Towards The Use of Aesthetics in Decision Making: Kolmogorov Complexity Formalizes Birkhoff's Idea

Misha Koshelev[1], Vladik Kreinovich[1], and Yeung Yam[3]

[1]Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, email {mkosh,vladik}@utep.edu
[2]Department of Mechanical & Automation Engineering
The Chinese University of Hong Kong, Shatin, NT, Hong Kong, China
email yyam@mae.cuhk.edu.hk

Abstract

Decision making is traditionally based on utilitarian criteria such as cost, efficiency, time, etc. These criteria are reasonably easy to formalize; hence, for such criteria, we can select the best decision by solving the corresponding well-defined optimization problem.

In many engineering projects, however, e.g., in designing cars, building, airplanes, etc., an important additional criterion which needs to be satisfied is that the designed object should be good looking. This additional criterion is difficult to formalize and, because of that, it is rarely taken into consideration in formal decision making.

In the 1930s, the famous mathematician G. D. Birkhoff has proposed a formula that described beauty in terms of "order" and "complexity". In the simplest cases, he formalized these notions and showed that his formula is indeed working. However, since there was no general notion of complexity, he was unable to formalize his idea in the general case. In this paper, we show that the existing computer-based general notion of object complexity (called *Kolmogorov complexity*) leads to a reasonable formalization of Birkhoff's idea and thus, leads to a possibility to take aesthetic criteria into consideration in decision making.

1 Introduction

Traditional formalized engineering decision making does not take beauty into consideration. Decision making is traditionally based on utilitarian criteria such as cost, efficiency, time, etc. These criteria are reasonably easy to formalize; hence, for such criteria, we can select the best decision by solving the corresponding well-defined optimization problem.

In many engineering projects, however, e.g., in designing cars, building, airplanes, etc., an important aditional criterion which needs to be satisfied is that the designed object should be beautiful (or at least good looking). This additional criterion is difficult to formalize and, because of that, it is rarely taken into consideration in formal decision making.

How can we formalize beauty? Birkhoff's formula. In the 1930s, G. D. Birkhoff, one of the world leading mathematicians, has proposed a formula that described beauty in terms of "order" O and "complexity" C [2, 3, 4, 5, 6] (see also [8]). Namely, according to his formula, the beauty B of an object is equal to

$$B = \frac{O}{C}. \tag{1}$$

How can we describe "order" and "complexity"?

In the simplest cases, Birkhoff formalized these notions and showed that his formula is indeed working. Namely, he showed that the beauty of simple geometric patterns, of simple melodies, and even of simple verses can be well described by his formula. However, since there was no general notion of complexity, he was unable to formalize his idea in the general case. This is what we are planning to do in this paper.

2 Our main idea

Towards formalization of Birkhoff's formula. In our formalization, we will use the general computer-based notion of object complexity, which is widely used in computer science. For example, we can define the *complexity* $C(x)$ of an object x as the length $l(p)$ of the shortest program p (in a certain language) which generates this object. This notion of object complexity was originally proposed by G. Chaitin, A. Kolmogorov, and R. Solomonoff, and it is usually called *Kolmogorov complexity* (see, e.g., [9, 21]). Alternatively, we can use a *modification* of this notion which takes into consideration not only the *length* $l(p)$ of the program p (i.e., the number of bits in its computer description), but also the *time* $t(p)$ that the program p takes to generate the desired object x.

In order to choose an appropriate formalization, let us start with an informal discussion of Birkhoff's ideas.

Informal motivations: the ideas behind Birkhoff's notions. In Birkhoff's description, *complexity* of an object looks like time which is necessary to generate this object. For example, he defines the complexity of a polygon as the number of its vertices, etc. Intuitively, it is clear that beauty must be reasonably simple, so, all other characteristics being equal, the more over-complicated the object is, the less beautiful it is.

Similarly, Birkhoff's *order* looks like a simplicity of the description: if we can describe an object by using a shorter text, then its order is higher. If the only way of describing an object is to enumerate all its pixels (all its nodes for a melody, all its vertices for a polygon, etc.), then this object does not have much order in it. Intuitively, it seems reasonable that an object with some order in it should (all else being equal) look prettier than an object with less order. How can we formalize this notion of "order"?

By a *description*, we mean a *complete* description, i.e., a description which is detailed enough so that, given this description, we can uniquely reconstruct the object. In other words, the description must serve as a *program* for a computational device which, given this description, reconstructs the object. In these terms, the length of the description is the length $l(p)$ of this program p. So, the smaller $l(p)$, the more order is there in the object.

Summarizing our discussion of complexity and order, we can conclude that the beauty $B(x)$ of an object x depends on the *time* $t(p)$ of the program p which generates x, and on the *length* $l(p)$ of this program, i.e., that $B(x) = f(t(p), l(p))$ for some function $f(t, l)$. The only thing we know about the function $f(t, l)$ is that it should monotonically decrease with the increase of each of the variables t and l.

