technische universität dortmund

computer science 12

Bachelor/Master Thesis

Smart Striding in Scheduling Tables for Improved General Purpose Compression

Lars Willemsen
Dr.-Ing. Georg von der Brüggen

Otto-Hahn Str. 16
Technische Universität Dortmund
Email: lars.willemsen@tu-dortmund.de
August 19, 2026

Certification based scheduling may call for the use of fully deterministic scheduling behavior that is not supported by the use of scheduling algorithms directly. In such cases, we may want to use *scheduling tables* directly linking time to task allocations without any dynamic arbitration. Currently, such scheduling tables are of prevalent use in e.g. Time Triggered Ethernet[1].

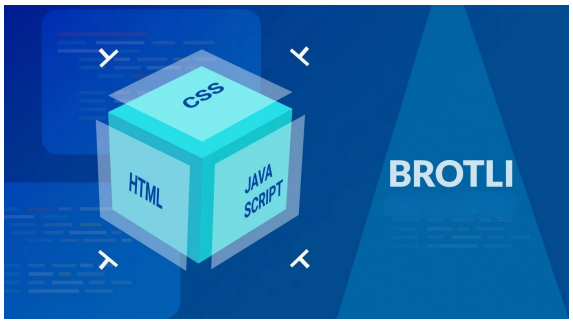

Figure 1: Brotli is a SOTA general purpose compressor

Unfortunately, such tables may become large in practice, especially when their host is a resource-constrained microcontroller. Currently, there exists no purpose-built compression method for such tables, making the immediate fallback options rely on general purpose compressors. State Of The Art (SOTA) compressors are often built from two primitive blocks:

- Lempel-Ziv compression (LZ77 or LZ78) [2]
- Huffman frequency coding [3]

Specifically LZ77 based compression squashes redundancy in the form of runs of repeating items, and struggles with interleavings, even if regular. As such, a pattern like `AAAAA` is highly preferable to a pattern like `ABABA`. In order to make the input data more susceptible to compression by SOTA general-purpose compressors we may apply a *striding strategy*. A striding strategy is an invertible transformation to the input data which may both transform as well re-arrange symbols.

One example of a commonly encountered pattern in scheduling tables is the **Arithmetic Progression** (AP). Such a structure is described by a period T, a phase ϕ and a cardinality (the amount of elements):

$$A_i \;=\; \{\, \phi_i + k \cdot T_i \mid k \in \{0, 1, \ldots, n_i - 1\} \,\} \qquad (1)$$

When such structures are present, we may instead reduce runs of elements to *deviations* from the predicted pattern. For example the pattern `[0, 5, 11, 15, 17, 25, 30]` may be described as deviations from the AP $\{0 + k5\}$, resulting in `[0, 0, 1, 0, -3, 0, 0]`. If we rearrange this data we obtain a contiguous run of five zeroes: a highly compressable pattern.

In this thesis, you will familiarize yourself with three SOTA compressors. You will then be given a dataset of scheduling tables which we aim to compress with aforementioned compressors without inventing a new compressor: instead, we wish to stride the data in such a way the resultant compressed data becomes smaller than the compressor would have achieved without applying striding. You will be given access to a novel striding method for APs, which has several open questions attached to it. You are also allowed to explore your own methods of striding if reasonable ground for it can be established. In the thesis you will then argue for redundancy you found in the tables that the SOTA compressors themselves my not be susceptible to squashing without the aid of striding.

Required Skills:

- Highly familiar with programming in Python
- Familiarity with basic Information Theory

Acquired Skills after the thesis:

- You have become deeply familiarized with the basic building blocks of general purpose compression
- You have explored and argued about redundancy in an established source
- You have argued for improved SOTA compression through smart striding

References:

[1] TTTech Aerospace. `https://www.tttech.com/aerospace`

[2] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[3] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.