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 $\sigma \le n$, to store it in a compressed form while supporting the following operation:

ACCESS(i): 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: $H_0^{gap}(X) = H_0(X')$. Such measures are not entirely new, as predictive coding followed by entropy coding is a standard technique. However, the problem of storing X using $H_0^{gap}(X)$ bits such that ACCESS is supported quickly is not well studied.

Theoretical result

Theorem

A sequence X can be stored in $H_0^{gap}(X) + O(n) + o(S)$ 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, unattractive 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.

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

Dataset	n	σ	$H_0(X)$	$H_0^{gap}(X)$
NASDAQ	12,286,701	10,359	12.66	5.80
Insect	56,483,460	11,357	11.39	8.08
FIFA	741,092	2,990	8.48	9.36
Queens	9,572,417	296,300	0.87	1.62

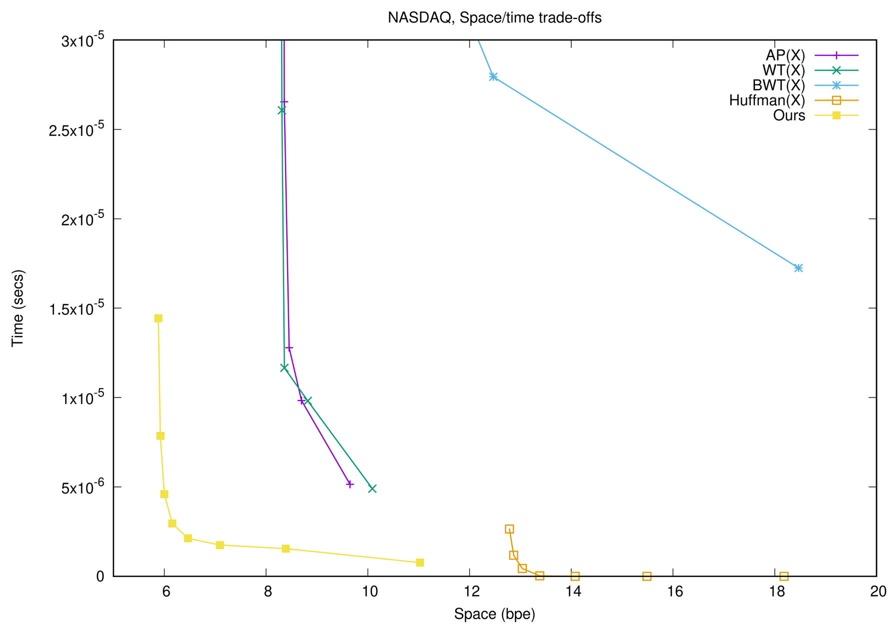
- For the time series data, $H_0^{gap}(X)$ is significantly smaller than $H_0(X)$.
- Queens data is dominated by repetitions of a single element, hence the anomaly.

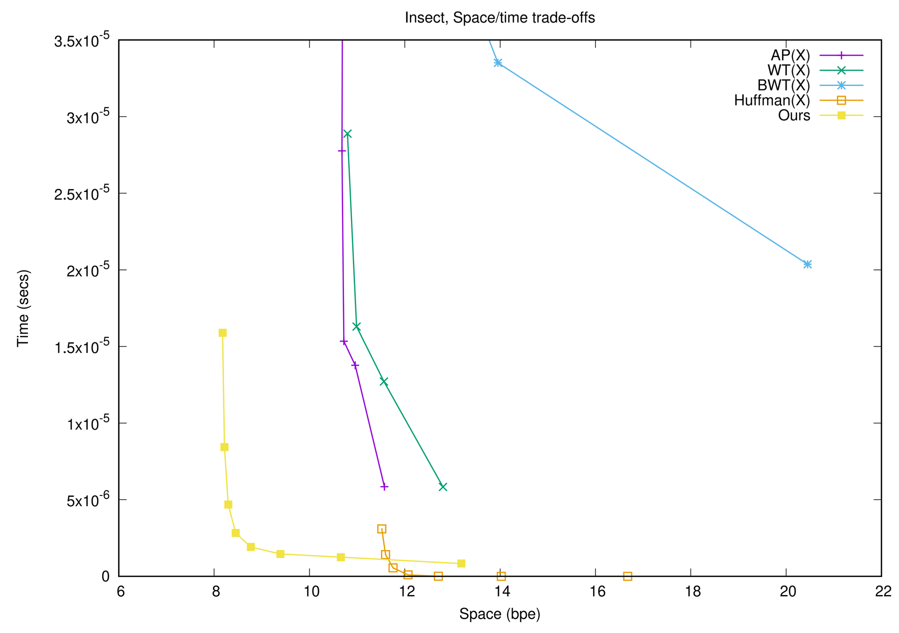
Experiments (Implementations)

