good humor and countless insightful comments. At times writing a thesis can make one feel lost, uncertain, and full of doubt. Despite this, I consistently walked out of my meetings with Leen feeling a renewed sense of confidence, enthusiasm and optimism. Thanks are also due to Peter van Emde Boas for reading early drafts, taking the time to meet, and providing useful references. Having comparatively few years' experience in the field, it has been exceptionally handy to have someone with a wealth of knowledge accumulated over decades willing to consider my ideas. Thanks are also extended to Paul Vitányi, Jos Baeten, Hector Zenil, Antonio Rueda-Toicen, Jouke Witteveen, Shahrad Jamshidi and Peter Bloem for their time, input and advice.

Thanks are also due to Dave, Dean, Grzesiek, Jelle, Jonathan, Julia, Krsto, Kyah, Marlou, Max, Mina, Morwenna, Mrinalini, Noor, Rachael, Robert, Robin, Saúl, and Silvan. Also special thanks to Nachiket for sharing my enthusiasm for Winkel 43.

I am grateful to Everett Piper, Chetney Nelsen, Katy Jamshidi and Valentin Vogelmann for their comments on the penultimate draft.

Finally I would like to extend my gratitude and thanks to my partner for emotional support, spectacular cooking, and for moving to Europe to allow me to pursue this opportunity. I feel as though I have an unfair advantage over my fellow students just by having you in my life.

Contents

1 Introduction **3**

2 The Mysteries of the Turkmen and their Textiles **6**
2.1 History and Culture 6
2.2 Controversy 10
2.3 Phylogenetic Analysis 14

3 Knitting and Computability **16**
3.1 Complexity in Textile Patterns 16
3.2 Simulating a Turing Machine by Knitting 17

4 Kolmogorov Complexity **22**
4.1 Compression Algorithms 26
4.2 Coding Theorem 32

5 Sophistication and Enduring Patterns **36**
5.1 Patterns 39
5.2 Pseudo-Random Designs 39
5.3 Algorithmic Complexity 40

6 Measuring Similarity **42**
6.1 Conditional Kolmogorov Complexity and Information Distance 43
6.2 Normalized Information Distance 43
6.3 Normalized Compression Distance 44
6.4 Applications 44
6.5 NCD with Images 46
6.6 Algorithmic Calculus 47

7 Algorithmic Complexity and the Turkmen Textiles **48**
7.1 Linearization of Gols 49
7.2 Implementation 51
7.3 Results 52
7.4 Discussion 53

8 Conclusion **56**

Bibliography **56**

1 | Introduction

What makes designs appealing? Given an $n \times n$ grid in which each cell can take on one of two colors, there are $2^{n \times n}$ different possible ways of coloring in the grid. When observing old patterns in textiles, wood carvings, pottery and stone reliefs we do not sense a uniform distribution over all possible patterns: we see a small subset, usually consisting of pictorial representations of things encountered in real life, such as plants and animals, or abstract designs. The latter sort is the focus of this thesis.

Figure 1.1: Left: Detail from a Berber cloak, Atlas Mountains in Morocco, British Museum. Right: Detail from Poqomam *sut* (cloth used for both ceremonial and daily activities), Palín, Guatemala, Tropenmuseum.

Around the world one finds patterns that are distinct but share some features. Compare the pictures in Figure 1.1, one from Morocco and the other from Guatemala.

Although the patterns are different, similar features are present. Both patterns contain triangles, symmetry, and alternating color work. Both pieces were also woven. When designing and producing patterns certain features like symmetry and repetition are preferred and evidently have been for a long time. They are also easily expressed by short programs, making them compressible.

Clearly, there is a degree of similarity in the patterns cultures prefer to reproduce. One example is that of meandering patterns. Liu and Toussaint [38] examined Roman mosaic patterns found in a manor in England and pointed out similarities between those and patterns found elsewhere in the world. These patterns can all be generated by similar algorithms. One such algorithm involves meandering around evenly spaced points. This will be discussed in Chapter 3 and an example will be given.

Why is it that people produce these patterns specifically? Perhaps there are shapes that the eye likes to see and the mind likes to create, like the ear enjoying syncopation in music. What might the qualities of these patterns be? Is there a consistent degree of complexity to them? Researchers such as Koppel and Atlan [35], Friedenberg and Liby [23], Gauvrit, Soler-Toscano, and Zenil [24], and Gauvrit, Soler-Toscano, and Guida [25] have asked similar questions before. Perhaps there is a range of complexity that is appealing, a sweet spot between blank space and white noise-type chaos.

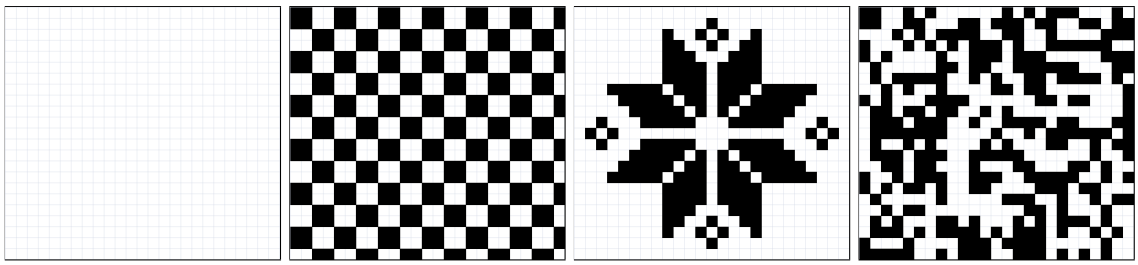


Figure 1.2: Simple to complex patterns. The far right pattern was generated using Excel's built-in pseudo-random number generator.

In order to examine complexity in textiles in a more precise way, there are some concepts with which one needs to be familiar. First the history of the Turkmen tribes and their textiles will be discussed. Then knitting will be used as an example of how textile production involves language and computability. Information about algorithmic complexity, its approximations, and arguments for and against two methods of approximating an object's complexity will be given. Next, we will investigate the matter of whether or not there might an appealing range of complexity using designs from old textiles. Following this, a metric called the Normalized Compression

Distance will be presented along with some of its applications. A recently proposed alternative will also be introduced. Finally the methodology and results of an alternative analysis of Turkmen genealogy based on algorithmic complexity will be given and discussed in the greater context of our understanding of Turkmen history thus far.

2 | The Mysteries of the Turkmen and their Textiles

In carpet ornaments, in combinations of colors, that have come to us over the threshold of centuries and millennia, there resound different melodic echoes of artistic creativity of the past that have stood firm against the pressure of inexorable all-destroying time. -N. Burdukov, 1904

2.1 History and Culture

Oghuz, the mythical great grandson of Japhet, to whom Noah had given the East, had six sons. Each of them in turn had four sons. Each of these 24 descendants of Oghuz was given a brand sign, the totem of a bird of prey, and a cut of meat to receive at feasts [3]. The sons were the heads of their respective tribes, later known as the Turkmen tribes. The tribes respected their totem birds and would not kill or hunt them. The cut of meat a tribe was assigned was important: the best cuts of meat were awarded to the highest ranked tribes for feasts and a social ordering was implied. Around the early 10^{th} century, the Turkic tribes spread out from the Aral Steppe (today the Kazakh Steppe) and distributed themselves over modern-day Uzbekistan, Turkmenistan, Iran, Afghanistan, Iraq, Syria, Turkey, and reached as far west as the Balkans and as far south as India. The Uighurs of Tarim Basin, in western China, and the Cossacks of Russia share these origins along with many other groups.

Turkmen were pastoral nomads, meaning they inhabited different areas depending on the time of year. Communities within the tribe would be split: some would look after livestock and others would grow crops. Some tribesmen stayed put in areas for extended periods of time. Those tribesmen that grew crops would plant in early spring, head north to escape the heat and dangerous pests for the hottest months of the year, and return to harvest their crops at the end of the summer.

Tribesmen looking after livestock would travel to find water and grazing land for their sheep, goats, camels, and other animals. Tribesmen lived in hardy tents called yurts that would be rolled up for travel, although more settled people might have had permanent sheds and fences surrounding their yurts for keeping animals. They did not generally have dressers or cupboards and instead stored belongings in bags hanging from lattice-like structures on the walls of the yurt [46, 3, 54, 43].

The most important written historical documents attesting to the movements of the tribes and their genealogy are from the 11^{th}, 14^{th}, and 17^{th} centuries [44]. Mahmud Kashgari recorded the names and brand signs of 22 tribes in Arabic during the 11^{th} century. Two more tribes had already separated and formed the Khalaj people, some of whom traveled to what is today India. More information about the Turkmen comes from the record of Rashid al-Din in the 14^{th} century. He recorded the names of 24 tribes, their brand signs, totem birds, and cuts of meat. When comparing the names and brand signs from Mahmud Kashgari's records and those of Rashid al-Din, some similarities are observable but time shows itself through the many changes in the list. A more extensive account is given by Abul Ghazi Bahadur, the Khan of Khiva (a city that lies today in Uzbekistan). He declared war on the Turkmen and killed many. In the years immediately preceding his death he wrote a genealogy of the Turkic peoples using Rashid al-Din's account and other written sources available at the time, and drawing from Turkic oral traditions, see Tehrani and Collard [54], Azadi [3]. From 1525 to 1535, the Salor, Saryk, Yomut, Ersari, and Tekke tribes lived in the Khorassan area. One scholar reported a relationship between the greater Salor tribe and a group of other tribes, making a Salor/Outer Salor distinction. More probably it was the case that the places the other tribes inhabited were referenced with respect to where the larger Salor tribe was because the Salor tribe was so important at the time according to Azadi [3]. The Tekke and Saryk tribes were recorded as being related to the Salor Toi-Tumas tribe. The Ersari tribe was separate from these but almost as large as the Salor. This is consistent with records of tribute: one fortieth of each tribe's sheep were paid and records from the 16^{th} and 17^{th} centuries reflect payments of 16,000 sheep each from the Salor and Ersari, but the Tekke, Saryk, and Yomut only gave 8,000 each.

The tribes were in a constant state of warfare and conducted raids on sedentary communities to obtain livestock and other goods, sometimes people. Not only did the tribes move with the seasons, they travelled far in search of pasture, water, and other resources. Multiple tribes might occupy a particular stretch of territory within a few years [3]. Because of this little reliable historical information is available about the movements of the tribes and few traces are left. The Turkmen remained chiefly nomadic until their suppression by Russia in the late 19^{th} century. After this, tribes

became mostly sedentary.

Weaving was a well-established tradition among Turkmen and it took years to become a decent weaver. The rugs covering the floors of the yurts, bags storing food and utensils, and decorative items were all woven on portable looms. The sheep were sheared in spring and autumn, supplying large quantities of wool. Different types of wool from different seasons were required for the warp, weft, and pile. Each type of fiber came from different parts of animals of different ages. Dyes were made from plants and insects gathered from the Turkmen's surroundings, see Spooner et al. [52] for more information. After settling, some tribes also began to use synthetic dyes in their wool. The wool would be hand spun with portable spindles. Especially elaborate decorations would be constructed for wedding processions, when a bride would be brought to her new home on a camel. The textiles were hand knotted and tribes tied knots in symmetric or asymmetric ways as discussed in Tehrani and Collard [54]. Density could be upwards of 3,000 knots per square decimeter. The skill of the weaver determined how many knots could be tied in a day. According to William Irons, an anthropologist, "one woman could weave roughly one square foot in a day of heavy weaving, about twelve hours at the loom." Rugs could sometimes take three or four women months to complete according to Spooner et al. [52].

Weaving was strictly done by women. Being a skilled weaver gave the craftswoman standing in her tribe [52]. Skills and designs were passed from mother to daughter and there was a great deal of time to weave, especially in winter. Women did not have contact with women from other tribes and even marriage took place within a single tribe. Endogamy is estimated to account for 90% of marriages in one particular tribe observed by Irons [32]. A genetic study indicated that the genes of people from separate tribes differed significantly from one another, corroborating this estimation (see Turaeva et al. [58]).

Because patterns were not passed on in written form or with diagrams, any small mutations that occurred in a pattern would be independent of changes that occurred in patterns elsewhere. This gave rise to a unique set of designs reflected in rugs, bags, and decorative ornaments. When entire tribes split, one part setting off in another direction, the women carried the patterns with them. Because Turkic people share a common origin, this process occurred repeatedly over a long period of time. One of the initial recorded occurrences was the splitting off of two tribes recorded by Mahmud Kashgari in the 11^{th} century when two tribes left, forming the Khalaj people.

Figure 2.1 shows a map of Turkic languages and the approximate locations of the communities that speak them, illustrating the eventual widespread distribution of the Turkmen tribes, courtesy of the World Atlas of Language Structures from Dryer

Figure 2.1: A map showing areas where Turkic languages are spoken today, see Dryer and Haspelmath [20].

and Haspelmath [20]. Language is used as a proxy for phylogenesis by Guglielmino, Viganotti, Hewlett, and Cavalli-Sforza [27], an analysis of cultural traits of African societies. This gives credence to the suggestion that culture follows language in this case.

Turkmen women use a variety of beautiful geometric designs. Some of these are believed to have originated from the totem birds of the original tribes. Other ornaments clearly depict horses (see Azadi [3]). One octagonal design called a gol is hypothesized by some to have a special tribal significance as an emblem. The presence or absence of such ornaments is used to help classify textiles and determine their origin. Other features used to do this are how the knots are tied (symmetrically or asymmetrically), the type of dye used, and other structural features as shown in Tehrani and Collard [54].

Turkic peoples were self-reliant before the Russian suppression that took place during the late 18^{th} century. After becoming sedentary, tribes joined markets and began participating more strongly in established economies. European and American collectors expressed a preference for Salor designs [54, 43, 22] and the purpose of rug weaving likely shifted towards economic production from domestic production.

2.2 Controversy

Mystery and disagreement surrounds the ornaments, their origins, and how they are used. It could be the case that all symbols on the carpets are original, but this would be strange - a group of people that happen to be genealogically related yet manage to independently conceive of very similar ornaments, many of which are octagonal. This seems far-fetched. It is more likely that the same ornaments were used by the original tribes and that these ornaments travelled with the women who wove them, creating a line of artifacts witnessing matrilineal descent in a patrilineal society.

