Application of Kolmogorov Complexity to Image Compression: It Is Possible to Have a Better Compression, But It Is Not Possible to Have the Best One

Sanjeev Subbaramu[1], Ann Q. Gates[1,2], and Vladik Kreinovich[1,2]

[1]Department of Computer Science and
[2]NASA Pan-American Center for
Earth and Environmental Studies,
University of Texas at El Paso, El Paso, TX 79968
emails {sanjeev,agates,vladik}@cs.utep.edu

1 The Problem of Image Compression

Image processing is important but difficult. An important part of data processing is processing images. One of the main problems with storing and processing images is that an image contains a large amount of data. A simple image on a PC contains about 1 Megabyte of data, and a good photo contain thousand times more: about 1 Gigabyte (10^9 bytes) of data. These problems are especially acute for Geographic Information Systems (GIS), where hundreds and thousands of images are stored.

Some image processing algorithms are reasonably simple (e.g., detecting a certain pattern in weather analysis). These algorithms usually do not require large amounts of computations.

However, there are many other algorithms, especially related to geophysical and environmental applications, where the (directly observable) data supplied by the image serves as an input to a system of partial differential equations, which are used to determine the desired quantities.

Image compression is needed. The data amount makes it very difficult to store, transmit, or process an image pixel-by-pixel. Instead, different *image compression* schemes are used that compress the data into a usually shorter file.

There exist different image compression schemes; most of these methods are based on some numerical methods applied to the original image: e.g., instead of storing the image, we expand the image (or its parts) into Fourier (or wavelet) series and store the coefficients. Among the most widely used, we can name gif, jpeg, PostScript, their zipped and compressed versions, etc.; for detailed description, see, e.g., [3]).

We want a good (ideally, the best) image compression scheme. On each image, some image compression schemes compress better, some compress worse. It is desirable to have as good a compression scheme as possible.

Ideally, we would like to have the *best* compression scheme, i.e., a scheme that compresses better than all the others. This ideal "best" compression scheme may not exit or may not be (easily) computable. In this case, we would like, at least, to find the best among the *existing* schemes.

In some cases, we know the best compression schemes, but they may not be the best in a general situation. For some specific problems, the best image compression scheme is known. For example, for a problem of detecting the Surface Mounted Devices on a photo of a manufactured computer chip, wavelet coefficients provide the best image compression (see, e.g., [2, 1]; for 1-D images, see [4]). In more precise terms, this image compression is optimal for black-and-white images with sharp edges.

For other types of images, other compression techniques turn out to be the best.

In GIS, we have images of very different types, that usually do not fit into one of the known classes; so we usually do not know which compression will work better.

What we are planning to do in this paper. In this paper, we describe the results of the (preliminary) theoretical analysis of the problem of choosing the best compression scheme. Our main results are: that there is no *best* compression scheme, but we can always find a *better* one.

The formalization of our problem will be shown to be closely related to the notion of *Kolmogorov complexity* (see detailed explanations below) that is very useful in theory of computing.

Future plans. We view our theoretical result as (reasonably) optimistic: that we can always find a better image compression scheme.

In the future work, we therefore plan to actually look for such better schemes.

2 Towards Formalization of the Problem

What is an image? In the computer, everything is represented as a sequence of 0's and 1's. In particular, an image is also a sequence of 0's and 1's.

The original image usually comes (e.g., from a satellite) pixel-by-pixel, as a sequence of 0's and 1's that describe the image's intensity at different pixels. An arbitrary sequence of 0's and 1's can, therefore, be interpreted as an image.

Therefore, in the following text, by an *image*, we will understand an arbitrary sequence of 0's and 1's, i.e., an arbitrary *binary word*.

What is a compression scheme? Towards a formal definition. A compression scheme means that we have an algorithm s that transforms an arbitrary image (binary word) w into a *compressed image* $i = s(w)$. We must also be able to *uncompress* this image, i.e., we must also have a second algorithm u that restores the compressed image (transforms $s(w)$ into w). In other words, we must have $u(s(w)) = w$ for all words w.

When is one compression scheme better than the other one? The main goal of compression is to compress, i.e., to reduce the storage space. Therefore, ideally, we would like to say that a compression s_1 is *better* that a compression s_2 if for all possible words w, the length $|s_1(w)|$ of the first compression does not exceed the length $|s_2(w)|$ of the second compression.

In real life, however, things are not that simple. An image file not only contains the data itself, it also contains the *header* that explains what this image describes. For example, for a satellite photo, the image file must contain the location of the satellite, the time when the photo was taken, etc.

Similarly, a compressed file must contain a similar information + the information about which exactly compression scheme was used.

