45th Southeastern Symposium on System Theory
Baylor University, Waco, TX, USA, March 11, 2013

On the Improbability of Algorithmic Specified Complexity

Winston Ewert & Robert J. Marks II
Dept. of Electrical & Computer Engineering
Baylor University
Waco, Texas

William A. Dembski
Discovery Institute
Seattle, WA

***Abstract*—An event with low probability is unlikely to happen, but events with low probability happen all of the time. This is because many distinct low probability events can have a large combined probability. However, some low probability events can be seen to follow an independent pattern. *Algorithmic specified complexity* (ASC) measures the degree to which an event is improbable and follows a pattern. We show a bound on the probability of obtaining a particular value of algorithmic specified complexity. Consequently we can say that high ASC objects are improbable.**

***Index Terms*—Keywords: specified complexity, algorithmic information theory, Kolmogorov complexity**

I. Introduction

Low probability events are often claimed to not happen. But this is fallacious because low probability events take place all of the time. Any snowflake's pattern is highly improbable, but this does not prevent low probability snowflakes from existing. The common occurrence of low probability events seems paradoxical within the rubric of the probability paradigm. The paradox is resolved after recognizing there are often very many improbable events such that the total probability of such low probability events can actually be quite large.

To illustrate, assume that there exists 10^{1000} possible snowflakes each of which is equally likely. This means that any given snowflake pattern has a probability of 10^{-1000}. Thus

$$\Pr[\Pr[X] \leq 10^{-1000}] = 1. \tag{1}$$

where X is a random variable corresponding to a particular snowflake pattern. The equation states that we have a high probability (actually a certainty) of obtaining a very low probability event.

However, it is commonly assumed that no two snowflakes are alike. This is because such an event has dramatically lower probability than the occurrence of of a single specified snowflake. The probability of a second specified snowflake being identical to the first specified snowflake, however, is

$$\Pr[\text{First snowflake } = x, \text{ and the second snowflake } = x]$$
$$= 10^{-1000}10^{-1000} = 10^{-2000}.$$

But this is the same as the second snowflake having some other specification.

$$\Pr[\text{First snowflake } = x, \text{ and the second snowflake } = y]$$
$$= 10^{-2000}$$

where $x \neq y$. We revisit this example is Section III-B.

Another example is the specified arrangement of sand on a beach. Any one particular arrangement of the sand is highly improbable. But an arrangement of the sand to spell words is not less probable then an arrangement without words. We would say, however, that the forming of such words through wind and water would be next to impossible.

How, then, do we resolve the seeming paradox that improbable events happen frequently? One resolution of the paradox is through the viewpoint of *specified complexity* [4]. An object with specified complexity is, as the name states, both specified and complex. For an object to be complex means that it is improbable. Specification means the object exhibits some independent pattern. The identical snowflakes exhibit a particular pattern: one snowflake is an exact replica of the other. This is what sets the pair off as distinct from all other pairs of snowflakes. Words written in the sand also exhibit a pattern: they form English letters.

Improbability gives us a good way to quantify the complexity of an object, but methods of measuring specification are less obvious. One method uses Kolmogorov complexity [1, 8, 7]. Kolmogorov complexity is defined to be shortest computer program length required to reproduce a specified bitstring description of an object. For identical snowflakes, the first snowflake can be described in detail followed by the computer command DUPLICATE. In contrast, describing two distinct snowflakes would require a longer program because each snowflake would have to be described separately. The program would be about twice as long as the program for the identical snowflakes. The identical snowflakes require a shorter program to describe them because they follow more of a pattern. Short programs and hence smaller Kolmogorov complexity corresponds to objects which follow a pattern.

Kolmogorov complexity suffers from the property of being unknowable [2]. There is no method to compute the Kolmogorov complexity of an object with arbitrary length. However, we can give upper bounds for the Kolmogorov complexity. If a given bitstream of 1000 bits can be compressed without loss to 200 bits, we are assured that the Kolmogorov complexity of the 1000 bits on the operating system equals or exceeds 200 bits. Consequently we can show from this bound that there is a pattern to the input, but we cannot determine whether there is a pattern we are missing. Additionally, Kolmogorov complexity quantities contain an unknown additive constant that allows it to be applicable to any computer language. The constant can be thought of as the length in bits of one computer language translating into another. As a result of any modeling using Kolmogorov complexity, the quantity is a useful theoretical construct [3].