The origins of the gol are disputed. Below are some of the main theories from the literature.

- Moshkova [43] theorizes that there are living gols and dead gols and draws a distinction between a gul and a gol, the former meaning flower and the latter signifying a tribal ornament. Living gols are those actively used as tribal symbols and occur as main motifs on the grandest rugs. Dead gols would be those that belonged to tribes that are no longer established and, rather than occurring as a main motif of a large rug, are relegated to smaller, less conspicuous pieces like comb carriers or bags. See also Azadi [3].

- Mackie and Thompson [39] argued that one type of ornament on a particular rug provided a missing link between tribal gols and palmettes in a Persian design.

- Eiland Jr [22] put forth the idea that the gol has no tribal significance whatsoever and is descended from patterns in silks traded between China and the West since approximately 300 AD.

- Baker [4] points out that the possibility of the design originating in a court and filtering down to the tribes through urban carpet production is not explored.

Just because the above viewpoints are juxtaposed does not mean they are equally likely. Like a well-read but poorly spoken academic publicly debating a charismatic but ill-informed layman, placing these theories side by side gives too much weight to the less likely among them.

Moshkova [43] claims, on one hand, that the women of a conquered tribe must give up their own gol, begin producing the gols of the dominant tribe, and destroy textiles containing the old gol. On the other hand, she also says that tribes conquering other tribes might actually adopt the gol of a dominated tribe in some circumstances.

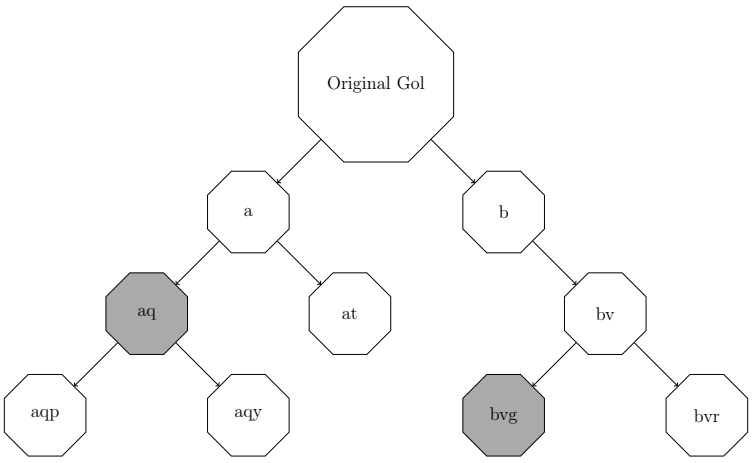


Figure 2.2: Possible evolution of tribal gols from an original. We may only have access to a few nodes on the tree, such as **aq** and **bvg**. The tree is not necessarily binary; one gol might have more than two descendants depending on the splitting of the tribes.

Rugs, a long-traded commodity, would likely be too valuable for a conquering tribe to destroy. Furthermore the two claims concerning a conquering tribe's gol seem inconsistent. There is no observational or other evidence given in her paper to support these broad claims, and although it is a translation surely no conscientious translator would omit such crucial information.

The comment of Baker [4] that the ornaments may have originated with a court and then spread through urban production does not appear to be supported by any facts and seems to be a critique of the eagerness with which rug scholars and enthusiasts, so-called 'Turkomaniacs', jump to broad conclusions with little evidence. She references a late 20^{th} century anthropological field work account when claiming tribes borrowed one another's patterns. During the time period in question (10^{th} century to mid-19^{th} century), tribes were often at war with one another and women did not marry into other tribes. Because the tribes did have contact with sedentary societies, this hypothesis cannot be ruled out.

Similar ornaments appear on textiles over a wide geographic region. Remaining agnostic as to the origin of these symbols, let us simply suppose that there is one that Turkic peoples came into contact with early on. Assuming the women of the Turkmen tribes began long ago with some sort of octagonal ornament, mutations would have occurred and resulted in different ornaments of the same general shape. The process would resemble a tree that branches at points in time when tribes or

parts of tribes separated. As discussed above, this occurrence in Turkmen tribes is well-documented. Over time a sequence of designs has emerged. Each octagonal ornament is a node on a larger tree where the original ornament sits at the root, see Figure 2.2. Through generations of tribes separating and moving around, unique ornaments were born and changed again, yielding a set of diversely ornamented textiles. It is not safe to assume that we have all of the nodes on the tree. In fact, we cannot be sure how large or diverse the tree has become. What we do have is a small set of nodes, the ornaments used on the rugs, *chuvals*, and other textiles that are in museums or collections shown publicly represented in Figure 2.2 by shaded nodes.

Each distinct ornament is hypothesized to correspond to a tribe: it seems highly unlikely that women of distinct tribes, separated by both geography and social segregation, would independently develop the same ornaments. Ornaments should follow matrilineal descent, and because there was not intermarriage between the tribes the ornaments follow the tribes.

But which ornaments correspond to which tribes? Due to the lack of historical records documenting the movements of tribes and the absence of dwellings or other remnants of tribal inhabitation we do not know which tribes went where when (and probably never will).

The ornaments of some tribes are discussed in the literature, for example Thompson's identification of the 'S-group' and subsequent identification of the textiles as products of the Salor tribe. The gols of the Tekke, Ersari, and Yomut are also established in the literature.

In information theory, one considers problems involving signals transmitted over noisy channels. The repeated transmission of ornaments from mother to daughter over generations is analogous but rather than a noisy channel mutations are introduced through some combination of cognitive bias, subjective preference, and the general propensity of people to doodle.

Rug attribution is complicated by these and other reasons. It is not uncommon for experts to disagree about the origins of a carpet. For example, Moshkova and Eiland disagree about the provenance of one particular piece shown in Figure 2.3: Moshkova attributes the gol to the Yomut but Eiland believes it to be from the Tekke, see the remarks in the translation of Moshkova [43].

There are issues further complicating accurate rug attribution. After their suppression by Russia, the weaving practices of tribes likely changed. Now liable for state taxes the tribes were required to participate more strongly in the established economy. Westerners preferred Salor designs, giving an economic incentive for tribeswomen to use certain ornaments over others. If tribes are identified on the

Figure 2.3: A *chuval* (storage bag) identified by Moshkova as Yomut and Eiland as Tekke.

basis of the ornaments they use, this muddies the water for anyone trying to infer a textile's origin. A rug woven by a rare or obscure tribe makes it more desired by collectors, inflating the price. As a result the consumer side of the economy also complicates the accuracy of the process by incentivizing inventive rug attribution. As discussed by Baker [4] and David and Saunders [18], Westerners romanticize the nomadic lives of Turkmen and value authenticity, a nebulous concept that makes pieces woven in yurts by nomads worth more than finely crafted urban textiles. The more 'tribal-looking' the piece, the more it might be worth.

Finally, the Turkic language has been written down using different alphabets: first in modified Arabic, then in Russian Cyrillic, and then the Latin alphabet [46]. Multiple transliterations of the same word generate confusion: Moshkova's distinction between 'gul' and 'gol' is seen as ridiculous by some other scholars and the literature is rife with multiple spellings of the same tribes' names.

As a result, Turkmen textiles present quite a classification puzzle.

2.3 Phylogenetic Analysis

Creating evolutionary trees from information about data is common in biology, but there is disagreement about the use of biological phylogenetic analysis as a tool of inference in the field of material culture. These techniques perform best when traditional transmission is strong and cultural exchange is weaker according to Tëmkin and Eldredge [55], allowing each culture to distinguish itself in a marked way. As discussed above, endogamy is estimated to account for over 90% of marriages in a well-studied Turkmen tribe, the Yomut [33]: women from other tribes sometimes married into the Yomut tribe, but Yomut women did not marry outside their tribe. There is reason to believe that this was the case for other tribes as well: in a genetic study Turaeva, Ginter, Revazov, Garkavtseva, and Sotnikova [58] found that the main tribes were highly isolated and the differences between gene frequencies in tribes were significant. Given that weaving was an exclusively female activity, the patterns of each tribe would not have been widely dispersed. Also, patterns and technologies were passed on from mother to daughter without written instructions or diagrams. This allowed each tribe to distinguish itself without interference from others.

The nomadic lifestyle and common ancestry of the tribes as well as their social traditions make their woven patterns the ideal test case for phylogenetic analysis. Phylogenetic analysis in textiles has been conducted with respect to specific features of textiles, including physical features like type of dye and orientation of knots, and design features like what characteristics the gol has [54].

Cultures evolve over time. But what is the nature of this evolution? Given two cultures A and B, is a wholly new culture C formed over time through some mechanism, or do cultures A and B morph into C through contact with one another? Ethnogenesis occurs when cultural evolution takes place through the exchange of ideas and practice. Phylogenesis occurs when cultural evolution occurs as the population divides. Tehrani and Collard [54] empirically investigated whether ethnogenesis or phylogenesis was taking place through a case study. Ethnogenesis is usually represented by a reticulated graph resembling a lattice. Phylogenesis, on the other hand, is represented by 'family trees', also known as dendrograms. Because a bifurcating tree model is simpler than a lattice, it was used as the null model in this paper to assess whether or not phylogenesis or ethnogenesis was occurring in line with the principle of parsimony.

60 woven artifacts from museums in the United Kingdom, Russia, Germany, and the United States were used in the study. The analysis of the textiles took into account structural features, like whether knots are tied symmetrically or asymmet-

rically, as well as whether the wool was dyed with natural or synthetic dyes. The presence or absence of decorative ornaments was also noted. A permutation tail probability test determined that there was a phylogenetic signal by randomly permuting a subset of the data and comparing the length of the most parsimonious tree found in the subset to the most parsimonious tree found for the unpermuted data. A second analysis assessed how well the tree fit the data by examining how many additional assumptions were required to construct the tree. The results suggested that the Ersari and Saryk tribes are more closely related to one another than to other tribes and that the Ersari, Salor, and Saryk are more closely related to one another than to the Tekke, see Tehrani and Collard [54] for more specific information. These results were interpreted as suggesting that phylogenesis was the dominant cultural evolutionary process. After the Turkmen's defeat by Tsarist Russia, ethnogenesis contributed to change in Tekke weavings: they began to have Salor designs. As noted above Salor textiles were the most-sought after Turkmen textiles in the West, so there may have been strong economic reasons for Tekke weavers to emulate Salor designs.

The work of Tehrani and Collard [54] demonstrated how phylogenetic analysis could be conducted using these textiles, but the features used were not design-specific and included structural features. Later on, we will build our own evolutionary tree of Turkmen tribes using different features.

3 | Knitting and Computability

Computing has some of its origins in textile production. Joseph-Marie Jacquard invented a loom that relied on punched cards to lift specific harnesses, each of which held threads. The lifting of these threads allowed a shuttle to pass through a specific path and over iterations of the process create complex designs. The loom served as inspiration for Charles Babbage's Analytical Engine, the theoretical precursor of the modern computer, and Ada Byron is regarded as the first programmer due to an algorithm written for the Analytical Engine to use to compute the Bernoulli numbers in 1843. More details can be found in Park and Jayaraman [45] and Toole [57].

Belcastro [7] showed how knitting Klein bottles and other mathematical objects can yield a better understanding of topological spaces and offered a proof of how every topological space can be knit in Belcastro [6]. Patterns used to create textiles using different methods, like knitting, crocheting, and tatting can help us to further our understanding and sharpen our intuitions about what language can express in terms of computability.

3.1 Complexity in Textile Patterns

The complexity of producing an object can vary across textile production methods. For example, it might be trivially easy to produce the Sierpinski Triangle in tatting and there may be a straightforward way to translate this pattern into one for crocheting. There is also a schema for translating crochet patterns into knitting patterns (and back) but we cannot assume this will be as clear because of the fundamental differences between tatting and crocheting, on one hand, and knitting on the other.

This leads us to another question: when are two patterns the same? Suppose we define a *stitchomorphism* to be a one-to-one function f of technique $\mathfrak{T}$ to $\mathfrak{T}'$ that preserves the stitches, compound stitches, and turns. In theory, the rote translation of one technique's pattern to another is possible, but in practice it could be the case that adding an extra stitch, working one extra row, or otherwise permuting

the pattern will yield a more accurate rendering of the shape being created than a stitchomorphism.

3.2 Simulating a Turing Machine by Knitting

Each type of textile has its own language. The language of knitting is a finite alphabet specified in each pattern, where some stitches are composed of other stitches. Two fundamentally different atomic stitches are knit stitches, usually k in patterns, purl stitches, represented p. There are also yarn overs, yo and others. In knitting, complex patterns can easily be produced by a short program, see Figure 3.1. Many knitting patterns signify that a sequence is to be repeated by placing it between asterisks.

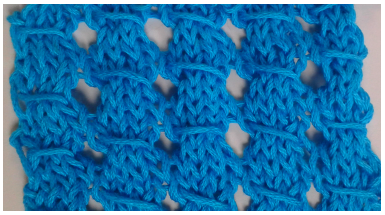


Figure 3.1: Program: Row 1: **k**, Row 2:**p4, yo**, Row 3: **drop yo, yo, sl1, k3, psso**, Row 4: **p**. From Alexis Layton, Ravelry, *Waffle Blanket*.

Analogies were given in a Computational Model of Knitting , see [1], and Howard [31] pointed out that the elementary cellular automaton Rule 110 is Turing complete (as proven by Cook [17]) and can be knit, thus knitting can be seen as Turing complete. To take this a step further, we will discuss the simulation of a Turing machine by knitting.

Following Hopcroft et al. [30], a Turing Machine is composed of a finite control, a tape divided into cells, a finite input written on the tape in the finite input alphabet, and those spaces that surround the finite input contain blanks. The tape extends as far as necessary in either direction. A tape head is scanning one of the tape cells. The tape head starts by scanning the leftmost input cell on the tape, then moves, possibly changing states, writing a tape symbol on the cell scanned replacing the current symbol, and moving one cell to the left or right.

