

# A New Universal Two Part Code for Estimation of String Kolmogorov Complexity and Algorithmic Minimum Sufficient Statistic

Scott Evans, Gary Saulnier\*, and Stephen F. Bush

{evans, bushsf}@research.ge.com; saulng@rpi.edu

*Abstract—A new universal compression algorithm is introduced that provides an estimate of an Algorithmic Minimum Sufficient Statistic for a given binary string. This algorithm computes and encodes an estimated Minimum Descriptive Length (MDL) partition of symbols in a string for compression. Using Symbol Compression Ratio (SCR) as a heuristic this algorithm produces a two-part code for a string where the codebook or model estimates the algorithmic minimum sufficient statistic.*

## I. INTRODUCTION

Practical application of the rich theory of Kolmogorov Complexity [4] requires computable estimators. Accessible estimators that are useful in various applications (see [7][3]) include the class of universal compression algorithms such as LZ78 [6] and its variants. While useful in estimating Kolmogorov Complexity, these codes are not designed to discriminate the model and data portions of the best two-part code or discern Algorithmic Minimum Sufficient Statistics [5].

We introduce a new algorithm for determining the optimal partition of a string in spirit of MDL using Symbol Compression Ratio (SCR) as a heuristic. Our new compression technique estimates Kolmogorov Complexity under MDL principles with a string modeled as the concatenation of a finite set of symbols. This technique not only provides a compression technique, but also results in a simple binary tree model of a string that can be used to generate a typical data set to which the string belongs. The binary tree model and associated probability distribution form an estimate of an Algorithmic Minimum Sufficient Statistic for the string.

## II. UNIVERSAL TWO PART CODES

Following [5], an MDL decomposition of a binary string  $x$  considering finite set models is given by:

$$K_{\varphi}(x) = \{K(S) + \log_2 |S|\},$$

Where  $S$  represents a finite set of which  $x$  is a typical element. The minimum possible sum of descriptive cost for set  $S$  (the model cost encompassing all regularity in the string)

and the log of the sets cardinality (the required cost to enumerate the equally likely set elements) corresponds to an MDL two part description for string  $x$ . We seek a universal compression algorithm that differentiates the regular (model) and random (data) portions of the compressed string and minimizes the sum of these descriptions (an MDL description). Consider that a codebook of phrases were to be derived as the optimal codebook or model induced from a set of data in an MDL sense. If all regularity is contained in the selection of the codebook related probability distribution (the string model), a set of typical strings generated from this model could be easily constructed. The codebook and probability distribution is then considered an estimated Algorithmic Minimum Sufficient Statistic, the size of which is an estimate of string sophistication [8]. Compressed string size remains an estimate of Kolmogorov Complexity.

The entropy of a distribution of symbols defines the average per symbol compression bound in bits per symbol for a prefix free code. Huffman coding and other strategies can produce an instantaneous code approaching the entropy when the distribution is known [1]. In the absence of knowledge of the model, one way to proceed is to measure the empirical entropy of the string. However, empirical entropy is a function of the partition and depends on what sub-strings are grouped together to be considered symbols.

Our goal is to optimize the partition (the number of symbols, their length, and distribution) of a string such that the compression bound for an instantaneous code, (the total number of encoded symbols  $R$  time entropy  $H_s$ ) plus the codebook size is minimized. We define the approximate model descriptive cost  $M$  to be the sum of the lengths of unique symbols, and total descriptive cost  $D_p$  as follows:

$$M \equiv \sum_i l_i, D_p \equiv M + R \cdot H_s.$$

While not exact (comma costs are ignored in the Model, while possible redundancy advantages are not considered either), these definitions provide an approximate means of breaking out MDL costs on a per symbol basis. The analysis that follows can easily be adapted to other model cost assumptions.

## III. SYMBOL COMPRESSION RATIO

In seeking to partition the string so as to minimize the

Extended Summary Submitted for review to the DIMACS Workshop on Complexity and Inference, June 2–6 2003. The authors are with GE Research and Rensselaer Polytechnic Institute\*. This work is funded in part by DARPA Fault Tolerant Networks Project contract F30602-01-C-0182.

total string descriptive length  $D_p$ , we consider the length that the presence of each symbol adds to the total descriptive length and the amount of coverage of total string length  $L$  that it provides. Since the probability of each symbol,  $p_i$ , is a function of the number of repetitions of each symbol, it can be easily shown that the empirical entropy for this distribution reduces to:

$$H_s = \log_2(R) - \frac{1}{R} \sum_i r_i \log_2(r_i) .$$

Thus, we have:

$$D_p = R \log_2(R) + \sum_i l_i - r_i \log_2(r_i)$$

with

$$R \log_2(R) = \sum_i r_i \log_2(R) = c \sum_i r_i$$

where  $c = \log_2(R)$  is a constant for a given partition of  $R$  symbols. Defining:

$$c = \log_2\left(\frac{L}{2}\right) ,$$

(estimating  $R$  as  $L/2$ ) enables a per symbol formulation for  $D_p$  and results in a conservative approximation for  $R \log_2(R)$  over the likely range of  $R$ . The per-symbol descriptive cost can now be formulated:

$$d_i = r_i [\log_2(L/2) - \log_2(r_i)] + l_i$$

Thus we have a heuristic that conservatively estimates the descriptive cost of any possible symbol in a string considering both model and data (entropy) costs. A measure of the compression ratio for a particular symbol is simply the descriptive length of the string divided by the length of the string "covered" by this symbol. We define the Symbol Compression Ratio (SCR) as:

$$\tilde{\lambda}_i = \frac{d_i}{L_i} = \frac{r_i \left[ \log_2\left(\frac{L}{2}\right) - \log_2(r_i) \right] + l_i}{l_i r_i}$$

This heuristic describes the "compression work" a candidate symbol will perform in a possible partition of a string.

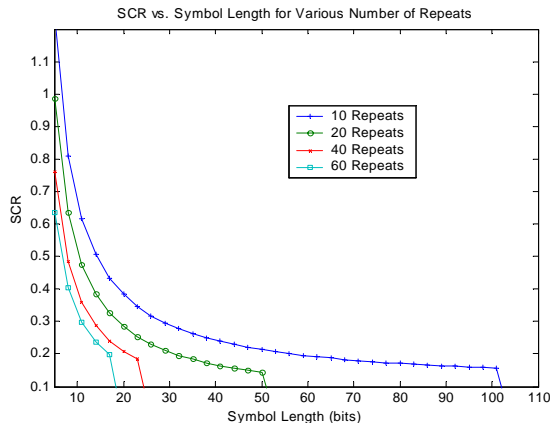


Figure 1 SCR vs. Symbol Length for 1024 bit String

Examining SCR above it is clear that good symbol

compression ratio arises in general when symbols are long and repeated often. But clearly, selection of some symbols as part of the partition is preferred to others. Figure 1 shows how symbol compression ratio varies with the length of symbols and number of repetitions for a 1024 bit string.

#### IV. OPTIMAL SYMBOL COMPRESSION RATIO (OSCR) ALGORITHM

The Optimal Symbol Compression Ration (OSCR) algorithm forms a partition of string  $x$  into symbols that have the best symbol compression ratio among possible symbols contained in  $S$ . The algorithm is as follows:

##### OSCR Algorithm

1. Form a binary tree of sub-strings contained in  $S$  that occur  $\geq$  some user defined minimum frequency and note the frequency of occurrence.
2. Calculate the SCR for all nodes (sub-strings). Select the sub-string from this set with the smallest SCR and add it to the model  $M$ .
3. Replace all occurrences of the newly added symbol with a unique character to delineate this symbol. Repeat steps 1 and 2 with the remaining binary string elements until no binary elements remain.
4. When a full partition has been constructed, use Huffman coding or another coding strategy to encode the distribution,  $p$ , of symbols.

The following comments can be made regarding this algorithm:

1. This algorithm progressively adds symbols that do the most compression "work" among all the candidates to the code space. Replacement of these symbols left-most-first will alter the frequency of remaining symbols.
2. A less exhaustive search for the optimal SCR candidate is possible by concentrating on the tree branches that dominate the string or searching only certain phrase sizes.

As an example consider the 40 bit string below:

$$x = 0011001001000010100101001100100110011001.$$

The first pass of the algorithm will produce the binary tree of symbol frequencies shown in Figure 2. Note, in building this tree we noted and utilized the fact at the second level of the tree the SCR of 12 repetitions of the symbol 01 is  $< .5$ , thus we did not expand tree nodes with 2 or less repeats. The symbol 001 is repeated 10 times and has the smallest symbol compression ratio. Iterating the algorithm produces the partition: ABAACCACBACBABAABABA. The entropy of this symbol distribution is 1.48 bits per symbol. This can be approximated by the Huffman encoding indicated, which achieves an expected encoded length of 1.5 bits per symbol. Encoded with this binary tree the original string is mapped to:

$$x' = 101110000100011000110111011011$$

Thus the encoded message has been reduced to 30 bits from the original 40 bits. The descriptive cost of the codebook is greater than the sum of the lengths of symbols, which is equal to 5 bits. Depending on the strategy for delineating the separation between code words and defining the prefix free

encoding of the codebook, descriptive cost could increase by as much as  $I \log_2(I)$  bits.