Using conditional Kolmogorov complexity [6] we define *algorithmic specified complexity* (ASC) [5] as

$$ASC(X, C, P) = -\log_2 P(X) - K(X|C)$$

where

- X is the object or event or under consideration
- C is the context– the presumed information which can be used to describe the object
- $K(X|C)$ is the Kolmogorov complexity of X given C. This quantity can not be computed exactly but can be bounded.
- $P(X)$ is the probability of the occurrence of X.

By taking into account both the probability and the Kolmogorov complexity of an object, the ASC measures the degree to which an event fits the presumed probability distribution. The $\log_2 P(x)$ term measures the complexity of the object, whereas $-K(X|C)$ measures the specification. If an event happens which has a high ASC, we should conclude that since it has a low probability and the rare property of compressibility, it gives us strong indication to believe that the assumed probability distribution is incorrect.

The usefulness of this definition depends on the wide variety of constructs that are compressible. This includes for example simple pattern, such as "01" repeated 32 times. It also includes valid English text, which given a knowledge of the English language can be compressed. Its also include complex functioning systems because they can be described by their functionality rather then the system that produces that functionality. Thus Kolmogorov complexity captures a wide variety of objects that we deem "special." Thus we can usefully apply this metric to a wide variety of objects.

II. A Bound on the Probability of ASC

The following theorem quantifies the unlikelihood of obtaining a high ASC event.

Theorem 1. *The probability of obtaining an object exhibiting α bits of ASC is less then or equal to $2^{-\alpha}$.*

$$\Pr[ASC(X, C, P) \geq \alpha] \leq 2^{-\alpha} \qquad (2)$$

Proof:

$$\Pr[ASC(X, C, P) \geq \alpha]$$

$$= \Pr[-\log_2 P(X) - K(X|C) \geq \alpha]$$

$$= \Pr[P(X) \leq 2^{-\alpha - K(X|C)}]$$

Let β be the set of all events in the domain of X such that $P(X) \leq 2^{-\alpha - K(X|C)}$.

$$\Pr[ASC(X, C, P) \geq \alpha] = \sum_{x \in \beta} P(x).$$

The definition of β is such that we have an upper bound on $P(x)$. Thus

$$\begin{aligned}\Pr[ASC(X, C, P) \geq \alpha] &\leq \textstyle\sum_{x \in \beta} 2^{-\alpha - K(x|C)} \\ &= 2^{-\alpha} \textstyle\sum_{x \in \beta} 2^{-K(x|C)}.\end{aligned}$$

Since Kolmogorov complexity can assume prefix free code [3], a distribution over all programs is defined by

$$\Pr[X = x] = 2^{-K(x|C)}.$$

$\sum_{x \in \beta} 2^{-K(x|C)}$ is a summation over this distribution for some subset of the values, thus it less then or equal to one as dictated by the Kraft inequality [3].

$$\Pr[ASC(X, C, P) \geq \alpha] \leq 2^{-\alpha} \qquad (3)$$

This proves the theorem. ■

From the main result of Theorem 1 in (2),

$$-\log_2 \Pr[ASC(X, C, P) \geq \alpha] \geq \alpha$$

It is therefore unlikely to obtain a high value of ASC. Low probability events commonly occur, but high ASC events do not.

III. Examples

The definition of ASC uses both complexity and specification. We can look at various cases of these parameters to see how ASC is affected.

A. Uniform Specification and Complexity

1) Compressible Sequences: Suppose that we have 256 items each with equal probability of occurring and each of the same compressed length. The only way for 256 items to all have the same minimum length is to use 8 bit codes for all of them. For any item in the collection we then have

$$\begin{aligned}ASC(X, C, P) &= -\log_2 P(X) - K(X|C) \\ &= -\log_2 \frac{1}{256} - 8 \\ &= 8 - 8 \\ &= 0 \text{ bits of ASC}\end{aligned}$$

And the bound on 0 bits of ASC is

$$\Pr[ASC(X, C, P) > 0] \leq 2^{-0} = 1$$

The probability in this case is clearly 1 because all objects will have the same ASC.