In order to demonstrate that knitting can simulate a Turing Machine we define the counterparts of the components of a Turing Machine for a Stitch Machine. Two needles extend arbitrarily far in either direction with cast on, knit, or purled stitches. A knit stitch is one where the yarn is brought forwards through the previous stitch, creating a v-shape. A purl stitch is one where the yarn is brought backwards through

Figure 3.2: The knit head of a simulated Turing machine. The first stitch on the left needle is the one being scanned. All stitches on the right needle are cast on, equivalent to the blanks on a Turing tape. The input stitches are placed on the left needle and the knit head begins at the rightmost stitch of the input.

the previous stitch, creating a small bump. A yarn over loops the yarn over the needle.

The points of these two needles serve as our tape head, called a knit head, see Figure 3.2. The stitches are the counterpart of the tape. The input stitches are a finite number of knit or purled stitches and cast on stitches surround the input stitches (the counterparts of blanks on Turing tape) and extend arbitrarily far in either direction. We might run out of yarn with which to cast on stitches on the left side or the right side. If this happens, the human operator must locate another skein and cast on more stitches. Pieces of wire of the proper gauge can be used to extend the knitting needles if they are not long enough. The pieces of wire can be used as storage for the tape and wound around spools if necessary. Let us assume that there is as much yarn and needlespace as needed.

The stitches located closest to the points of the needles are to the immediate left and right of the knit head and the further away the stitches are from the points of the needles the further they are from the knit head. The knit head scans the stitch immediately to the left of the gap between the stitches, the first stitch on the left needle. The knit head starts by scanning the rightmost input stitch then moves, possibly changing states, and knitting, purling, or yarning over the stitch, ensuring it is the first stitch occurring on the left needle (slipping it back from the right after knitting or purling it, if necessary) before moving one stitch to the left or right. The knit head moves one stitch to the right by slipping the first stitch on the right needle to the left needle and moves one stitch to the left by slipping the first stitch on the

left needle to the right needle. Now that we have described the basic operations we move on to the translation of the description of a Turing Machine.

A Turing Machine is described by a $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, where Q is a finite set of states of the finite control, Σ is the finite set of input symbols, Γ is the set of complete tape symbols, δ is the transition function, q_0 is the start state, B is the blank, and F is the set of accepting states where the machine halts. Σ is a subset of Γ and F is a subset of Q. δ has arguments q and X, where q is a state and X a tape symbol. $\delta(q, X)$, if defined, is a triple (p, Y, D_M) where p is the next state, Y is the tape symbol to be written, and D_M is the direction the tape head moves on the tape, either left (L) or right (R). a tuple $S = (R, \Theta, \Lambda, \pi, r_0, C, A)$ to which we can map M.

We map each tuple M to $S = (R, \Theta, \Lambda, \pi, r_0, YO, A)$, where R is a set of states specifying the transition function π for each stitch in Λ, Θ is finite set of input stitches, Λ is the finite set of complete stitch symbols, π has arguments r and Z where r is a state and Z is an input stitch and $\pi(r, Z)$, if defined, is a triple (t, W, D_S) where t is the next state, W is the stitch to be written, and D_S is the direction to move in, left (L) or right (R).

By convention let $\{0, 1, B\}$ correspond to $\{K, P, YO\}$ where K is knit, P is purl, and YO is yarn over. When yarning over a stitch, the stitch being scanned is pulled off of the end of the left needle and a loop is put in its place.
Q to R: For each element of Q create a corresponding unique element of R as $q_i = r_i$.
Σ to Θ: The input symbols $\{0, 1\}$ are mapped to $\{K, P\}$ where 0=K and 1=P.
Γ to Λ: The tape symbols $\{0, 1, B\}$ are mapped to $\{K, P, YO\}$ where $B = YO$.
δ to π: Define $\delta(q, X) = (p, Y, D_M)$ as $\pi(r, Z) = (t, W, D_S)$ where D_S is the inverse of D_M (L goes to R and L goes to R), W is the stitch to be made (defined by Γ to Λ above), and t is the next state (defined by Q to R above).
q_0 to r_0: $q_0 = r_0$, see Q to R above.
B to YO: $B = YO$ as defined by Γ to Λ above.
F to A: For each q_i in F, let A contain the corresponding r_i.

To illustrate this simulation, consider the proper subtraction function from Hopcroft et al. [30]. Proper subtraction does not give negative numbers as output or accept them as input, so if two numbers are equal or the larger of the two numbers is being subtracted, the result will be 0 and otherwise it will be the natural number-valued difference between two non-negative integers. Let the machine M_{PS} take as input two numbers, x and y, and let $x - y$ be the number we want to compute. M_{PS} takes as input a tape with x 0's and y 10's. If we wanted to compute 4-2, for example, the input tape would read 00001010 on M_{PS}. Table 3.1 shows the proper subtraction function for M_{PS} and Table 3.2 shows the translation of the function to a Stitch

Machine for S_{PS}. Once the computation is finished, the result of the function is printed on the tape of M_{PS}: if $x > y$, then $x - y$ 0's are on the tape. If $x \leq y$, then the tape is completely blank. For S_{PS}, if $x > y$, then $x - y$ K stitches are on the needle. If $x \leq y$, then the needles only contain YO stitches. See Figure 3.3 and Figure 3.4 for examples of r_0 and r_6 for S_{PS}, respectively.

Table 3.1: Turing Machine M_{PS}

State	*Symbol*		
	0	1	B
q_0	(q_1,B,R)	(q_5,B,R)	-
q_1	(q_1,0,R)	(q_2,1,R)	-
q_2	(q_3,1,L)	(q_2,1,R)	(q_4,B,L)
q_3	(q_3,O,L)	(q_3,1,L)	(q_0,B,R)
q_4	(q_4,O,L)	(q_4,B,L)	(q_6,0,R)
q_5	(q_5,B,R)	(q_5,B,R)	(q_6,B,R)
q_6	-	-	-

Table 3.2: Stitch Machine S_{PS}

State	*Stitch*		
	K	P	YO
r_0	(r_1,YO,R)	(r_5,YO,R)	-
r_1	(r_1,K,R)	(r_2,P,R)	-
r_2	(r_3,P,L)	(r_2,P,R)	(r_4,YO,L)
r_3	(r_3,K,L)	(r_3,P,L)	(r_0,YO,R)
r_4	(r_4,K,L)	(r_4,YO,L)	(r_6,K,R)
r_5	(r_5,YO,R)	(r_5,YO,R)	(r_6,YO,R)
r_6	-	-	-

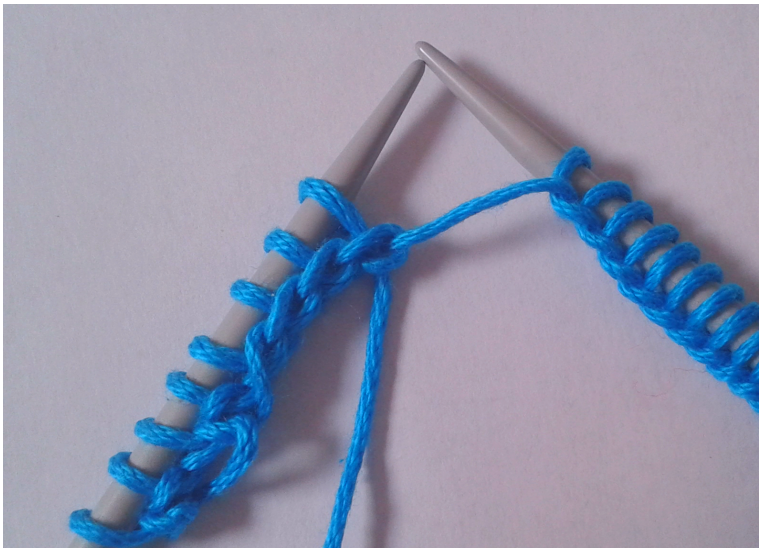


Figure 3.3: Tape for S_{PS}. The knitter can read the input stitches $KKKKPKPK$ from right to left on the left needle, allowing the knitter to compute 4-2.

Figure 3.4: The output of the computation of 4-2 by S_{PS}. The two K stitches are on the right needle. The output need not be aesthetically pleasing.

4 | Kolmogorov Complexity

Imagine hearing a carpenter hammering in the distance. Generally one notes a rhythm, an evenly spaced set of strikes, perhaps with occasional pauses in between. It is difficult to imagine a person doling out uneven strikes with a hammer, like a child playing Whack-A-Mole.

Regular sequences are easily produced by short programs. While hearing hammering in the distance, one could adjust a metronome to produce ticks in concert with the strikes of the hammer. If the strikes of the hammer were irregular it would not be possible to do so. This may have to do with the human faculty responsible for beat induction, be it a domain-specific or domain-general one (see Honing [29] for more information). The human ability to compress information is what allows us to learn.

A string's Kolmogorov complexity $K(x)$[1] is the length of the shortest program that produces it. If a string is highly regular, as in the case of construction hammering, we can write a short program to produce it. If a string has little regularity, like the hammer strikes generated by a child playing Whack-A-Mole, then a longer program will be needed in order to produce it. Consider the following examples of strings with low Kolmogorov complexity.

0101010101010101010101010101010101010101010101010101

Rather than reproducing the whole string bit by bit, we could leverage the redundancy in the string to write a short program that says `print 01 24 times`, or for the following sequence, `for` n `in` $\mathbb{N}$`, print 1, print 0` n `times`.

10100100010000100000100000010000000100000000...

[1]Kolmogorov complexity is often written $C(x)$ in the literature following Li and Vitányi [37] and $K(x)$ is used for prefix-free Kolmogorov complexity, but in this paper we reserve $C(x)$ to refer to *the compressed length of x*.

The problem of assigning probabilities in accordance with the principle of parsimony has intrigued researchers for a long time. A helpful summary of developments in scientific thought is given in Kirchherr, Li, and Vitányi [34]. Epicurus, a Greek philosopher who was alive around 300 B.C., stated 'There are some phenomena to which it is not enough to assign one cause. We must enumerate several, though in fact there is only one.' Occam's Razor instructs us to choose the 'simplest' hypothesis from among many, though the word simple can be difficult to pin down. Bayes' Rule allows us to derive the likelihood of a hypothesis being correct given its prior probability and its probability given some observed data. But how do we assign prior probabilities to hypotheses in the first place?

In 1960 R. J. Solomonoff published a paper on inductive inference, see [51]. The basic idea is to assign probabilities to strings based on the length of the shortest programs that produce them (with no input). The same ideas were developed independently by A.N. Kolmogorov and then G.J. Chaitin a few years later, see Li and Vitányi [37], the classical textbook, for detailed information on the original papers and the development of the field. Algorithmic complexity is also called Kolmogorov complexity and Kolmogorov-Chaitin complexity. The more complex a string is, the longer the shortest program that produces it will be. The strings with the highest Kolmogorov complexity are called Kolmogorov random strings. There is no shorter way to produce these strings than to simply list each bit of them. As a result the shortest program that produces one of these strings is the length of the string itself plus some constant, like the number 5 to account for the inclusion of the characters in the word `print`. Because there is no shorter way to produce these strings than to list each bit of them individually, they are incompressible.

There are several proofs of the existence of incompressible strings (see Li and Vitányi [37]). One of these goes as follows:

> There are 2^n binary strings of length n. There are $2^n - 1$ binary programs shorter than n. Each program only produces at most one string. Therefore at least one string is incompressible.

Some strings can be produced with a shorter program in one language than another. The Invariance Theorem states that given any descriptive or programming language and a program that produces a string, the language that can produce the string with the shortest program is only off by some constant. The constant does not even depend on the string, only on the programming language. The proof shows that a translation from one language to another can be expressed by some constant number of characters (Li and Vitányi [37]). This gives a somewhat unsatisfying upper bound.

The notion of Kolmogorov complexity is useful for proving theorems using the Incompressibility Method. The proofs involve selecting a random object, noting that the object is incompressible, showing that the object has some property, and then showing that if it does not have the property then the object is compressible, contradicting the main premise of the proof. Kolmogorov complexity is uncomputable. One proof of this claim involves a reduction of the Halting problem to computing Kolmogorov complexity. There are many other applications of Kolmogorov complexity, for more information see Li and Vitányi [37].

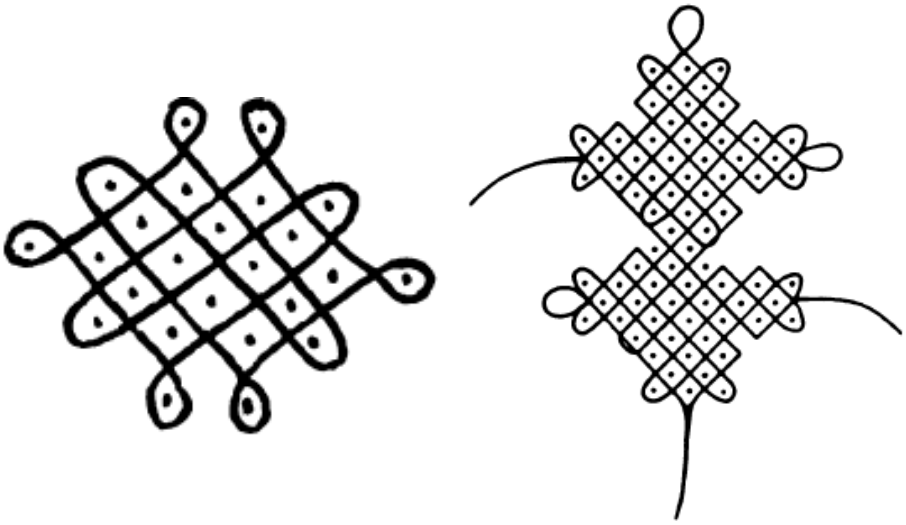


Figure 4.1: Sketches of Tchkokwe *sona* drawings from Angola (left) and Tamil *kolam* drawings from southern India (right). [26, 38]

Kolmogorov complexity has applications in the realm of two- and three-dimensional patterns as well. For a fascinating take on origami-folding algorithms, see Duong [21]. Liu and Toussaint [38] examined Roman mosaic patterns found in England and pointed out similarities between those and patterns found elsewhere in the world, like the Tchkokwe *sona* drawings from Angola and *kolam* art from Tamil, in southern India, shown in Figure 4.1. These patterns are constructed by similar algorithms where lines are drawn between regularly spaced points. An algorithm that might be used to construct a meander pattern is described by Gerdes [26] and given in Liu and Toussaint [38]:

> 'First construct a square that encloses all the dots such that the distance between the square and the dots is half the distance between two horizontally

> adjacent dots. Now imagine that this square is either a billiard table, or made up of mirrors. To construct the...curve start a billiard ball (or beam of light in case of mirrors) rolling from a point directly on the square and above the upper leftmost dot at an angle of 45 degrees. Then just follow the path of the ball until it returns to the starting point, remembering that (like light) whenever the ball hits an edge of the square it bounces (or reflects) at an angle of 45 degrees, thus turning by an angle of 90 degrees. To trace out the remaining curves, repeat this procedure starting on all the points directly above the dots contained in the top row of dots.'