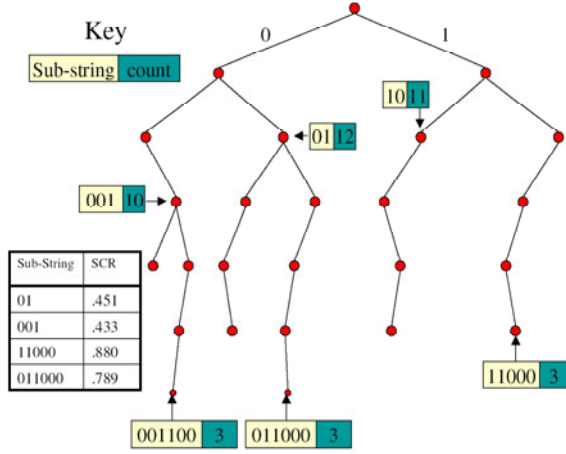


Figure 2 OSCR Tree

Substring	Symbol	Probability	New Code
001	A	.5	0
1	B	.3	10
0	C	.2	11

Table 1 Symbol Distribution

The previous example illustrates the concept of the OSRC algorithm. As is the case with LZ78 and other compression algorithms, greater compression is realized on strings of longer length. In contrast with LZ78 which forms a codebook greedily based on a single pass and adds unique phrases as they appear, the partition for OSRC is formed based on a heuristic that considers two part encoding costs.

## V. TYPICAL SET GENERATION

A typical set containing given string  $x$  can easily be constructed from the OSRC codebook and the probability distribution inherent in the partition. The combinatorial counting method known as ordered selection with specified replacement [9], which fixes the number of times each object in the set can be chosen defines the size of a typical set for a given partition. The number of ways to make an ordered selection of  $R$  repetitions from a set of  $I$  OSRC symbols with exactly  $r_i$  selections of object  $i$  is:

$$|S| = \frac{R!}{r_1! r_2! \dots r_I!}.$$

Taking the logarithm of this set size enables comparison between this estimate and the data portion  $RH_s$  of our decomposition. Using a logarithmic form of Stirling's formula:

$$\log_2(n!) \approx \log_2 \sqrt{2\pi} + (n + .5) \log_2 n - n \log_2 e,$$

we have:

$$\log_2 |S| = \log_2(R!) - \log_2(r_1! r_2! \dots r_I!),$$

$$= [R + .5] \log_2 R - [I - 1] \log_2 \sqrt{2\pi} - \sum_{i=1}^I [r_i + .5] \log_2(r_i),$$

$$= RH_s + .5 \log_2 R - [I - 1] \log_2 \sqrt{2\pi} - .5 \sum_{i=1}^I \log_2(r_i).$$

Dividing the above equation by  $R$  and letting  $R$  approach infinity results in a per-symbol cost that approaches the entropy as expected. The example string in the previous section would have a typical set such that:

$$\log_2 |S| = \log_2 \left( \frac{20!}{10!6!4!} \right) = 25.2 \text{ bits}.$$

The Stirling's formula approximation above achieves an estimate within .05 bits of this result.

An alternative method to using Huffman coding for encoding this string is the send the OSRC model and probability distribution through some efficient means followed by the index (in this case requiring 26 bits) in the lexicographic enumeration of all strings in this set. The sum of the codeword lengths in the codebook is put forward as a starting point for estimating model cost size. This provides the opportunity to consider model size (sophistication) of a string in addition to complexity estimation.

## VI. CONCLUSIONS

Symbol compression ratio is a heuristic useful for two-part code formulation. The algorithm provides a computable (although computationally more expensive than LZ based algorithms) two-part universal compression technique that provides not only an estimate of Kolmogorov Complexity but also an estimated algorithmic minimum sufficient statistic capable of generating a typical set from which a string belongs. A more thorough presentation of current work and future work addresses precise methods of model encoding, methods for minimizing computational expense, and applications of this and related techniques.

## REFERENCES

- [1] Cover, T. M. and Thomas, J. A. Elements of Information Theory. Wiley, NY, 1991
- [2] Evans, S, Bush, S. F., and Hershey, J., "Information Assurance through Kolmogorov Complexity", DARPA Information Survivability Conference & Exposition II, 2001, Proceedings Vol 2, pp 322-331.
- [3] Evans, S. and Barnett, Bruce, "Conservation of Complexity for Information Assurance," MILCOM 2003.
- [4] Li, M. and Vitányi, P. An Introduction to Kolmogorov Complexity and Its Applications, Springer, NY 1997
- [5] Gacs, P., Tromp, J. T., and Vitanyi, P. "Algorithmic Statistics", IEEE Transactions on Information Theory, Vol 47, No 6, September 2001, pp. 2443-2463.
- [6] Ziv, J. and Lempel, A. "Compression of individual sequences via variable length coding," IEEE Trans. Inform. Theory, vol IT-24, pp. 530-536, 1978.
- [7] Li, M. Chen, Li, X. Max, B. and Vitanyi, P. "The Similarity Metric" 14th ACM-SIAM conference on discrete algorithms, January 12-14, 2003 Baltimore, MD, USA
- [8] P. Vitanyi, Meaningful information, Proc. 13th International Symposium on Algorithms and Computation (ISAAC'02), Lecture Notes in Computer Science, Vol 2518, Springer-Verlag, Berlin, 2002, 588-599.
- [9] Rosen, K. H. Handbook of Discrete and Combinatorial Mathematics, CRC Press, New York, 2000.