2) A Rare Compressible Sequence: Suppose we have a single sequence that can be compressed into 2 bits but has a probability of 2^{-256}. Then we calculate the ASC

$$\begin{aligned} ASC(X,C,P) &= -\log_2 P(X) - K(X|C) \\ &= -\log_2 2^{-256} - 2 \\ &= 256 - 2 \\ &= 254 \text{ bits of ASC.} \end{aligned}$$

The bound on this is

$$\Pr[ASC(X,C,P) > 254] \leq 2^{-254}$$

which is 4 times as probable as the actual value because there can be only up to 4 bit sequences of 2 bits in length.

3) A Common Compressible Sequence: Suppose that we have a single sequence that can be compressed into 2 bits and this happens half of the time. We calculate the ASC

$$\begin{aligned} ASC(X,C,P) &= -\log_2 P(X) - K(X|C) \\ &= -\log_2 \frac{1}{2} - 1 \\ &= 1 - 2 \\ &= -1 \text{ bits of ASC.} \end{aligned}$$

This gives the bound

$$\Pr[ASC(X,C,P) > -1] \leq 2^{1} = 2.$$

While the sequence is highly compressible, the high probability prevents it from having a large measure of ASC.

B. Snowflakes

Consider again the case of the snowflakes. There are, by our assumption, 10^{1000} possible snowflakes. We'll assume that we can describe each snowflake using a compact bit pattern taking $\log_2 10^{1000} = 1000 \log_2 10 \approx 3322$ bits. If we have to describe two distinct snowflakes, it will take

$$\begin{aligned} K(X|C) &= 3322 \text{ bits} + 3322 \text{ bits} + c \qquad (4) \\ &= 6644 \text{ bits} + c \end{aligned}$$

where c is some constant number of bits. The log-probability is

$$\begin{aligned} -\log_2 P(X) &= -\log_2 10^{-2000} = 2000 \log_2 10 \qquad (5) \\ &= 6644 \text{ bits.} \end{aligned}$$

So the ASC is

$$\begin{aligned} -\log_2 P(X) - K(X|C) &= 6644 \text{ bits} - 6644 \text{ bits} - c \qquad (6) \\ &= -c. \end{aligned}$$

Using Theorem 1 we obtain a bound of 2^c which, since $c > 0$, produces a bound above 1 for the probability of obtaining this pair of snowflakes. There is nothing unusual about an arbitrary pair of snowflakes.

However, if the two snowflakes are identical, we can describe them by a shorter program.

$$K(X|C) = 3322 \text{ bits} + c.$$

where c is some constant number of bits. The probability is the same, so the ASC is

$$\begin{aligned} -\log P(X) - K(X|C) &= 6644 \text{ bits} - 3322 \text{ bits} - c \\ &= 3322 \text{ bits} - c. \end{aligned}$$

Using the theorem, we bound the probability at $2^{-3322+c}$. Assuming that the constant c is sufficiently small, this is a vanishingly small probability and thus we can conclude that obtaining two identical snowflakes would be absurdly improbable. If we did find two identical snowflakes, we would have to conclude that our original assumed probability distribution was incorrect.

IV. Conclusion

The algorithmic specified complexity is a theoretical quantification measuring how well a probability distribution explains a given event. By using the bound in Theorem 1, we can establish the probability of obtaining particular amounts of ASC. We conclude that an object exhibiting high ASC is unlikely to arise. Given a high ASC object, we have evidence that the assumed probability distribution was incorrect.

Additional examples of ASC are available [5]. We are currently exploring the capabilities and limitations of the ASC measure.

References

[1] Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM (JACM)*, 13, 1966.

[2] Gregory J. Chaitin. *The Unknowable*. Springer, New York, New York, USA, 1999.

[3] Thomas M Cover and Joy A Thomas. *Elements Of Information Theory*. Wiley-Interscience, Hoboken, NJ, second edi edition, 2006.

[4] William A. Dembski. *The Design Inference: Eliminating Chance through Small Probabilities*, volume 112. Cambridge University Press, 1998.

[5] Winston Ewert, William A. Dembski, and Robert J. Marks II. Algorithmic Specified Complexity. In *Engineering and Metaphysics*, Tulsa, OK, 2012.

[6] AN Kolmogorov. Logical basis for information theory and probability theory. *Information Theory, IEEE Transactions on*, 14(5):662–664, September 1968.

[7] AN Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 1968.

[8] RJ Solomonoff. A preliminary report on a general theory of inductive inference. Technical report, Zator Co. and Air Force Office of Scientific Research, Cambridge, Mass, 1960.