If we have, say, only two compression schemes, then we need just 1 bit to select a scheme: 0 or 1. However, if we have designed a new (third) scheme, we have to assign to it a longer ID code, because both short codes (0 and 1) are already taken. Thus, even if the new compression scheme always leads to the same length of compressed code, the *file* with the compressed image will be at least 1 bit longer, because of the longer header.

Due to the possible difference in header length, the property that we want to formalize (that the first compression scheme is better) takes the following (slightly more complicated) form: for every word w, $|s_1(w)| \le |s_2(w)| + c$, where c is a difference in header lengths.

If we do not know the difference in header lengths, then we can say that a compression scheme s_1 is better that s_2 if there exists a constant c for which, for every word w, we have $|s_1(w)| \le |s_2(w)| + c$.

Now, we are ready for the formal definitions.

3 Definitions

Definition 1. *Let W denote the set of all binary words (i.e., the set of all finite sequence of 0 and 1). By a compression scheme, we mean a pair of algorithms (s, u) from binary words to binary words for which, for all w, $u(s(w)) = w$.*

Definition 2. *We say that a compression scheme (s, u) is better than the compression scheme (s', u') if there exists a constant c such that for all words w, we have $|s(w)| \le |s'(x)| + c$.*

Definition 3. *We say that a compression scheme (s, u) is the best if it is better than any possible compression scheme.*

Comment. The natural questions are:

- Can we construct the *best* compression scheme, i.e., the compression scheme that is better than *all possible* compression schemes?
- If the answer to the first question is "no", i.e., if there is no way to construct the compression scheme that is better than *all possible* compression schemes, then we have a second question: Can we at least get a compression scheme that is better than *all known* compression schemes?

To answer these questions, we will use the fact that similar definitions are already known in the theory of computing, under the name of Kolmogorov complexity (see, e.g., [5]).

4 Kolmogorov Complexity and its Relation to Our Problem

The notion of Kolmogorov complexity. The notion which is known as *Kolmogorov complexity* was defined independently by G. Chaitin, R. Solomonoff, and A. Kolmogorov. The Kolmogorov complexity $K(w)$ of a word w is defined as follows.

For every word w and for every algorithm u, we define the *complexity* of this word with respect to u as the shortest length of the word p for which $u(p) = x$, i.e., as $K_u(x) = \min\{|p| : u(p) = x\}$.

We then say that:

- an algorithm u is *better* than the algorithm u' if there exists a constant c for which, for all w, $K_u(w) \leq K_{u'}(w) + c$, and
- an algorithm u is *the best* if it is better than any other algorithm.

In Kolmogorov complexity theory, it is known that there exists the best algorithm u_{opt}. Complexity $K_{u_{\text{opt}}}(w)$ with respect to this best algorithm is called *Kolmogorov complexity.*

Similarity between our definitions and Kolmogorov complexity. Both in Kolmogorov complexity theory and in our problem, we are interested in the length of the "code" p from which we can uncompress the original word w.

In both problems, an algorithm is *better* if the corresponding "code" is shorter.

The main difference between our our definitions and Kolmogorov complexity.

- In Kolmogorov complexity theory, we were only interested in reconstructing the word w from its code p, i.e., only in *uncompressing* u. How to transform w into this short code p, was not of our concern.
  - We *did not require* that the "compression", i.e., transformation from x to the shortest code p, be algorithmic.
  - Moreover, for Kolmogorov complexity, we *cannot* require that this transformation be algorithmic, because it is known that Kolmogorov complexity is *not* algorithmically computable.
- In our case, we require that *both* the un-compression u and the *compression* s be algorithmic.

We will see that this difference changes the result: unlike Kolmogorov complexity, we no longer have the best scheme.

5 Our Results

Theorem 1.

- *For every finite list of compressions schemes* $(s_1, u_1), \ldots, (s_n, u_n)$*, there exists a compressions scheme* (s, u) *that is better than all of them.*
- *No compression scheme* (s, u) *is the best.*

Comment 1. For reader's convenience, all the proofs are placed in the last section.

Comment 2. Our proof is a modification of proofs known for Kolmogorov complexity. In particular, in proving the second part of our theorem, we were inspired by the proof that Kolmogorov complexity is not computable.

- From the viewpoint of a *specialist in theory of computing*, our proof is a simple result.
- However, to the best of our knowledge, our results have not been explicitly formulated for image compression. Therefore, we decided that it is important to publish this result, so that *researchers in image processing* (who are usually not specialists in theory of computing) will be aware of it.

Comment 3. We required only that compression and un-compression are computable, without limiting the computation time. Actually, if the algorithm takes too long, it makes not sense to use it. It is therefore reasonable to restrict ourselves, e.g., to algorithms that require quadratic, or linear time in terms of the input. (Similar modification is known for Kolmogorov complexity; it is called *resource-bounded* Kolmogorov complexity.)

Definition 4.