The resulting pattern can be transformed easily into the patterns found in Roman mosaics if one changes the diagonal lines into a combination of vertical and horizontal ones, see Figure 4.2, and makes twists at the intersections. It may be the case that people producing these drawings in various parts of the world are not thinking of billiard tables, mirrors, or beams of light when constructing them, but the algorithms are variations of one another and the ideas are the same.

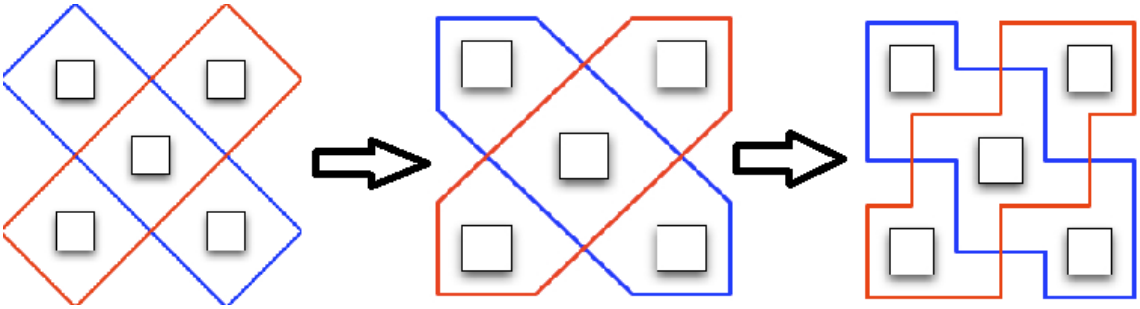


Figure 4.2: How to transform a meander pattern into a design found in Roman mosaics, as demonstrated by Liu and Toussaint [38].

The algorithms that produce these meander patterns are similar, thus we would like a notion of complexity that captures this. Kolmogorov complexity is precisely such a notion. Two-dimensional patterns can also be captured in a one-dimensional string by linearizing them, but the pattern may be less apparent.

Because it is uncomputable, in practice algorithmic complexity is approximated using different methods. One of these methods uses lossless data compression algorithms. Another involves the estimation of Kolmogorov complexity by frequencies. In the following subsections, I will explain both methods, some of their applications, and objections to their use.

4.1 Compression Algorithms

Data compression has its roots in communications: given a finite alphabet and some messages that need to be transmitted, what is the most efficient way to encode the messages? Morse and others developed a code for telegraph systems that used the frequencies of letters to determine their codeword lengths. Later discoveries by Shannon, Fano, Huffman, and Kraft advanced information theory. The advent of computers necessitated the solving of some related problems. One of these was how to store information. Due to limited storage availability, other ways of making data smaller were needed. The general purpose of compression algorithms is to take some quantity of data as input and give a smaller program that reproduces the data as output. Lossy compression algorithms sacrifice some information so that the rest of the data can be efficiently encoded. Lossless compression algorithms ensure the integrity of the data while removing redundancies and conserving some amount of space, like freeze dried food taken up into space for later reconstitution. When 'zipping' files to send them via email, one of these algorithms are used.

The basic idea of compression algorithms can be easily conveyed by an example. Suppose you see 14 ducks swimming in a row. One of them is bright yellow and the rest are brown. 'The yellow duckling' is a short way of picking out a particular duck (assuming no visual impairment), as is 'the second one behind the yellow duckling', 'the last duckling' or 'the fifth duckling from the front'.

An object like a binary string can be specified by the shortest program that produces it (akin to picking out the yellow duckling) or by giving its length n and its index in the set $\{0,1\}^n$ to which it belongs. There are many sets an object can belong to. For example, 256 is the 256^{th} natural number. In the set of powers of 2, it is 8^{th}. In the set of perfect squares, it is 16^{th}. In the set of three-digit zenzizenzizenzic[2] numbers it is the sole member.

There are three basic types of lossless compression algorithms: block-sorting, sliding window (similar to dictionary), and statistic.

Block-Sorting Algorithms

The first sort of compression algorithm uses the Burrows-Wheeler transform, the Move-to-Front transform, and a statistical compressor. An example of this type of compression algorithm is *bzip2*. An example is given below to demonstrate how it works.

[2] An outdated word meaning 'to the eighth power'.

Burrows-Wheeler Transform

Discovered by Wheeler in 1983 and published in a paper in 1994, this algorithm cleverly uses shifted permutations of a string to minimize the amount of space it takes up. For example, consider the string 'ALFALFA'.

Step 1 Create a matrix of shifted permutations of the word, moving the first letter of the string to the last position each time.

ALFALFA
LFALFAA
FALFAAL
ALFAALF
LFAALFA
FAALFAL
AALFALF

Step 2 Sort the list lexicographically.

AALFALF
ALFAALF
ALFALFA
FAALFAL
FALFAAL
LFAALFA
LFALFAA

Step 3 Record the last column of letters and note the number of the row where the string occurs

AALFAL **F**
ALFAAL **F**
ALFALF **A**
FAALFA **L**
FALFAA **L**
LFAALF **A**
LFALFA **A**
FFALLAA, 3

At this point, the Burrows-Wheeler Transform is complete. Note that the result appears to be more easily compressible: there are three pairs of repeated characters

in the string. The entropy of the sequence is approximately 1.56 bits.

Move-to-Front
Step 4 Create a lexicographically ordered list of all symbols that occur in the string.

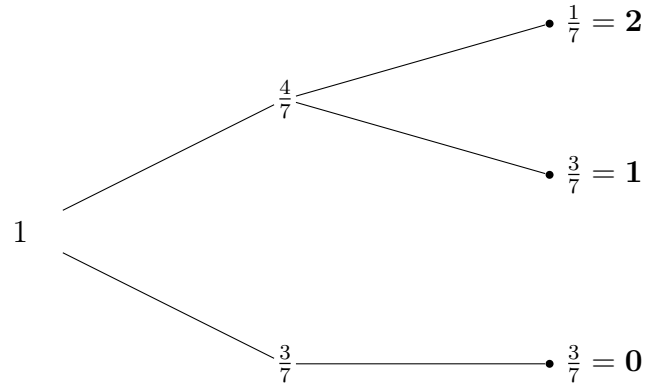
[A,F,L]

For each character in the string, record its index in the list and move the character to the front of the list. In computer science it is customary to begin indexing at 0.

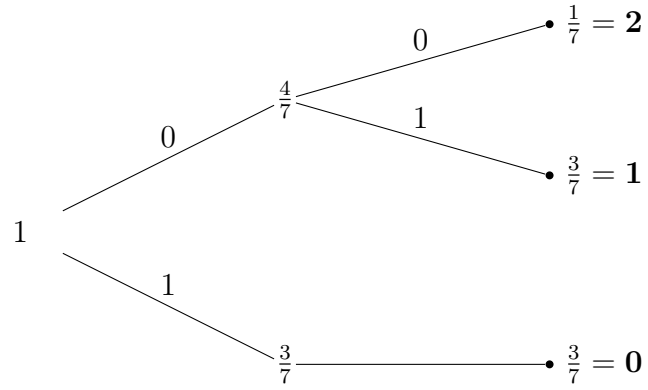
Letter	*Index in String*	*Output*	*List*
			[A,F,L]
F	1	1	[F,A,L]
F	0	10	[F,A,L]
A	1	101	[A,F,L]
L	2	1012	[L,A,F]
L	0	10120	[L,A,F]
A	1	101201	[A,L,F]
A	0	1012010	[A,L,F]

The resulting output is a string with a probability distribution that has less entropy, about 1.45 bits. While this may not seem like a big difference, keep in mind that the alphabet alone has 26 characters. When processing a block of text that includes punctuation and special characters the difference can be more pronounced.

Huffman Coding
Step 5 Encode the sequence by calculating the probability distribution of each character. Create a binary tree for the probability distribution by repeatedly merging the smallest probabilities together until they total 1.



Label each pair of branches with 0 and 1.



The codeword for each character is the concatenation of the binary characters leading to it.

Symbol	*Probability*	*Codeword*
0	$\frac{3}{7}$	1
1	$\frac{3}{7}$	01
2	$\frac{1}{7}$	00

The resulting code is uniquely decodable, prefix-free, and as short as possible given the probability distribution and these two constraints.

Original string: ALFALFA
Encoded string: 10010110100

A, L, and F are ASCII characters and each ASCII character takes up 8 bits when being stored. 0 and 1 only take up 1 bit each. *bzip2* has a default block size of 900 KB. If a string is larger than this, then it will be compressed in multiple blocks. In our example if the block size were 4 bytes then 'ALFA' and 'LFA' would have been compressed separately.

Sliding Window Algorithms

The next sort of compression algorithm splits a string into sections and searches for substrings occurring in the earlier string, also called the sliding window, to efficiently encode strings occurring in the lookahead window. After this encoding, statistical compression is performed. Again, we compress the string 'ALFALFA' to demonstrate the algorithm.

Step 1 Initialize the sliding window as empty and the lookahead as the entire string to be compressed. For each character k in the lookahed window, see if it occurred in the sliding window. If it does not, output the character 0 and the subscript 1 to indicate that the original character must be stored. If it does, find the longest substring in the sliding window that matches the characters "k, $k+1$, $k+2$,..., $k+n$" and output the number of characters before k that it occurs and the length n of the substring.

Word	*Sliding Window*	*Lookahead Window*	*Output*
ALFALFA		ALFALFA	0_1
A**L**FALFA	A	LFALFA	0_1 0_1
AL**F**ALFA	AL	FALFA	0_1 0_1 0_1
ALF**A**LFA	ALF	ALFA	0_1 0_1 0_1 3_4

Substrings may overlap with the lookahead window, like in the last step of processing the string 'ALFALFA': when the second 'A' is reached the substring 'ALFA' is found and the lookahead and sliding windows overlap by one character.

Step 2 Huffman coding is applied following the last step of the Block-Sorting Algorithm example.

An example of this type of compression algorithm is *gzip*. The sliding window size is 32 KB [13].

Statistic Algorithms

The final type of compression algorithm uses changes in the probability distributions of symbols, bigrams, trigrams, and other substrings occurring in a data stream to construct a code that is easily compressed using entropy coding, like Huffman or arithmetic coding. When starting to compress a file, it would be useful to know the probability distribution of each character in advance so codeword lengths could be chosen optimally. Furthermore, if one is compressing a specific data type, like a text written in the English language, then it is possible to simply pass the decoder an appropriate distribution (with "u" occurring after "q" with high probability, for example). Another option is to assume a uniform distribution over all characters at the outset and update their probabilities as the algorithm processes the stream.

Brilliantly, this one-pass algorithm requires no such knowledge or commitment in advance. The order of a Prediction by Partial Matching (PPM) algorithm specifies how many characters before the character that is being considered are used to search

for substrings ending in that letter. If a higher order match is not found, the algorithm searches for lower order substrings. For example, an order 3 PPM algorithm would search for trigrams and bigrams ending in the current character. Each time a new symbol is encountered in a given context, an escape symbol is given, followed by the symbol's seven-digit ASCII representation. This lets the decoder know to add the symbol to the alphabet it is using to reconstruct the string. Then, the new symbol is assigned a new probability in the relevant table. A useful feature of this algorithm is that the probabilities in the tables are changed incrementally in concert with the number of times a substring occurs.

The PPMZ algorithm, a specific implementation of this type, does not use a single-order model but performs a computation to estimate which higher-order contexts to use [48]. This type of algorithm performs well in practice. It also does not have a block size or sliding window that limits pattern recognition.

Objections

Specifications (like block size) play a large role in how efficiently data can be compressed. This is a problem for block-sorting and sliding window compression algorithms: if the length of a pattern exceeds the size of the window, then compression will not take place. To illustrate this point, consider the case in which the sliding window size is only two characters long and the word 'ALFALFA' is being compressed.

Word	*Sliding Window*	*Lookahead Window*	*Output*
ALFALFA		ALFALFA	0_1
A**L**FALFA	A	LFALFA	0_1 0_1
AL**F**ALFA	AL	FALFA	0_1 0_1 0_1
ALF**A**LFA	LF	ALFA	0_1 0_1 0_1 0_1
ALFA**L**FA	FA	LFA	0_1 0_1 0_1 0_1 0_1
ALFAL**F**A	AL	FA	0_1 0_1 0_1 0_1 0_1 0_1
ALFALF**A**	LF	A	0_1 0_1 0_1 0_1 0_1 0_1 0_1

In this extreme case, compression does not take place at all because no character occurs twice within three places of itself.

Compression algorithms do not pick up all types of structure. When new regularities are found new compression algorithms can be created (personal communication, P. Vitányi, May 22 2018). Current compression algorithms concentrate on the recurrence of substrings, for example, and do not detect arbitrarily large patterns.

4.2 Coding Theorem

In 1974 Leonid A. Levin proved a theorem relating the frequency of a string's production with its Kolmogorov complexity, see Li and Vitányi [37].

$$m(x) = 2^{-K(x)+O(1)}$$
$$-\log_2 m(x) = K(x) + O(1)$$

According to the theorem, the probability of a string being produced is equal to $2^{-K(x)}$. This means that if a string has low Kolmogorov complexity then it has a high probability. If a string has high Kolmogorov complexity then it has a low probability. This also connects the Kolmogorov complexity of a string to the Universal Distribution, a probability distribution that dominates all others, as discussed in Kirchherr, Li, and Vitányi [34]. This motivates the idea of approximating Kolmogorov complexity from below by seeing which strings are produced when running all possible computations.