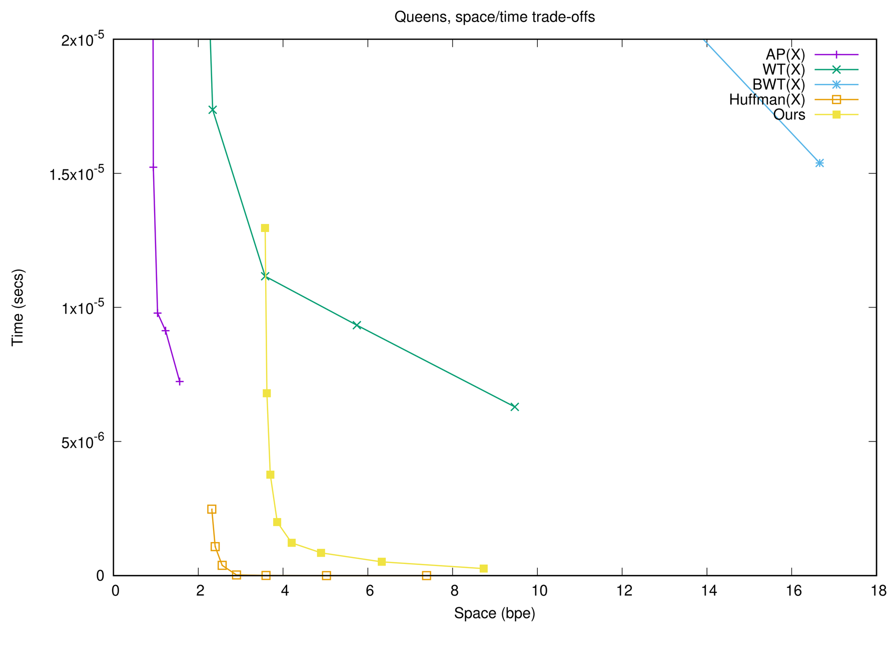
We implemented/tested the following. `AP(X)`, the alphabet-partitioning data structure [3]; `WT(X)`, a balanced wavelet tree data structure, with bit vectors compressed using Raman et al's approach [7]; and `BWT(X)`, Burrows-Wheeler compressed suffix array. We use the `sdsl` implementations of these data structures. In addition we compared with: `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.