- *By a complexity function, we mean a strictly increasing function $f(n)$ from natural numbers to natural numbers.*
- *By a complexity class, we mean a pair (f_u, f_s) of complexity functions.*
- *We say that a compression scheme (s, u) belongs to the complexity class (f_s, f_u) if there exists a constant C such that:*
  - *for every word w, the algorithm $s(w)$ finished its work in time $\le C \cdot f_s(|w|)$ (producing a compressed image $p = s(w)$); and*
  - *for every compressed image p, the algorithm $u(p)$ finished its work in time $\le C \cdot f_u(|p|)$.*

Theorem 2. *Let (f_s, f_u) be a complexity class, and let $(s_1, u_1), \ldots, (s_n, u_n)$ be compressions schemes from this class. Then, there exist a compression scheme (s, u) from the same complexity class that is better than all of them.*

6 Proofs

6.1 Proving the first part of Theorem 1

What we need to prove. Let compression schemes $(s_1, u_1), \ldots, (s_n, u_n)$ be given. Let us construct a compression scheme (s, u) that is better than all of them.

Constructing a compression algorithm $s(w)$. Let us first construct $s(w)$. For an arbitrary word w, to define $s(w)$, we apply all n compression schemes s_i to w and choose the i for which the compressed image $s_i(w)$ is the shortest, i.e., for which

$$|s_i(w)| = \min(|s_1(w)|, \ldots, |s_n(w)|). \tag{1}$$

Then, to construct $s(w)$, we place i in the first $\lceil \log_2(n) \rceil$ bits, and then place $s_i(w)$. The result is $s(w)$.

Constructing the corresponding un-compression algorithm $u(p)$. Let us now define the corresponding un-compression $u(p)$. If we get a compressed image $p = s(w)$, then (by our construction of $s(w)$), the first $\lceil \log_2(n) \rceil$ bits of the word p contain the integer i – the number of the compression scheme that was actually used. So, to reconstruct w, we apply the corresponding uncompress function $u_i(q)$ to the remaining part $q = s_i(w)$ of the compressed image p.

Proving that the constructed compression scheme is indeed better. Let us now show that the new compression scheme (s, u) is indeed better than all n given ones (s_i, u_i). Without losing generality, let us prove that (s, u) is better than (s_1, u_1).

Indeed, by construction of the scheme (s, u), the word $s(w)$ is the result of concatenation of $\lceil \log_2(n) \rceil$ bits (that contain i) and the word $s_i(w)$, where i is defined by the formula (1). Hence,

$$|s(w)| = \lceil \log_2(n) \rceil + |s_i(w)|. \tag{2}$$

Since the length $|s_i(w)|$ is the smallest of the lengths $|s_1(w)|, \ldots, |s_n(w)|$, we have, $|s_i(w)| \le |s_1(w)|$. Adding $\lceil \log_2(n) \rceil$ to both sides, we conclude that $\lceil \log_2(n) \rceil + |s_i(w)| \le \lceil \log_2(n) \rceil + |s_1(w)|$. Due to (2), we now have $|s(w)| \le \lceil \log_2(n) \rceil + |s_1(w)|$, i.e., for $c = \lceil \log_2(n) \rceil$, we have $|s(w)| \le c + |s_1(w)|$ for all w. So, (s, u) is indeed better than (s_1, u_1).

Similarly, (s, u) is better than $(s_2, u_2), (s_3, u_3) \ldots, (s_n, u_n)$. The first part of the theorem is proven.

6.2 Proving the second part of Theorem 1

Re-formulating the statement that we need to prove. We want to prove that there is no best compression scheme. A compression scheme (s, u) is the best if it is better than any other compression scheme (s', u'). So, to prove that there is no best compression scheme, we must show that for every compression scheme (s, u), there exists a compression scheme (s', u') for which (s, u) is *not* better than (s', u').

By definition, the statement that (s, u) is better than (s', u') means that $\exists c_{>0}\, \forall w\, (|s(w)| \le |s'(w)| + c)$. Hence, the statement that (s, u) is *not* better than (s', u') means that for each c, there exists a word w_c for which $|s(w_c)| > |s'(w_c)| + c$, i.e., for which $|s'(w_c)| < |s(w_c)| - c$.

Let us design a compressing algorithm s' for which such words w_c exists for all c, i.e., for which there exists words $w_1, \ldots w_n$, for which $|s'(w_1)| < |s(w_1)| - 1, |s'(w_2)| < |s(w_2)| - 2, \ldots |s'(w_n)| < |s(w_n)| - n$.

Base: constructing w_1. Let us start with constructing w_1. By definition of a compression scheme, the mapping $s(w)$ maps different words into different ones. There are infinitely many words, so, no matter what length L we choose, we cannot map all the words x into words $s(w)$ of length $\le L$ (because there are only finite number of words of fixed length).

Hence, for every L, there exists a word w, for which $|s(w)| > L$.