Coding Theorem Method

Soler-Toscano et al. [50] ran a large number of 5-state, 2-symbol Turing machines until they had halted or run for more than 500 steps. The Turing machines were selected in such a way that those that were likely to halt were not included and machines that were redundant were not included. If a Turing machine halted, its output string was recorded. The frequencies of the production of strings were tracked. The strings that were produced most frequently included '0000000', '1111111', and '1010101'. Strings produced less frequently included '1110100' and '0010111'. Running Turing machines one by one for a certain number of steps, recording the output strings of machines that stop, and estimating the Kolmogorov complexity of the strings based on this probability distribution is called the Coding Theorem Method (CTM). Soler-Toscano et al. [50] tout the method as providing an alternative to the use of compression algorithms to approximate Kolmogorov complexity. This is particularly useful for short strings because compression algorithms do not work very well.

In Zenil et al. [62], the data from this experiment was used to check if strings that were predicted to have low Kolmogorov complexity by the CTM would be more compressible by compression algorithms than those that were predicted to have higher Kolmogorov complexity. Files were created by concatenating 100 strings of a specific length and range of predicted Kolmogorov complexity. Because the original strings were binary, an alphabet using pairs of letters from 'a' to 'p' was constructed (aa, ab, ac,...,ap,ba,bb,...,op,pp). This step was taken to minimize the compressor's

detection of coincidental patterns occurring in the file and instead compressing the strings themselves in turn. For example, if the following string was put into a file, a compression algorithm would try to compress 0's and 1's across the boundaries of the individual strings.

String 1: 0101100101
String 2: 1001101101
String 3: 1110010010
Concatenation: 01011001011001101101111001001O

Encoding the strings with different characters, while it may not prevent this effect altogether, should help keep the compressor in check.

String 1 ($0 = ap$, $1 = ck$): apckapckckapapckapck
String 2 ($0 = el$, $1 = fn$): fnelelfnfnelfnfnelfn
String 3 ($0 = jm$, $1 = go$: gogogojmjmgojmjmgojm
Concatenation: apckapckckapapckapckfnelelfnfnelfnfnelfngogogojmjmgojmjmgojm

100 strings of a particular range of predicted Kolmogorov complexity were put together in a file using this encoding procedure. 100 files of this type were made for each of the 10 partitioned frequency spaces. This procedure was performed separately for strings of length 10, 11, 12, 13, 14, and 15, yielding 6,000 files. The files were compressed using an implementation of DEFLATE (also called *zlib*) from Mathematica. DEFLATE uses sliding window compression discussed in the last section. They also used the block-sorting compression algorithm *bzip2* to see if the results would be different; they were not. The results indicated that generally, those files containing frequently produced strings were more compressible and infrequently produced strings were less compressible. This indicates that the Coding Theorem Method generates approximations of Kolmogorov complexity that are consistent with the same approximations by compression algorithms.

Block Decomposition Method

Zenil et al. [62] ran a similar experiment in two dimensions. Langton's Ant is a Turmite, or two-dimensional Turing machine, that has been shown to be capable of universal computation, see Figure 4.3.

Using a large sample of Turmites, roughly the same procedure as Soler-Toscano et al. [50] was performed. Machines were given a runtime of 1,500 steps and redundant machines were excluded. The output matrices of those machines that halted

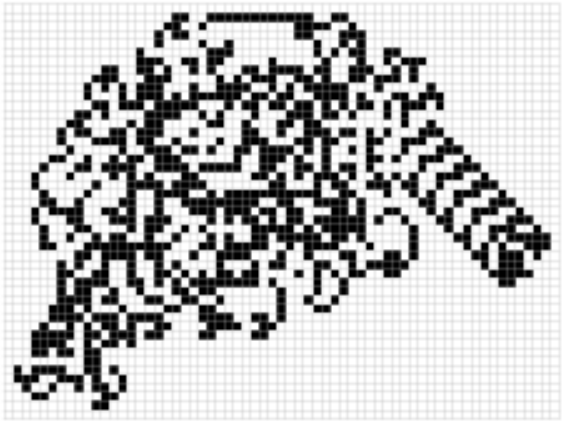


Figure 4.3: Langton's Ant after 10,647 steps, see Weisstein [60]

were used to construct a probability distribution over grids. Zenil et al. [62] use this probability distribution to estimate the Kolmogorov complexity of two-dimensional objects.

It could be the case that estimating Kolmogorov complexity in two-dimensions somehow does not work, so the results were checked in the following way. The set of all 4-state, 2-symbol Turing machines operating in one dimension is a subset of the set of all 4-state, 2-symbol Turing machines operating in two dimensions. A previous experiment produced data about the output strings of the former type of machine. This was compared with the data from the 4-state, 2-symbol experiment. The complexity of the strings from the two experiments was very strongly correlated. This lends weight to the results of the second experiment involving Turmites.

Using this chain of reasoning and the compressibility of the strings from the 5-state, 2-symbol experiment, Zenil et al. [62] motivate the use of the Block Decomposition Method (BDM). Counting the frequency of the occurrence of all 3×3 and 4×4 submatrices of the output of the Turmites, a probability distribution was calculated. This resulting probability distribution is used to estimate the Kolmogorov complexity of matrices. The Kolmogorov complexity of a larger matrix (greater than 4×4) is estimated to be the sum of the complexities of its subarrays plus some constant to join the subarrays. While it would be ideal to have these probability distributions over arbitrarily large $n \times n$ arrays, generating and analyzing the data is too computationally expensive at this point in time.

Objections

The excluding of redundant machines, filtering out of machines that will not halt, and discarding machines after a certain number of steps has an effect on the resulting probability distribution. Because the probability distribution is used to estimate Kolmogorov complexity, our confidence in this measure might be limited. In Soler-Toscano et al. [50], over 40% of 5-state, 2-symbol Turing machines are excluded. On the other hand, Calude and Stay [12] showed that if a program has not halted in a certain number of steps then the probability that it will is small and the researchers set a well-motivated runtime. Another issue is that neither 5-state, 2-symbol nor 4-state, 2-symbol two-dimensional Turing machines are universal. Without a universal Turing machine, Levin's proof of the Coding Theorem does not go through and it could be the case that the machines used do not approximate the correct distribution.

5 | Sophistication and Enduring Patterns

Textile producers struggle to come up with new patterns that are sufficiently appealing to grab a consumer's attention. Mathematics and computer science are employed in this pursuit. While exploring problems in number theory using the construction of stunted trees, Miller [41] created appealing tapestry designs. Ding and Shao [19] discussed the use of cellular automata in pattern design generation for knitting machines. One of the reasons for doing so was because designing satisfactory patterns is very time consuming. Marfo and Martey [40] used polar, ellipse and trigonometric functions in MATLAB to generate beautiful designs for use in embroidery, knitting, and fabric printing. The fact that researchers and manufacturers employ these methods to come up with new designs indicates that it is challenging.

Researchers have also attempted to pin down that which makes designs appealing. Schmidhuber [49] defined low-complexity art as art that can be defined by a computer program and obeys certain subjective constraints. The author illustrated his point with cartoons composed from sections of circles. He also commented on the subjectivity of taste and the possibility of human artistic taste being governed by an algorithm. Those aspects that Schmidhuber does not specify are precisely those that are most challenging to capture: subjective appeal. How can we tell when a pattern is appealing and when it has no order, and is it possible to write an algorithm that captures this?

There are also notions related to computability. Adriaans [2] gave an extensive list of proposed formal measures of meaningful information. Sophistication, from Koppel and Atlan [35] and others, is a measurement of complexity that can theoretically be used to discern random objects from ordered ones by quantifying only meaningful information. The more sophisticated an object is, the longer the shortest program that sufficiently describes it will need to be, where 'sufficiently describes' takes the form of a two-part code: a model, or representation of an object's struc-

tural properties, and noise that is in the object but not a part of its structure. This is an elegant concept because strings or objects that are incompressible are captured by short programs, for example a program that returns whatever object it is given. In essence the most random objects are filtered out, leaving an ordering of objects that should otherwise line up with our intuitions about what is simple and what is sophisticated, or perhaps complex in an appealing way.

Bloem, de Rooij, and Adriaans [10] gave two objections to theories of sophistication. The first is the problem of over-fitting and under-fitting. Finding the proper representation of an object is not a trivial matter. An approachable example given was the painting *Impressions of a Sunrise*: should it be viewed as an image? A painting? A series of pixel values? And are there any strokes in the painting that are not fundamental to its structure and instead constitute noise? The other problem concerns the length of a model of sophistication. An analog of the Invariance Theorem is hypothesized to hold for sophistication but Bloem, de Rooij, and Adriaans [10] illustrate that there is no reason to believe this is the case. On these grounds the authors propose that there is no such thing as sophistication, see Bloem et al. [10, page 4].

These considerations are important but they should not restrain us from empirically looking for a measure of meaningful information or something similar, even if it does not have such pleasing properties. Using compressors we may be able to find likely candidates. It might also be useful to see where compressors go wrong, confounding our expectations of what ought to be sophisticated and what ought to be random. We know that we do not have all possible compressors and it might shed light on types of regularity that are not yet captured by compression algorithms.

Others have investigated this question with a different strategy. Friedenberg and Liby [23] conducted a study where participants were shown screens containing different black and white pixel images. The images varied in complexity, amount of black space versus white space, and the placement of white squares. Two different conditions were used: images were sized as 10×10 pixels and as 15×15 pixels. Complexity was estimated using GIF compression as an approximation of Kolmogorov complexity. The results were interpreted as showing a correlation between complexity and beauty. As we have seen above, some approximations of Kolmogorov complexity may be better than others. While Friedenberg and Liby [23] used GIF compression, using PNG or other algorithms to estimate an image's complexity may yield different results. Gauvrit, Soler-Toscano, and Guida [25] analyzed the method and data from the same study and argued that the notions of entropy and complexity were not correctly interpreted in the data analysis. A wholly different measure of algorithmic complexity was used: instead of GIF, BDM from the previous section was used. Entropy was

interpreted as classical Shannon binary entropy, $H = -p\log_2(p) - (1-p)\log_2(1-p)$ (because there were only two colors used), which corresponds to the overall density of the color in the image. A reinterpretation of the results suggests participants displayed a preference for high entropy and low algorithmic complexity.

Here, we will take a further step. Where Friedenberg and Liby [23] examined random patterns, we will examine rough approximations of a small set of old and geographically diverse patterns. Taking these patterns in as simple a form as is reasonable, we will compress them using compression algorithms and measure them with BDM. We will also generate pseudo-random matrices to simulate white noise and compare their compressed lengths and BDM values while holding entropy constant. Can approximations of algorithmic complexity distinguish order from chaos in two dimensions in line with our intuitions?

Figure 5.1: A pair of pants likely worn for horse riding found in the Tarim Basin dated to the late 2$^{\text{nd}}$ millennium BC. Note the patterned band midway down the leg.

5.1 Patterns

Designs that have withstood the test of time are those that still look appealing. There may be a complex design in some textiles unearthed by archaeologists that we no longer have the capacity to recognize, but the author finds it unlikely that the human brain would have evolved so quickly in a few thousand years. We confine our attention to recognizable patterns.

The first pattern is found on a pair of pants that were discovered in an ancient cemetery in the Tarim Basin, today in western China, by Beck et al. [5], see Figure 5.1. The next is from the border of the Pazyryk rug, the oldest known carpet found so far, from the Altai region in southern Siberia, dated to the 5th Century BC (see Harris [28]). A sock from around 1,000 AD found in Egypt is also included from Bush [11], as is a simplification of the Selburose pattern from Norway that began to be used there in the mid-19th century. The basics of the patterns were put into a 17 $\times$ 17 matrices as shown in Figure 5.2, binarized, and put into CSV files.

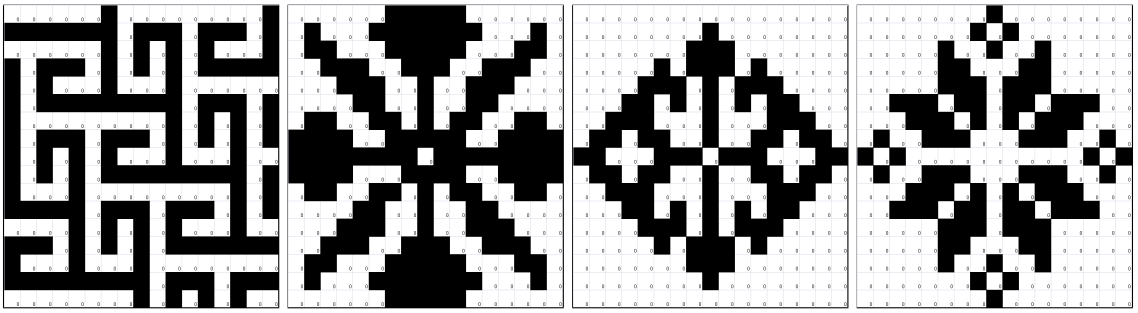


Figure 5.2: From the left, designs from Yanghai Pants, Pazyryk Rug, Egyptian Sock, and Selburose.

5.2 Pseudo-Random Designs

Using Excel's RAND() function, matrices were generated that have approximately the same entropy as the patterned matrices, see Figure 5.3.

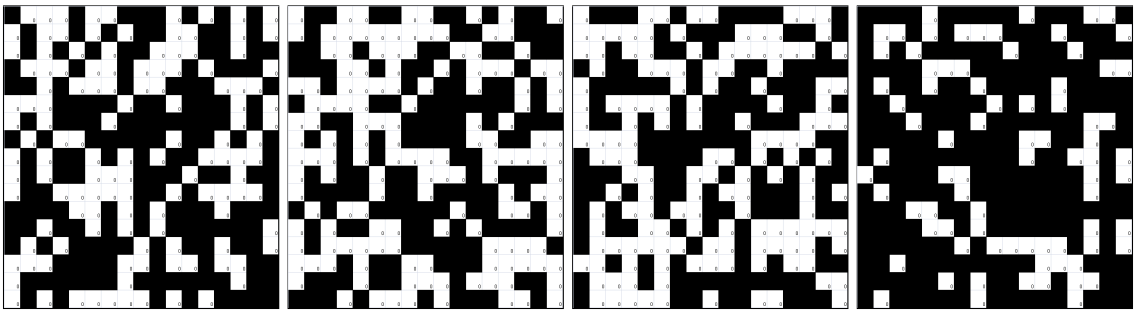


Figure 5.3: From the left, RNG1, RNG2, RNG3, and RNG4.