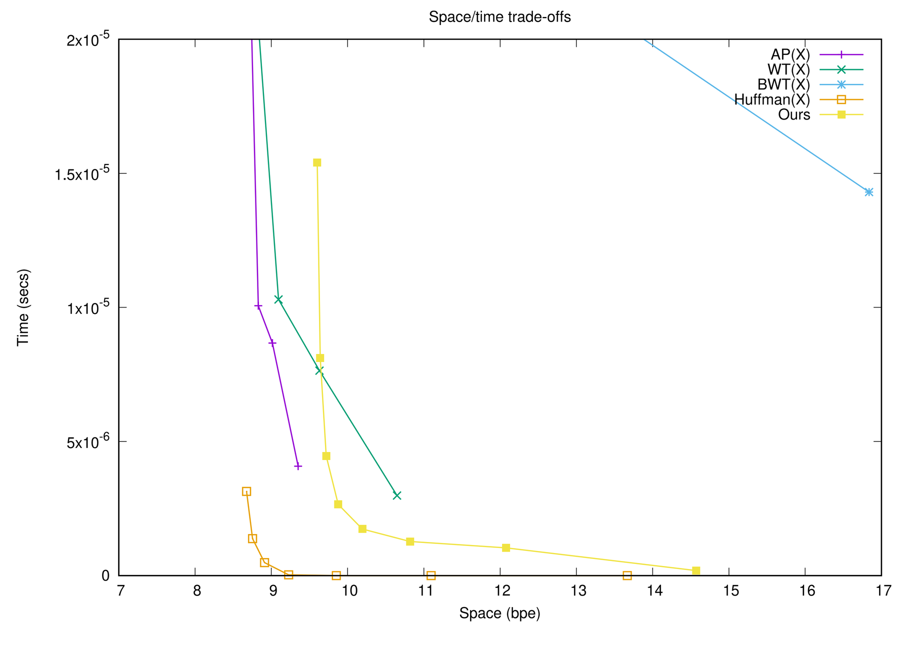
The measures targeted are H_0 (`AP`, `Huffman`, `WT`), H_0^{gap} (`Ours`) and higher-order entropy `BWT`.

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 Numeral System codes, or by DAC codes. Another direction is F2V codes, such as Tunstall codes.

References

- Diego Arroyuelo and Rajeev Raman. "Adaptive Succinctness". In: *String Processing and Information Retrieval - 26th International Symposium, SPIRE 2019, Segovia, Spain, October 7-9, 2019, Proceedings*. Ed. by Nieves R. Brisaboa and Simon J. Puglisi. Vol. 11811. Lecture Notes in Computer Science. Springer, 2019, pp. 467–481. ISBN: 978-3-030-32685-2. DOI: 10.1007/978-3-030-32686-9_33. URL: https://doi.org/10.1007/978-3-030-32686-9%5C_33.
- A. Bagnall et al. "The Great Time Series Classification Bake Off: a Review and Experimental Evaluation of Recent Algorithmic Advances". In: *Data Mining and Knowledge Discovery* 31 (3 2017), pp. 606–660.
- Jérémy Barbay et al. "Efficient Fully-Compressed Sequence Representations". In: *Algorithmica* 69.1 (2014), pp. 232–268. DOI: 10.1007/s00453-012-9726-3. URL: https://doi.org/10.1007/s00453-012-9726-3.
- Philippe Fournier-Viger et al. "The SPMF Open-Source Data Mining Library Version 2". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III*. Ed. by Bettina Berendt et al. Vol. 9853. Lecture Notes in Computer Science. Springer, 2016, pp. 36–40. ISBN: 978-3-319-46130-4. DOI: 10.1007/978-3-319-46131-1_8. URL: https://doi.org/10.1007/978-3-319-46131-1%5C_8.
- Esben Rune Hansen, S. Srinivasa Rao, and Peter Tiedemann. "Compressing Binary Decision Diagrams". In: *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*. Ed. by Malik Ghallab et al. Vol. 178. Frontiers in Artificial Intelligence and Applications. IOS Press, 2008, pp. 799–800. ISBN: 978-1-58603-891-5. DOI: 10.3233/978-1-58603-891-5-799. URL: https://doi.org/10.3233/978-1-58603-891-5-799.
- Alistair Moffat and Andrew Turpin. "On the implementation of minimum redundancy prefix codes". In: *IEEE Trans. Communications* 45.10 (1997), pp. 1200–1207. DOI: 10.1109/26.634683. URL: https://doi.org/10.1109/26.634683.
- Rajeev Raman, Venkatesh Raman, and Srinivasa Rao Satti. "Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets". In: *ACM Trans. Algorithms* 3.4 (2007), p. 43. DOI: 10.1145/1290672.1290680. URL: https://doi.org/10.1145/1290672.1290680.