Compressing and Randomly Accessing Sequences
Laith Ali Abdulsahib  Diego Arroyuelo  Rajeev Raman

Introduction
We consider the following problem. Given a static sequence $X[1..n]$ of $n$ symbols from an alphabet $\{0,\ldots,\sigma - 1\}$, where $\sigma \leq n$, to store it in a compressed form while supporting the following operation:

$$\text{ACCESS}(i): \text{returns } X[i].$$

We consider sequences with “large” alphabets, specifically where $\log_\sigma n$ is small. Examples include time series data, data used for sequential pattern mining and from an algorithm for representing BDDs using a method by Hansen et al. [5]. Existing sequence compressors that target higher-order entropy of $X$ may perform poorly on such sequences.

We investigate the effectiveness of the following measure of compression for such sequences $X$, while preserving fast ACCESS. We create a new sequence $X'$ that is comprised of differences between successive elements of $X$, specifically, $X'[i] = X[i] - X[i-1]$ (take $X[0] = 0$). The measure we consider is: $H_{\text{gap}}^{\text{comp}}(X) = H_0(X')$. Such measures are not entirely new, as predictive coding followed by entropy coding is a standard technique. However, the problem of storing $X$ using $H_0^{\text{comp}}(X)$ bits such that ACCESS is supported quickly is not well studied.

Theoretical result

**Theorem**

A sequence $X$ can be stored in $H_{\text{gap}}^{\text{comp}}(X) + O(n) + o(S)$ bits and support ACCESS in $O(1)$ time, where $S = \sum_{i=1}^{n} |X'[i]|$.

This is obtained by partitioning the elements of $X'$ into subsequences of non-negative ($X^+$) and negative ($X^-$) values, using a compressed bit-vector to separate the two, and applying [1, Theorem 7] to each of $X^+$ and $X^-$. This result is, however, unattainable in practice due to the $o(S)$ term.

Experiments (Datasets)

- NASDAQ: Obtained from values of the NASDAQ stock index from 1972 to the present.
- Insect: Obtained from insect wing beat sound data, obtained from the UEA/UCR time series classification repository [2].
- FIFA: Sequences of click stream data from the website of FIFA World Cup 98 [4].
- Queens: The sequence of non-tree edge endpoints arising in the BDD compression algorithm of Hansen et al. [5], for a BDD of the 14-queens function.

NASCAD and Insect are created by concatenating 1000 copies of the original dataset, each copy scaled by adding a random value.

Experiments (Implementations)

We implemented/tested the following, $\text{AP}(X)$, the alphabet-partitioning data structure [3]; $\text{WT}(X)$, a balanced wavelet tree data structure, with bit vectors compressed using Raman et al’s approach [7]; and $\text{BWT}(X)$, Burrows-Wheeler compressed suffix array. We use the $\text{ads}_{\text{1}}$ implementations of these data structures. In addition we compared with: $\text{Huffman}(X)$, which divides the original sequence into blocks, and Huffman-codes each block. ACCESS is supported by randomly accessing a block (using headers) and decoding a block (using a modification of Turpin’s code [6]). Finally, Ours partitions $X$ into $X^+$ and $X^-$ as above, and stores them using essentially the same blocked Huffman as above. In each case, varying the block size yields a space/time trade-off.

The measures targeted are $H_0(\text{AP}, \text{Huffman}, \text{WT}), H_{\text{gap}}^{\text{comp}}(\text{Ours})$ and higher-order entropy $\text{BWT}$.

Results

Main Conclusions, Future Work

- BWT performs badly in both space and time.
- Either Huffman (Queens, FIFA) or Ours (NASDAQ, Insects) usually performs the best.

It would be useful to consider replacing Turpin’s codes by Asymmetric Numerical System codes, or by DAC codes. Another direction is F2V codes, such as Tustin codes.

References