5.3 Algorithmic Complexity

All matrices were compressed using *LZMA*, *ZSTD*, *BZIP*, and *ZLIB*. The matrices were also measured using BDM, using both 3×3 and 4×4 methods from the `R` implementation developed by and discussed in Zenil, Soler-Toscano, Kiani, Hernández-Orozco, and Rueda-Toicen [63]. *PNG* is a compression algorithm as well as a storage format, so these values are also included. The results are shown in the table below.

Pattern	*Entropy*	*Compressed Lengths*				*BDM*		*PNG* Size
		LZMA	*ZSTD*	*BZIP*	*ZLIB*	4×4	3×3	
Yanghai Pants	1	136	127	101	101	232.6	258	4.2 KB
Pazyryk Rug	1	148	172	102	95	441.4	396.2	4.44 KB
Egyptian Sock	0.89	136	149	99	88	397.5	320.2	4.49 KB
Selburose	0.89	140	158	95	88	381.9	324.9	4.6 KB
RNG 1	1	172	144	114	118	482.8	396.3	4.46 KB
RNG 2	1	172	161	115	125	492.9	422.9	4.65 KB
RNG 3	1	180	161	116	126	494.2	425.4	4.56 KB
RNG 4	.89	164	155	111	114	455.3	368.5	4.03 KB
RNG 5	.88	164	151	111	118	478.7	370.7	4.71 KB

The two sample Kolmogorov-Smirnov (K-S) test is a non-parametric test that uses an empirical distribution function to check whether two different samples were taken from the same distribution. The test makes no assumptions about the distribution(s) from which the samples were drawn. For each compression algorithm and both BDM

options, let the first sample be the values of the patterns from textiles and let the second sample be the RNG values. The null hypothesis is that the samples were drawn from the same distribution. The alternative hypothesis is that the samples were drawn from different distributions. Because this is an exploratory investigation, let us set the α level, the chance of rejecting the null hypothesis incorrectly, to 2.5%. [1]

A middle line has been inserted into the results, shown in Table 5.3 to help the reader distinguish patterns from arbitrary matrices. The reader can observe some differences in the values above and below the separating line in the cases of *LZMA*, *BZIP*, and *ZLIB* compression algorithms in particular. The K-S test indicated that there was a difference in the distributions from which the samples were drawn for the *LZMA*, *BZIP*, and *ZLIB* compression algorithms as well as the 4×4 BDM values of patterns and RNG matrices. *ZSTD*, *PNG*, and BDM values do not distinguish well between the arbitrary and well-ordered matrices according to the K-S test at the 2.5% α level.

Entropy alone is insufficient to account for our judgments of beauty, order, or appeal. If it were sufficient then the leftmost images in Figure 5.2 and Figure 5.3 should be equally beautiful, orderly, or appealing. Algorithmic complexity appears to capture something more. The exploratory investigation here is not meant to conclusively demonstrate that sophistication exists, only to sow doubts about the assertion that it does not. While we may not have a model that distinguishes perfectly between patterns and arbitrary matrices, these results suggest we should not abandon our search for an algorithmic model of meaningful information. The next step might be to try holding both entropy and algorithmic complexity constant while varying meaningful information. If this can be done then perhaps there is no reasonable measure.

[1] 5% is the α level frequently used in studies but is not a stringent standard in the opinion of the author. The debate over statistical significance is ongoing but including statistical test results is customary, insufficient as they may be.

6 | Measuring Similarity

Techniques used in data mining and artificial intelligence often rely on statistics. Statistics are found by using a function (like averaging) on measurements taken from data in a sample that is assumed to represent a larger population. The fundamental assumption underlying statistical inference is the Law of Large Numbers. Thus in order for these methods to work, a great deal of data is required to reveal the underlying central tendency. However, there are other measures to be considered.

Given two objects, one way to consider what they have in common is by thinking about what changes would have to be made to one to turn it into the other.

(a) 0101010101010101
(b) 0110010101010101

For example, given string (a), one only has to switch the third and fourth digits of the sequence to get string (b). String (a) is easily transformed into string (b) and vice versa, thus we can say that strings (a) and (b) are similar. This is the fundamental idea motivating Normalized Compression Distance (NCD): the length of a program that produces x given y (or y given x) is a fair measurement of their similarity. In practice, compression algorithms are used to estimate this distance. To motivate the use of this measure, first the concepts of conditional Kolmogorov complexity, Information Distance, and Normalized Information Distance will be discussed. Another way of quantifying change is through Algorithmic Calculus. This is another parameter-free way of measuring change using the Coding Theorem Method or Block Decomposition Method. This technique and some of its applications will also be discussed.

6.1 Conditional Kolmogorov Complexity and Information Distance

As explained above, the Kolmogorov complexity of a string x is the shortest program that produces it. This is usually written $C(x)$. $K(x)$ is the prefix-free version of Kolmogorov complexity and is often discussed because of its useful properties, thus we will employ it here. The shortest program that produces one string when given some other string as input is called the conditional Kolmogorov complexity of x given y and is written $K(x|y)$. Above, we would signify the shortest program that produces string (a) given string (b) as $K(a|b)$.

We call the longer of $K(x|y)$ and $K(y|x)$ the information distance between x and y. The larger one is used because the two are not necessarily the same. Consider $K(\epsilon|x)$ where ϵ is the empty string. For any string x, this is small: just produce the empty string, regardless of the input. But for most x, $K(x|\epsilon)$ should be a good deal longer, see Bennett et al. [8] for details.

6.2 Normalized Information Distance

Suppose two strings (c) and (d) that you are comparing are quite long, say 1,000,000,000 bits, but their only difference is the transposition of two digits at the very beginning of the string.

(c) 101010101010...
(d) 011010101010...

Compare $K(c|d)$ with $K(a|b)$. Although both pairs are only slightly different from one another, strings (c) and (d) are a lot longer than strings (a) and (b). Because of this, strings (c) and (d) are more similar than strings (a) and (b) even though they have about the same information distance.

In order to put the similarities into context, we normalize them by dividing by the larger of the Kolmogorov complexities of the strings being compared.

$$NID(x,y) = \frac{max\{K(x|y),K(y|x)\}}{max\{K(x),K(y)\}} = \frac{max\{K(x,y)-K(x),K(x,y)-K(y)\}}{max\{K(x),K(y)\}}$$

where $K(x,y)$ is the shortest program that produces x and y together. The Kolmogorov complexity of a string, as stated in the previous section, is uncomputable. The conditional Kolmogorov complexity of one string with respect to a different string

is similarly uncomputable. The Normalized Information Distance between two different strings is not even semi-computable, see Terwijn, Torenvliet, and Vitanyi [56]. Thus we calculate something different.

6.3 Normalized Compression Distance

Using compression to approximate Kolmogorov complexity, there is a way to approximate the Normalized Information Distance between two strings. Let $C(x)$ be the compressed length of the string x with respect to some reference compressor (similarly for the concatenation $C(x, y)$ of two strings).

$$NCD(x, y) = \frac{min\{C(x,y),C(y,x)\}-min\{C(x),C(y)\}}{max\{C(x),C(y)\}}$$

Due to the current imperfect nature of compressors, it is often the case that $C(x, y) \neq C(y, x)$. In order to avoid overestimating the compression distance the smaller of the two is used. Cilibrasi and Vitányi [15] shows that if the compressor used satisfies the properties of idempotency, monotonicity, symmetry, and distributivity up to an additive term, then NCD is a similarity metric.

Unlike neural networks and other models that require large amounts of data, training and additional assumptions about the data, NCD is a parameter-free, feature-free, and alignment-free way of measuring all types of similarity. Intuitively the NCD metric should be able to capture any sort of similarity a compressor can.

6.4 Applications

This technique has been used to construct phylogeny trees, structures that reveal relationships among objects in a class. Cilibrasi and Vitányi [15] used mammalian genetic material, the Universal Declaration of Human Rights written in 52 different languages, Russian literature, music, handwritten digits and other data sets to show the robustness of the method.

An interesting and approachable application was done by Bennett, Li, and Ma [9]. Chain letters attempt to convince a recipient to copy the letter's contents and send it onwards to other recipients. While many chain letters nowadays might be sent via email, they used to be sent by post. Transmission of a chain letter is similar to a game of 'Telephone', where small mutations that happen through iterated noisy transmissions might yield quite a different result at the end of the chain. A chain letter might be photocopied several times and then become illegible, necessitating a retyping of the letter and giving the typist an opportunity to introduce mutations.

A collection of 33 chain letters were typed into text files and compressed. A tree was built using a likely candidate as a root node. Some interesting phenomena were observed through the mutations of the letters over time. Place names, amounts of money, and even the title of the letter were changed through repeated transmissions.

When using NCD, it is important that an appropriate reference compressor is used. Some compressors perform better than others on particular types of data because of differences in regularities. For example, DNA sequences tend to have long runs of a single letter, like 'AAAAA' and reverse complement structures (see Figure 6.1), where the nucleotides adenine and thymine pair together and so do guanine and cytosine. Strands of DNA twist together and when they are looked at in a single strand, the probability of the occurrence of some sequences is higher after others.

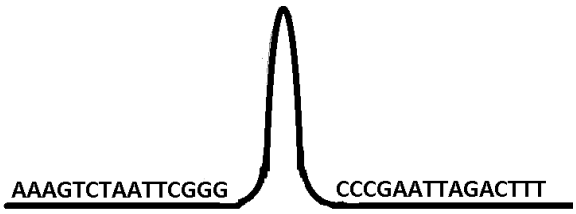


Figure 6.1: The reverse complement structure in DNA. A pairs with G and C pairs with T. If the letters on one side were changed into their complement nucleotides, the sequences would be symmetric.

Using one of the types of compressors we have looked at so far might not work as well on DNA. GenCompress, an algorithm specifically designed for use with DNA sequences, is a one-pass algorithm that searches for prefixes of a string that can be changed slightly to form a substring that occurs later as explained in Chen, Kwong, and Li [14]. While this might seem like a minor change, it makes a substantial difference when compressing large amounts of data.

Testing a compressor for use with a particular data type is an assessment of its idempotency. One useful way to test a reference compressor is to take a large set of data and compress it with itself or with two copies of itself. $NID(x, x)$ should be 0 (plus some constant) for all x and between 0 and 1 for all x and y. $NCD(x, y)$ in practice is between 0 and 1.1 because compressors do not work perfectly. Testing several types of compression algorithms for compatibility with a data type is an important step to take before measuring the NCD between objects of a given type.

A legitimate concern when choosing a compressor is block size or sliding window size. As illustrated in Section 3.1, using too small of a sliding window size will

prevent compression from taking place. In an empirical investigation Cebrián Ramos, Alfonseca, and Ortega [13] demonstrated how NCD goes awry when objects are compressed that do not fit inside compression blocks or sliding windows. In order to use these types of algorithms, the concatenation of the two objects being compressed must fit inside of the block or sliding window.

6.5 NCD with Images

In this paper we apply NCD to images of patterns found in textiles. Images are often large files and one must exercise caution when selecting a compressor.

Images are stored compressed, like in Portable Network Graphics (*PNG*) or *JPEG* files, or uncompressed, as in bitmap files. Storing uncompressed images takes up a great deal more space because the value of each pixel is stored separately. Image data is also stored, generally speaking, with a Row Major read-in. Starting from the top right pixel, data is read into a file all the way to the leftmost pixel, then the reader starts from the second pixel down on the right and continues left, and so forth like reading an English language book.

When using NCD on images, the method of linearization matters in at least some cases. A study was conducted where images were compressed with transformations of themselves. The translations were vertical and horizontal shifts, rotations, and vertical and horizontal flips. Methods of linearization used included Row-Major, Column-Major, and the Hilbert-Peano curve. Mortensen et al. [42] showed that the Row-Major linearization compressed the horizontally shifted images better and similarly for Column-Major and vertically shifted images. Several compressors were tested and the results were consistent. This demonstrates that NCD is sensitive to the method of linearization in images, regardless of the compressor used. When using NCD to measure image similarity, one needs to take care that the method of linearization does not unduly influence the results.

Vázquez and Marco [59] tested the performance of various compression algorithms when calculating NCD for image data. A variety of images were compressed with respect to themselves to check the idempotency of compressors. Both uncompressed and compressed image formats were used in the experiment. Results indicate that bitmap format was suitable and that Prediction by Partial Matching compressors, of the statistical compressor family, performed best as compressors. Block-based algorithms performed reasonably well when the image was small.

6.6 Algorithmic Calculus

Zenil et al. [64] show how the Coding Theorem Method and the Block Decomposition Method can be used to infer an underlying generating mechanism from changes towards and away from randomness. The general idea is to estimate the algorithmic complexity in a system at different points using CDT or BDM methods (mentioned above) and calculate the differences between them, checking to see if the system is becoming more or less random. Basically, one takes a set of observations, calculates the change in estimated algorithmic complexity over time, and finds the derivative of the function. An example involving an elementary cellular automaton is used. Starting from a scrambled set of observations, the observations are ordered by their complexity. Changes from one step to another are mapped to possible cellular automaton rules, showing how the generating mechanism can be inferred by the process of elimination. While elementary cellular automata are simple, this motivates how changes in more complex systems can be measured to reveal their generating mechanisms from a disorganized set. More complex applications were discussed, see Zenil et al. [64] for more information. BDM is also being assessed as a texture descriptor in imaging studies focused on brain tumors in the forthcoming paper Rueda-Toicen, Kiani, and Zenil [47].

Now that we have explored the concept of algorithmic complexity, ways of approximating it, and how these approximations can be used to calculate similarity, we will apply these methods to an interesting data set.