In particular, there exists a word w_1 for which $|s(w_1)| > 2$.

Base: constructing $s'(w)$ for small w. Let us pick a word p_1 of length 1, and set $s'(w_1) = p_1$. Then $|s'(w_1)| = 1$ and hence $|s'(w_1)| \le 2 - 1 < |s(w_1)| - 1$.

If the word p_1 was not used as a compression result, i.e., if we had $s(w) \ne p_1$ for all w, then we can simply keep old values of $s(w)$ for all $w \ne w_1$, i.e., define $s'(w) \ne s(w)$ for all $x \ne w_1$.

Otherwise, if there is a word w_1' for which $s(w_1') = p_1$, then:

- we assign $s'(w_1') = s(w_1)$ (i.e., "swap" the values of $s(w_1)$ and $s(w_1')$), and
- for all other words of length
$$\le L_1 = \max(|w_1|, |w_1'|, |s(w_1)|, |s(w_1')|),$$
we take $s(w) = s'(w)$.

Induction: constructing w_{k+1}. Suppose now that $w_1, \ldots, w_k$ are already defined, and that a function $s'(w)$ is defined, for all words of length $\le L_k$ in such a way that
$$|s'(w_i)| < |s(w_i)| - i$$
for $i = 1, \ldots, k$. Let us define w_{k+1}.

Let M_k be the largest length of $|s(w)|$ for all words w of length $|w| \le L_k$

First, similarly to how we proved the existence of w_1, we can show that there exists a word w_{k+1} for which $|w_{k+1}| > M_k$ and $|s(w_{k+1})| > M_k + k + 1$.

Induction: constructing $s(w)$. Let us pick a word p_{k+1} of length $M_k + 1$, and set $s'(w_{k+1}) = p_{k+1}$. Then,
$$|s'(w_{k+1})| = M_k + 1 \le (M_k + k + 1) - k < |s(w_{k+1})| - k.$$

There may be a word w'_{k+1} for which $s(w'_{k+1}) = p_{k+1}$. We have chosen w_{k+1} in such a way that $|w_{k+1}| > M_k$, i.e., that the length $|w_{k+1}|$ of w_{k+1} is larger than the largest length of $s(w)$ for all words w of length $|w| \le L_k$. Hence, we cannot have $w'_{k+1} \le L_k$, so, $|w'_{k+1}| > L_k$.

Let us now define $s'(w'_{k+1}) = s(w_{k+1})$ (i.e., we "swap" the values of $s(w_{k+1})$ and $s'(w_{k+1})$).

Now, for all other words of length
$$\le L_{k+1} = \max(|w_{k+1}|, |w'_{k+1}|, |s(w_{k+1})|, |s(w'_{k+1})|)$$
we define $s(w) = s'(w)$.

By repeating this construction for all k, we get a desired compression algorithm s'.

Constructing the un-compression algorithm $u(p)$. It is easy to construct the corresponding un-compression u'.

Conclusion. The second part of Theorem 1 is thus proven too.

6.3 Proving Theorem 2

The proof of the first part of Theorem 1 actually proves Theorem 2 as well: Indeed, if all compression schemes (s_i, u_i) belong to the complexity class (f_s, f_u), then, e.g., each of the compression $s_i(w)$ requires time $\le C_i \cdot f_s(|w|)$ for some C_i.

Hence, computing $s(w)$ requires applying *all* of them, which takes time $\le C \cdot f_s(|w|)$, where $C = C_1 + \ldots + C_n$; this computation time is still in the same complexity class (f_s, f_u).

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCCW-0089, by NSF grants No. DUE-9750858 and EEC-9322370, by the Institute for Manufacturing and Materials Management (IM^3), 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 greatly thankful to Dr. Luc Longpré for his help.

References

[1] A. E. Brito and O. Kosheleva, "Interval + Image = Wavelet: For Image Processing under Interval Uncertainty, Wavelets are Optimal", *Reliable Computing*, 1998 (to appear).

[2] A. E. Brito, O. M. Kosheleva, and S. D. Cabrera, "Multi-Resolution Data Processing is Optimal: Case Study of Detecting Surface Mounted Devices", *Proceedings of the International Conference on Intelligent Systems and Semiotics (ISAS'97)*, National Institute of Standards and Technology Publ., Gaithersburg, MD, 1997, pp. 157–161.

[3] B. Fortner, *The Data Handbook, A guide to understand the organization and Visualization of Technical Data*, Springer-Verlag, N.Y., 1995.

[4] V. Kreinovich, O. Sirisaengtaksin, and S. Cabrera, "Wavelet neural networks are optimal approximators for functions of one variable." *Proceedings of the IEEE International Conference on Neural Networks*, Orlando, FL, July 1994, Vol. 1, pp. 299-303.

[5] M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, N.Y., 1997.