In these term, the question of formalizing beauty can be reformulated in more mathematically-sounding terms: Which function $f(t, l)$ should we choose?

Which function $f(t, l)$ should we choose? It is well known in computer science that there is a *trade-off* between the program time and the program length. A short program usually uses only a few ideas of pseeding up computations, and thus, takes a reasonable amount of time to run. If we want to speed up the computations, we must add some complicated ideas and modify the algorithm. As a result, to make the program faster, we must usually make it longer. Vice versa, we can often shorten the program by eliminating some of the time-saving parts and thus, by making its running time longer.

This trade-off is not only true for programs written in the same programming language, the same trade-off is true if we compare programs written on programming languages of different level. For example, we can write a program in machine code (or in assembler language, which is close to the machine code).

- In a machine-code program, we have to spell out all necessary steps, so this program will be reasonably long. On the other hand, in a machine code program, every instruction will be immediately implemented, so running this program does not take too long.
- Alternatively, we can write our program in a high level programming language (e.g., in C++). In this case, the program is usually shorter, because we do not need to spell out all the details, it is sufficient to describe the construction that we want to use (like a loop or calling a function). However, when we run this short program, we first need to translate it into the machine code (i.e., *compile* it), and this compiling takes extra time.

Thus, we can get a shorter program which runs longer, or we can have a longer program which runs faster.

Our definition of the formalized beauty depended on the program p. It is reasonable to require that the "beauty" of an object x should not depend on which level we write this program p. Let us formalize this requirement.

By going to a different level of programming, we can cut a lot of bits from the length of the program. Let us describe this cut step-by-step and analyze what happens if we cut exactly one bit.

We are interested not in the abstract notion of beauty, but in the much more specific notion of the beauty of the engineering designs. When we talk about an engineering design, we mean that we have specifications which are usually relatively easy to check, and we want to find a design which meets these specifications. Checking is relatively easy, the most difficult part is finding the design.

If we cut a bit from the program that generates the design x, we get a new program p' which is exactly one bit shorter ($l(p') = l(p) - 1$). To generate the desired design x, since we do not know whether the deleted bit was 0 or 1, we can try both possible values of this bit (i.e., run two programs $p'0$ and $p'1$) and find out which of the two designs meets the original specifications. Thus, if we delete a bit, then instead of running the original program p once, we run *two* programs $p'0$ and $p'1$. Hence, crudely speaking, when we decrease the length of the program by 1, we thus get a double increase in the running time: $t(p') = 2t(p)$.

From this viewpoint, the fact that the beauty should not depend on the level means, in particular, that the values of $B(x)$ computed as $f(t(p), l(p))$ should stay the same if we replace the original program p by a one-bit-shorter program p'. In other words, we should have $f(t(p'), l(p')) = f(t(p), l(p))$. Since we know that $l(p') = l(p) - 1$ and $t(p') = 2t(p)$, we thus conclude that $f(2t(p), l(p) - 1) = f(t(p), l(p))$ for every program p. In other words, the desired function $f(t, l)$ must satisfy, for every two integers t and l, the following equation:

$$f(2t, l-1) = f(t, l). \qquad (2)$$

Functions which satisfy this equation can be explicitly described:

3 Main result and discussions

Definition. *We say that a function $f(t, l)$ is invariant if it satisfies the equation (2) for all positive integers t and l.*

Proposition. *A function $f(t, l)$ is invariant if and only if $f(t, l) = F(t \cdot 2^l)$ for some function $F(z)$ of one variable.*

Comment. For readers' convenience, the proof is given in the Appendix.

Our result justifies Birkhoff's formula. Let us show that this result justifies Birkhoff's formula (1). Namely, we will show that the search for the "most beautiful" design is equivalent to looking for a design for which the ratio (1) takes the largest possible value for appropriately defined quantities O and C.

Indeed, since we assumed that the function $f(t, l)$ is monotonically decreasing in both variables, we can conclude that the function $F(z)$ is monotonically decreasing too. So, looking for the "most beautiful" design means looking for the design generated by a program p for which the product $t(p) \cdot 2^{l(p)}$ takes the smallest possible value, or, equivalently, for which the inverse value $2^{-l(p)}/t(p)$ takes the largest possible value. We have already mentioned that the running time $t(p)$ is a natural formalization of Birkhoff's complexity C, and that Birkhoff's "order" O is a monotonically decreasing function of the program length $l(p)$. Thus, *looking for the most beautiful design means looking for a design for which the ratio O/C takes the largest possible value, where $C = t(p)$ and $O = 2^{-l(p)}$*. So, we indeed get a justification for Birkhoff's formula.