7 | Algorithmic Complexity and the Turkmen Textiles

Suppose we collected patterns from all over the world and used NCD to measure similarity among them. We can anticipate that patterns from geographically diverse places will be related: recall the similar designs from Morocco and Guatemala in Figure 1.1 and the meander algorithms used to generate designs by the Tamil, Tchkokwe, Romans, and others like the examples shown in Figure 4.1. These similarities, while interesting, may not necessarily indicate culture exchange or common ancestry. As indicated earlier people find a particular range of complexity appealing and this is indicated through the designs being carried forward by people who choose to reproduce them. Perhaps people since the dawn of humanity have carried the same notion of appealing designs and observing textile patterns from everywhere could give us clues about how people traveled over time and how they are related. This question is outside the scope of this thesis.

Rather than quantify over all textiles, we can examine a particular subset of designs through the lens of algorithmic complexity. The woven ornaments of the Turkmen are the ideal case for NCD analysis to tackle. We can also experiment with the algorithmic calculus from Zenil, Kiani, Marabita, Deng, Elias, Schmidt, Ball, and Tegner [64]. As explained in the previous chapter, anthropologists, art historians, and collectors struggle to understand the relationships between ornaments found in Turkmen textiles. This is part of a wider mystery about the genealogy of the Turkmen tribes, their movements, and their cultural evolution. Phylogenetic analysis of these textiles has already been done by Tehrani and Collard [54]. Looking at similarities among tribal ornaments can provide information about the genealogy of the tribes.

Historical records establish that the Turkmen have a common origin. Weaving is a cultural tradition among tribeswomen and is central to domestic life. In place of furniture, Turkmen use woven bags hung from the walls of their yurts. The main carpet of the room supposedly displays important tribal emblems (although the

author finds this point contentious). The door coverings, saddle bags, and traditional wedding decorations for camels are all woven. Women from different tribes are culturally segregated from one another. Weaving skills are passed from mother to daughter and intermarriage between tribes is not common. Some theories of the evolution of Turkmen ornaments posits an original gol. As tribes spread out from the Aral Steppe and wandered in different directions, the women kept their skills and patterns, and passing them on through generations. The small permutations that occurred through the transmission of these designs resulted in the independent evolution of ornaments. By measuring the similarity between the ornaments using NCD, a parameter-free metric, we can trace the evolution of the gol and derive a genealogy of Turkmen tribes.

The gols of the Salor, Tekke, Saryk, and Ersari are relatively well accepted in the literature (see Figure 7.1). Using these gols as a starting point, we can derive a genealogy based on NCD and see if it is consistent with other information about Turkmen history. After this, gols of uncertain provenance can be added to the analysis. While the names of the tribes may remain unknown, relationships can still be found.

7.1 Linearization of Gols

Gols found on rugs, bags, and other items appear in a multitude of vibrant colors, but their shape is consistent. Azadi [3] claims these ornaments are best analyzed through three parts:

- The geometry of the outline,
- The design of the interior, and
- The shape of the center.

The third feature is hypothesized to indicate the subdivision of a tribe. We will leave it out of our consideration and focus on the first two features.

Sketches are commonly used in the literature to display the main features of gols. These are generated by observing photographs of textiles or the woven items themselves. The sketches of tribal gols used in Tehrani and Collard [54] are used as a starting point along with an Arabatchi gol sketched from a *chuval* featured at the Metropolitan Museum of Art as shown in Figure 7.2.

As discussed in Mortensen et al. [42] the method of linearization matters when measuring NCD for image data. Keeping this in mind, the gols were oriented in

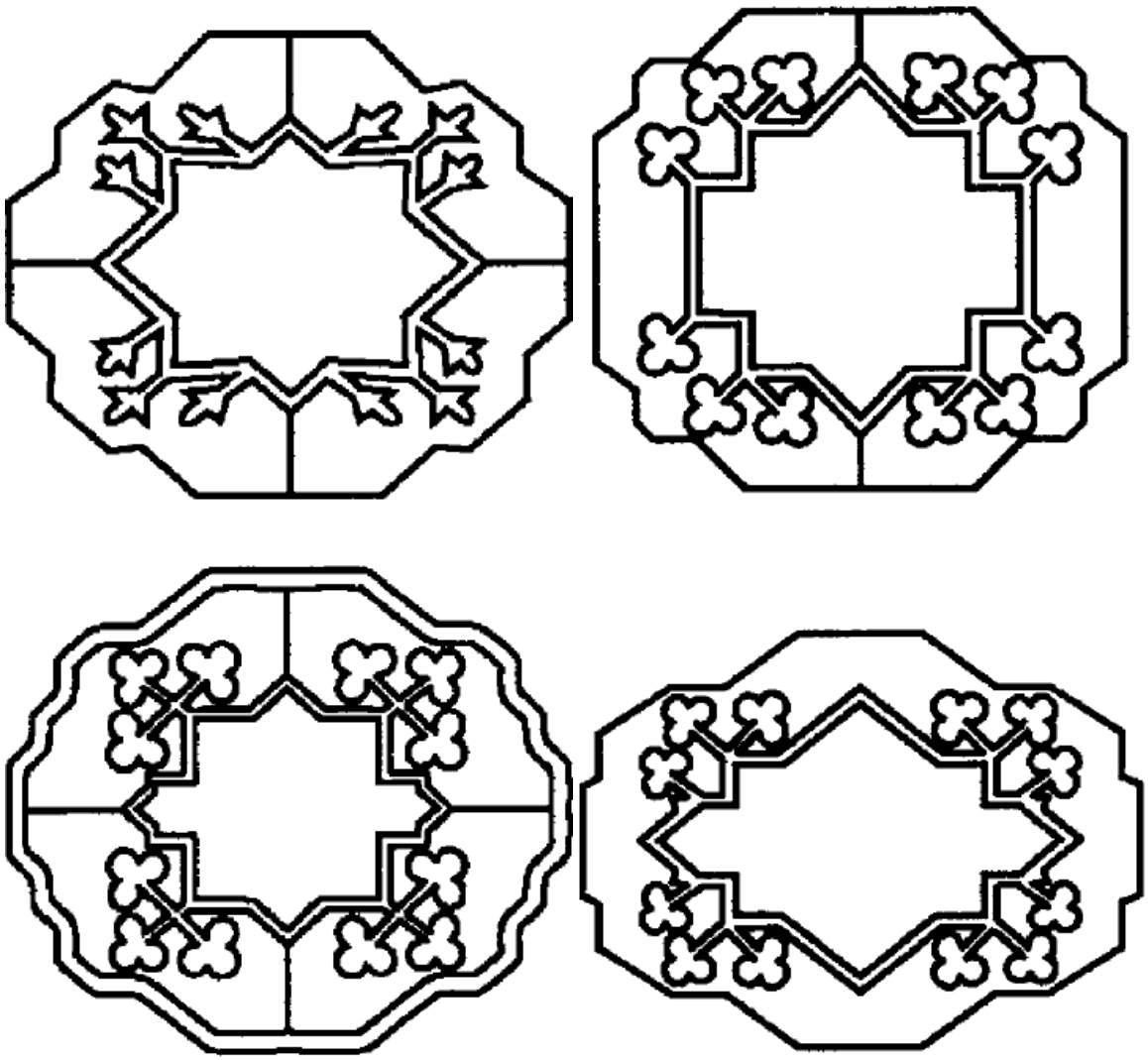


Figure 7.1: Sketches of Guls commonly used in phylogenetic analysis.

the same direction, centered in a square area and resized to 256×256 pixels using Python image processing. Using binary thresholding, the pixel values of the image were changed into 1's and 0's and put into CSV format. These CSV files were edited in Excel to ensure that noise (in the form of stray black or white pixels) was removed. The CSVs were then read into text files using a Row Major scan.

Figure 7.2: An Arabatchi *chuval* from the Metropolitan Museum of Art.

7.2 Implementation

After checking several compressors for idempotency using a large binary file, the files were compressed using LZMA, a modified sliding window algorithm. The sizes of the file were less than 135 KB each. The dictionary storing substrings for the algorithm is up to 4 GB, well over twice the size of any two files put together. This is crucial: if the concatenation of any two files were larger than this, the algorithm would not be able to compress the files properly. NCD was measured using a modified version of Landman [36] and Python code from a creator of `CompLearn` (R. Cilibrasi, personal communication, May 2, 2018) and a phylogenetic tree was built using BioPython's `Phylo` module, see Talevich et al. [53] and Cock et al. [16].

The algorithmic complexity of the matrices was also estimated using the 4×4 BDM method using the `R` implementation from Zenil et al. [63].

7.3 Results

The results of the NCD analysis show the Tekke and Saryk tribes as being related, the Ersari and Arabatchi tribes being related, and the Salor as sharing a common ancestor with all four, see Figure 7.3.

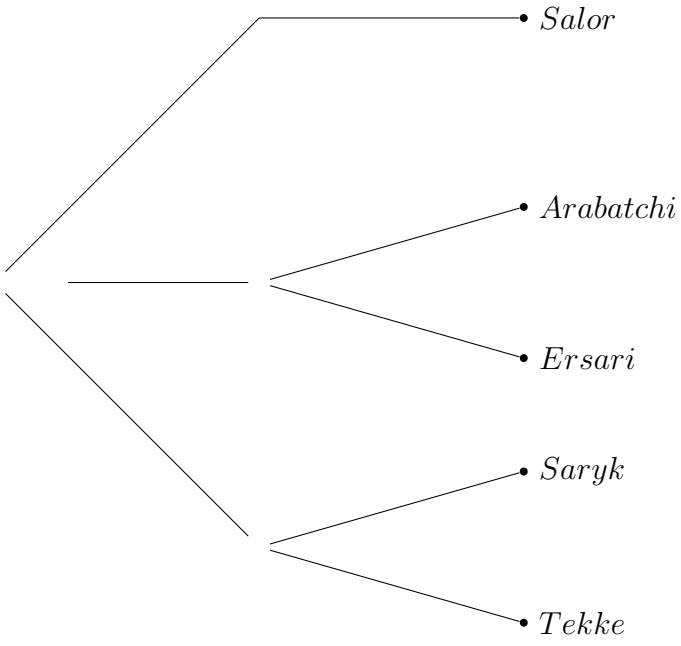


Figure 7.3: Phylogenetic tree based on NCD metric.

The BDM values are below in Table 7.1. They are consistent with the theory that the Ersari and Arabatchi tribes are related and the Tekke and Saryk are related.

Gol	*BDM, 4 × 4*
Arabatchi	8294.35
Ersari	9120.63
Saryk	9675.015
Tekke	10256.4
Salor	12256.81

Table 7.1: BDM Values for gols.

7.4 Discussion

A phylogenetic analysis was carried out using five ornaments from Turkmen textiles to find genealogical relations between the tribes. The analysis indicates that the Saryk and Tekke tribes are related and the Arabatchi and Ersari tribes are related.

Abul Ghazi Bahadur indicated that the Tekke and Saryk tribes were descendants of the Salor Toi-Tumas. This tree is consistent with the Tekke and Saryk tribes being related, but their connection to the Salor is less clear. Perhaps the Salor that used the gol rendered in this paper were not the Toi-Tumas, but another branch of the Salor tribe. In the English translation of Moskova's work, Franses and Eiland explain how some have speculated that the Arabatchi are a branch or subgroup of the Ersari [43]. This analysis provides support for that hypothesis.

Another phylogenetic analysis by Tehrani and Collard [54] (shown in Figure 7.4) suggested that the Ersari and Salor were related and that the Saryk were the most closely related to these two tribes. This conflicts with our analysis. However, separate nomenclature data is consistent with the tree from Tehrani and Collard [54], as is the geographic distribution of the tribes. This leads the researchers to question the accuracy of Abul Ghazi Bahadur's genealogy and recommend disregarding it altogether.

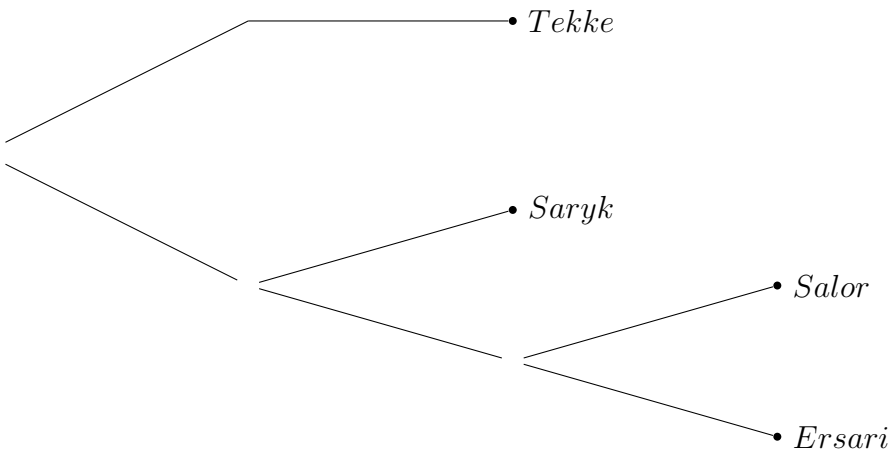


Figure 7.4: The results of the analysis from Tehrani and Collard [54].

Let us more closely examine the evidence supporting the analysis of Tehrani and Collard [54]. The first reason given is nomenclature data: according to an essay by Wood [61], individual names of clans within the Salor, Ersari, and Saryk tribes are claimed to be from the Oghuz lexicon but the Yomut and Tekke tribes have clan

names with Persian influences. This seems like shaky ground on which to stake a claim.

Another reason given is the geographic distribution of the tribes because the Ersari, Salor, and Saryk lived closer to Bokhara and Sarakhs while the Yomut and Tekke lived in Khorassan. Maps indicate these areas coincide. Furthermore the tribes were nomadic and their current geographic distribution does not necessarily serve as an indication of where they have lived or travelled.

Finally, Tehrani and Collard [54] conclude that the genealogy of Abul Ghazi Bahadur is flawed and should be disregarded because Turkmen tribes use their common origins to solidify cooperation and as a result the genealogical relationships related to Abul Ghazi Bahadur in Turkic oral traditions may have been overstated. The records of tribute indicate that the Saryk and Tekke may have had an incentive to exaggerate their relationship with the Salor because the Salor would have been the dominant tribe. Irons [32] indicated that groups depending on better established, larger tribes (tribes that would incorporate the smaller groups into their lineage) would emphasize genealogical ties for social or economic reasons. However, the tribute payments indicate that the Tekke and Saryk tribes were independent and sizeable, making it highly doubtful that they were a group dependent on a larger tribe. The fact that the Tekke and Saryk tribes were still distinct centuries later makes it clear that they were not being incorporated into the Salor and thus would not have had an incentive to fictionalize their ties. It is not reasonable to disqualify one of the few historical records attesting to the genealogy of the Turkmen on these grounds.

The analysis based on the algorithmic complexity of the textiles supports the historical genealogy of Abul Ghaza Bahadur, but let us consider the worst case scenario: suppose the gols of the tribes are not accurate attributed. The assumption that the gols are representative of the Salor, Ersari, Tekke, and other tribes is disposable. If we later discover that one or more of these ornaments is mistakenly attributed, this approach will still show a measure of similarity between whichever tribes produced these gols and any mysterious gols that may be unearthed in the future. As discussed in the previous section, much remains unknown about the movements and history of the Turkmen. Using parameter-free methods like calculating the algorithmic complexity of objects, whether they are actually emblems of the tribes or not, can show relationships that are otherwise invisible. Tribes might later be attached to gols through such analyses. The gols depicted in this paper are not the only ones: there are also *temirjin*, *qaradashli*, *erre*, *ertmen*, *kepse*, and *ayna* gols referenced in the literature and depicted in such books as Mackie and Thompson [39] and Pinner and Eiland [46]. Future analyses might use a wider variety of gols to investigate tribal genealogy. Rugs in Iran, Turkey, and other places also have octagonal ornaments.

Including these could indicate points of contact and help settle the debate surrounding the origins of the gol and generate a larger genealogy of peoples of Turkic origin. While these analyses may not be provide conclusive proof of relationships, their tight correspondence with scientific consensus in other fields such as biology shows they are robust.

NCD analysis can be used more broadly in art history, especially when few taxa are available. While neural networks and other tools may prove useful when there are very large numbers of taxa, NCD is sufficiently robust to deal with very few as well. Applying these techniques to images found on fragments of pottery and stone carvings could provide clues about the past.

8 | Conclusion

Algorithmic complexity is a useful tool for quantifying patterns in two dimensions. It is usually discussed with respect to binary strings but textile production methods provide a useful space in which to explore these concepts. We have also seen how knitting can simulate a Turing machine.

Attempts to capture only meaningful information have been made by Koppel and Atlan [35], Adriaans [2] and others and debated by Bloem, de Rooij, and Adriaans [10]. Gauvrit, Soler-Toscano, and Guida [25], Gauvrit, Soler-Toscano, and Zenil [24], and Friedenberg and Liby [23] conducted studies to examine how algorithmic complexity, entropy, and human judgments of beauty correlate. Using pieces of old textiles we have checked to see if the methods we have to estimate Kolmogorov complexity distinguish between arbitrary matrices and well-ordered ones with the same entropy. Even though the sample is quite small, the results are intriguing and we should not dismiss the notion of a computable measure of meaningful information so quickly.

Finally the potential of NCD and other estimations of Kolmogorov complexity to answer questions about genealogical relationships in the field of material culture, a branch of anthropology, was demonstrated. By investigating the genealogy of the Turkmen tribes using only superficial features of the ornaments used we built a phylogenetic tree that supports historical accounts of the Turkmen genealogy and suspicions of rug scholars.

Whether motivating archaeologists and art historians to use feature-free tools such as NCD for analysis or inspiring knitters to find shorter programs, algorithmic complexity provides interesting tools and ideas for two-dimensional pursuits.

Bibliography

[1] Computational Model of Knitting. `http://www.k2g2.org/blog:bit.craft:computational_model_of_knitting`. Accessed: 2018-06-06.

[2] Pieter Adriaans. Facticity as the amount of self-descriptive information in a data set. *arXiv preprint arXiv:1203.2245*, 2012.

[3] Siawosch Azadi. *Turkoman carpets and the ethnographic significance of their ornaments*. Crosby Press, 1975.

[4] Patricia L Baker. Twentieth-century myth-making: Persian tribal rugs. *Journal of Design History*, 10(4):363–374, 1997.

[5] Ulrike Beck, Mayke Wagner, Xiao Li, Desmond Durkin-Meisterernst, and Pavel E Tarasov. The invention of trousers and its likely affiliation with horseback riding and mobility: A case study of late 2nd millennium bc finds from turfan in eastern central asia. *Quaternary international*, 348:224–235, 2014.

[6] Sarah-Marie Belcastro. Every topological surface can be knit: a proof. *Journal of Mathematics and the Arts*, 3(2):67–83, 2009.

[7] Sarah-Marie Belcastro. Adventures in mathematical knitting. *American Scientist*, 101(2):124, 2013.

[8] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *IEEE Transactions on information theory*, 44(4): 1407–1423, 1998.

[9] Charles H Bennett, Ming Li, and Bin Ma. Chain letters & evolutionary histories. *Scientific American*, 288(6):76–81, 2003.

[10] Peter Bloem, Steven de Rooij, and Pieter Adriaans. Two problems for sophistication. In *International Conference on Algorithmic Learning Theory*, pages 379–394. Springer, 2015.

[11] Nancy Bush. *Folk Socks: The History and Techniques of Handknitted Footwear, Updated Edition*. Interweave Press, LLC, 2012.

[12] Cristian S Calude and Michael A Stay. Most programs stop quickly or never halt. *Advances in Applied Mathematics*, 40(3):295–308, 2008.

[13] Manuel Cebrián Ramos, Manuel Alfonseca, and Alfonso Ortega. Common pitfalls using the normalized compression distance: What to watch out for in a compressor. *Communications in information and systems*, 2005.

[14] Xin Chen, Sam Kwong, and Ming Li. A compression algorithm for dna sequences. *IEEE Engineering in Medicine and Biology Magazine*, 20(4):61–66, 2001.

[15] Rudi Cilibrasi and Paul MB Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[16] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[17] Matthew Cook. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

[18] Michael David and P Saunders. The new turkoman mythology. *Tribal visions*, pages 17–22, 1980.

[19] Yongsheng Ding and Shihuang Shao. Intelligent computation in the computerized flat knitting systems. In *Proceedings of the WSES International Conferences on Mathematics and Computers in Mechanical Engineering (MCME'99)*, pages 491–500, 2012.

[20] Matthew S. Dryer and Martin Haspelmath. The world atlas of language structures online. 2013.

[21] Bui Ha Duong. An information-theoretic approach to origami folding sequence generation from 3d shape models. Master's thesis, Japan Advanced Institute of Science and Technology, School of Information Science, 2 2017.

[22] Murray L Eiland Jr. Turkomans and scholarship: A retrospective view. *Oriental Rug Review*, 8:2, 1988.

[23] Jay Friedenberg and Bruce Liby. Perceived beauty of random texture patterns: A preference for complexity. *Acta psychologica*, 168:41–49, 2016.

[24] Nicolas Gauvrit, Fernando Soler-Toscano, and Hector Zenil. Natural scene statistics mediate the perception of image complexity. *Visual Cognition*, 22 (8):1084–1091, 2014.

[25] Nicolas Gauvrit, Fernando Soler-Toscano, and Alessandro Guida. A preference for some types of complexity comment on "perceived beauty of random texture patterns: A preference for complexity". *Acta psychologica*, 174:48–53, 2017.

[26] Paulus Gerdes. *Sona geometry from Angola: Mathematics of an African tradition*. Polimetrica, 2006.

[27] Carmela Rosalba Guglielmino, Carla Viganotti, Barry Hewlett, and Luigi Luca Cavalli-Sforza. Cultural variation in africa: Role of mechanisms of transmission and adaptation. *Proceedings of the National Academy of Sciences*, 92(16):7585–7589, 1995.

[28] Jennifer Harris. *5000 years of textiles*. British Museum Press, 1993.

[29] Henkjan Honing. Without it no music: beat induction as a fundamental musical trait. *Annals of the New York Academy of Sciences*, 1252(1):85–91, 2012.

[30] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 2001.

[31] Kris Howard. Knit one, compute one. linux.conf.au, 2017. URL `https://www.youtube.com/watch?v=7bHifmcVRQg`.

[32] William Irons. Nomadism as a political adaptation: the case of the yomut turkmen. *American Ethnologist*, 1(4):635–658, 1974.

[33] William Irons. Yomut family organization and demography. *The Human Biology of Pastoral Populations*, 30:251, 2002.

[34] Walter Kirchherr, Ming Li, and Paul Vitányi. The miraculous universal distribution. *The Mathematical Intelligencer*, 19(4):7–15, 1997.

[35] Moshe Koppel and Henri Atlan. An almost machine-independent theory of program-length complexity, sophistication, and induction. *Information Sciences*, 56(1-3):23–33, 1991.

[36] Davy Landman. Ncd. `https://github.com/DavyLandman/ncd`, 2015.

[37] Ming Li and Paul MB Vitányi. *Kolmogorov complexity and its applications*. Centre for Mathematics and Computer Science, 1989.

[38] Yang Liu and Godfried Toussaint. Unravelling roman mosaic meander patterns: a simple algorithm for their generation. *Journal of Mathematics and the Arts*, 4(1):1–11, 2010.

[39] Louise W Mackie and Jon Thompson. *Turkmen: Tribal carpets and traditions*. Textile Museum, 1980.

[40] Joseph Marfo and Ezekiel Mensah Martey. Creating designs through mathematical functions. *International Journal of Computer Applications*, 123(3), 2015.

[41] Jeffrey CP Miller. Periodic forests of stunted trees. *Phil. Trans. R. Soc. Lond. A*, 266(1172):63–111, 1970.

[42] Jonathan Mortensen, Jia Jie Wu, Jacob Furst, John Rogers, and Daniela Raicu. Effect of image linearization on normalized compression distance. In *Signal Processing, Image Processing and Pattern Recognition*, pages 106–116. Springer, 2009.

[43] VG Moshkova. Tribal göl in turkoman carpets. *from Pinner, Robert, and Michael Franses, Turkoman Studies I: Aspects of the Weaving and Decorative Arts of Central Asia*, 1980.

[44] Hasan B Paksoy. *Essays on Central Asia*. Carrie/EUI, 1999.

[45] Sungmee Park and Sundaresan Jayaraman. The wearables revolution and big data: the textile lineage. *The Journal of The Textile Institute*, 108(4):605–614, 2017.

[46] Robert Pinner and Murray L Eiland. *Between the Black Desert and the Red: Turkmen Carpets from the Wiedersperg Collection*. Fine Arts Museum of San Francisco, 1999.

[47] Antonio Rueda-Toicen, Narsis Kiani, and Hector Zenil. Texture analysis through estimations of kolmogorov complexity, 2018.

[48] David Salomon. *Data compression: the complete reference*. Springer Science & Business Media, 2004.

[49] Jürgen Schmidhuber. Low-complexity art. *Leonardo*, pages 97–103, 1997.

[50] Fernando Soler-Toscano, Hector Zenil, Jean-Paul Delahaye, and Nicolas Gauvrit. Calculating kolmogorov complexity from the output frequency distributions of small turing machines. *PloS one*, 9(5):e96223, 2014.

[51] RJ Solomonoff. A preliminary report on a general theory of inductive inference. *Report ZTB-135, Zator Co, Cambridge, MA*, 1960.

[52] Brian Spooner et al. Weavers and dealers: the authenticity of an oriental carpet. *The social life of things: Commodities in cultural perspective*, 1:95–235, 1986.

[53] Eric Talevich, Brandon M Invergo, Peter JA Cock, and Brad A Chapman. Bio. phylo: a unified toolkit for processing, analyzing and visualizing phylogenetic trees in biopython. *BMC bioinformatics*, 13(1):209, 2012.

[54] Jamshid Tehrani and Mark Collard. Investigating cultural evolution through biological phylogenetic analyses of turkmen textiles. *Journal of Anthropological Archaeology*, 21(4):443–463, 2002.

[55] Ilya Tëmkin and Niles Eldredge. Phylogenetics and material cultural evolution. *Current anthropology*, 48(1):146–154, 2007.

[56] Sebastiaan A Terwijn, Leen Torenvliet, and Paul Vitanyi. Normalized information distance is not semicomputable. *arXiv preprint arXiv:1006.3275*, 2010.

[57] Betty Alexandra Toole. Ada byron, lady lovelace, an analyst and metaphysician. *IEEE Annals of the History of Computing*, 18(3):4–12, 1996.

[58] SM Turaeva, EK Ginter, AA Revazov, RF Garkavtseva, and EI Sotnikova. Medical genetic study of the population of turkmenia. vi. intrapopulation variability from an analysis of marriage migrations and abo and hp marker systems. *Genetika*, 21(6):1039, 1985.

[59] Pere-Pau Vázquez and Jordi Marco. Using normalized compression distance for image similarity measurement: an experimental study. *The Visual Computer*, 28(11):1063–1084, 2012.

[60] Eric W. Weisstein. “langton’s ant." from mathworld–a wolfram web resource. `http://mathworld.wolfram.com/LangtonsAnt.html`. Accessed: 2018-05-24.

[61] William Wood. Turkmen ethnohistory. *Vanishing Jewels: Central Asia Tribal Weavings. Rochester Museum and Science Center, Rochester*, pages 27–41, 1973.

[62] Hector Zenil, Fernando Soler-Toscano, Jean-Paul Delahaye, and Nicolas Gauvrit. Two-dimensional kolmogorov complexity and an empirical validation of the coding theorem method by compressibility. *PeerJ Computer Science*, 1:e23, 2015.

[63] Hector Zenil, Fernando Soler-Toscano, Narsis A Kiani, Santiago Hernández-Orozco, and Antonio Rueda-Toicen. A decomposition method for global evaluation of Shannon entropy and local estimations of algorithmic complexity. *arXiv preprint arXiv:1609.00110*, 2016.

[64] Hector Zenil, Narsis A Kiani, Francesco Marabita, Yue Deng, Szabolcs Elias, Angelika Schmidt, Gordon Ball, and Jesper Tegner. An algorithmic information calculus for causal discovery and reprogramming systems. *arXiv preprint arXiv:1709.05429*, 2017.