Our result makes perfect sense from the pragmatic viewpoint. In the above formalization of the notion of the "most beautiful" design, we were using two things at the same time: the design x and the program p which generates this design. Let us separate the design from the program. Namely, for each possible design x, we can define its "beauty" $a(x)$ as the smallest possible value of the product $t(p) \cdot 2^{l(p)}$ for all possible programs p which generate this design. Then, finding the most beautiful design means finding the design x which satisfies all the requirements and for which thus defined quantity $a(x)$ takes the smallest possible value.

This notion $a(x)$ is known in the theory of Kolmogorov complexity: namely, it was introduced by Leonid A. Levin in [20] as one of the possible modifications of Kolmogorov complexity which takes into consideration not only the length $l(p)$ of the program, but its running time $t(p)$ as well. Levin has proven that if we are looking for an optimal (asymptotically fastest) universal algorithm for solving different search problems (like the problems of design; see, e.g., [15]), then this optimal algorithm should check all possible designs in the increasing order of their Levin's complexity $a(x)$ (see [1, 20, 21]).

Thus, our formalization of beauty makes perfect pragmatic sense: if we want to find the best design as fast as possible, we must first look among the prettiest designs (i.e., among the designs with the smallest possible value of $a(x)$), then among the next prettiest designs, etc.

4 Future work: we need a practically working tool

This theoretical idea is not yet a practically working tool.

- *Theoretically*, Birkhoff's idea seems to work well.
- However, *in practice*, there is a big obstacle to applying this idea, because Kolmogorov complexity is not algorithmically computable (see, e.g., [9, 21]).

  > Levin's modification of Kolmogorov complexity is actually computable but computing it requires too long time, so for all practical purposes, it is not computable at all.

What can we do to make this criterion practically useful?

How to transform this theoretical idea into a practically working tool? First idea: let's use wavelets. The *first* approach towards making this notion practical is to take into consideration the fact that the Kolmogorov complexity is not computable because it is based on considering *all* possible algorithms. If we limit the class of algorithms, we get a computable version of Kolmogorov complexity. This idea was used, e.g., in [16], where a similarly modified version of Kolmogorov complexity was used to successfully predict the time required for a human to remember a geometric pattern. How can we come up with reasonable computable analogues of complexity and order (symmetry)?

Complexity of a computer object (string, image, etc.) can be measured by the ability of compressing programs to compress them. Thus, to get a computable estimate for complexity, we can use an advanced compression algorithm (e.g., an algorithm that underlies the widely used zip compression), and measure the complexity by the length of the compressed object:

- if the compressed text is short, the object was easy;
- if the compressed text still takes many bits, the compressed object was complex.

To measure the *order* (= *symmetry*, see, e.g., [29]) of an object, we can, similarly, use compression procedures, but this time, only procedures which use symmetry to compress. The most widely known symmetry-motivated compression techniques is the *wavelet* compression (see, e.g., [10, 11, 12, 13, 14, 22, 23, 24, 25, 27, 28]). In view of this, we use the length of the wavelet compression as an indication of the order:

- an image with a short wavelet compression has high order, while
- an image whose wavelet compression has approximately the same length as the original image has low order.

We are planning to experimentally check that these definitions indeed lead to a reasonable characterization of beauty.

How to transform this theoretical idea into a practically working tool? Second idea: let's use fuzzy logic. The *second* approach towards making the notion of beauty practically useful is to take into consideration that we humans have a good intuitive understanding of beauty, and thus, even when we cannot give a precise description of what exactly is beautiful and what is not, we can give a reasonable good description of beauty in terms of words from the natural language.

Thus, in order to formalize the notion of beauty, it is reasonable to use a methodology which successfully formalizes such statements – namely, the methodology of fuzzy logic (see, e.g., [17, 26]). Our preliminary analysis shows that we can indeed get a good description of beauty in this manner. Namely, we have shown that simple rules from natural language explain why golden proportion looks aesthetically pleasing [18, 19].

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCCW-0089, by NSF grant No. DUE-9750858, and by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518.

The authors are thanful to Leonid A. Levin, Luc Longpré, and Ronald R. Yager for valuable comments.

References

[1] M. Beltran, G. Castillo, and V. Kreinovich, "Algorithms That Still Produce a Solution (Maybe Not Optimal) Even When Interrupted: Shary's Idea Justified", *Reliable Computing*, 1998, Vol. 4, No. 1, pp. 39–53.

[2] G. D. Birkhoff, "A mathematical approach to aesthetics", *Scietia*, September 1931, Vol. 50, pp. 133–146 (reprinted in [7], Vol. 3, pp. 320–333).

[3] G. D. Birkhoff, "A mathematical theory of aesthetics", *Rice Institute Pamphlet*, July 1932, Vol. 19, pp. 189–342 (reprinted in [7], Vol. 3, pp. 382–535).

[4] G. D. Birkhoff, *Aesthetic measure*, Harvard University Press, Cambridge, MA, 1933.

[5] G. D. Birkhoff, "Three public lectures on scientific subjects", *Rice Institute Pamphlet*, January 1941, Vol. 28, pp. 1–76 (reprinted in [7], Vol. 3, pp. 755–777).

[6] G. D. Birkhoff, "Mathematics of aesthetics", In: J. R. Newman (ed.), *The World of Mathematics*, Simon and Schuster, N.Y., 1956, Vol. 4, pp. 2185–2208.

[7] G. B. Birkhoff, *Collected Mathematical Papers*, Dover, N.Y., 1960.

[8] G. Birkhoff, "Mathematics and psychology", *SIAM Review*, 1969, Vol. 11, No. 4, pp. 429–467.

[9] C. Calude, *Information and randomness: An algorithmic perspective*, Springer-Verlag, Berlin, 1994.

[10] G. Carrillo, S. D. Cabrera, and A. A. Portillo, "Inspection of Surface-Mount-Device images using wavelet processing", *Proceedings of SPIE Applied Imagery Pattern Recognition*, Washington D.C., October 1996.

[11] C. Chui, *An Introduction to Wavelets*, Academic Press, 1992.

[12] R. R. Coifman, G. Matviyenko, and Y. Meyer, 'Modulated Malvar-Wilson bases", *Applied and Computational Harmonic Analysis*, 1997, Vol. 4, pp. 58–61.

[13] J. M. Gallegos, J. R. Villalobos, G. Carrillo, and S. D. Cabrera, "Sequential algorithms for the inspection of Surface Mounted Devices", *Proceedings of the 13th SPIE Conference on Intelligent Robots and Computer Vision, Boston, MA*, SPIE, 1994, Vol. 2354, pp. 215–226.

[14] J. M. Gallegos, J. R. Villalobos, G. Carrillo, and S. D. Cabrera, "Reduced-dimension and wavelet processing of SMD images for real-time inspection", *Proceedings of the IEEE Southwest Symposium on Images Analysis and Interpretation*, San Antonio, April 1996.

[15] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[16] R. M. Granovskaya, I. Ya. Bereznaya, and A. N. Grigorieva, *Perception of form and forms of perception*, Erlbaum, Hillsdale, NJ, 1987.

[17] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications,* Prentice Hall, Upper Saddle River, NJ, 1995.

[18] M. Koshelev, "Fuzzy Logic Explains the Golden Proportion," *BUlletin for Studies and Exchanges on Fuzziness and its AppLications (BUSEFAL)*, 1996, No. 67, pp. 14–17.

[19] M. Koshelev, "Fuzzy Logic Explains the Golden Proportion," *International Journal of Intelligent Systems*, 1997, Vol. 12, pp. 415–417.

[20] L. A. Levin, "Universal sequential search problems", *Problems of Information Transmission*, 1973, Vol. 9, No. 3, pp. 265–266.

[21] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*, Springer-Verlag, N.Y., 1997.

[22] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, Vol. 14, No. 7, pp. 710–732.

[23] S. Mallat and W. L. Hwang, "Singularity detection and processing with wavelets", *IEEE Transactions on Information Theory*, 1992, Vol. 38, No. 2, pp. 617–643.

[24] S. Mallat and W. L. Hwang, "Characterization of signals from multiscale edges", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992, Vol. 14, No. 7, pp. 710–732.

[25] Y. Meyer, *Wavelets algorithms and applications*, SIAM, Philadelphia, 1993.

[26] H. T. Nguyen and E. A. Walker, *A first course in fuzzy logic,* CRC Press, Boca Raton, Florida, 1997.

[27] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA, 1996.

[28] M. Vetterli and J. Kovacevic, *Wavelets and subband coding*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[29] H. Weyl, "Symmetry", In: J. R. Newman (ed.), *The World of Mathematics*, Simon and Schuster, N.Y., 1956, Vol. 1, pp. 671–724.

Allendux: Proof of the Proposition

From the equation (2), we conclude that $f(t,l) = f(2t,l-1)$. Applying the same equation (2) to the right-hand side of the new equality, we conclude that $f(2t,l-1) = f(2^2t,l-2)$, and thus, that $f(t,l) = f(2^2t,l-2)$. Similarly, we can prove that

$$f(t,l) = f(2^2t,l-2) = f(2^3t,l-3) = \ldots = f(2^kt,l-k)$$

for an arbitrary k. In particular, for $k=l$, we conclude that $f(k,l) = f(2^lt,0)$. Thus, the Proposition is true, for the function $F(z) = f(z,0)$.