A picture containing outdoor, water, sky, sunset

Description automatically generatedWith the usage of Adaptive Huffman Trees (AHTs), an always-optimal Huffman Tree is built while the data is being processed, so the compression ratio can easily be calculated any step along the way. The following graphs show this compression ratio as a file of the RGB triple bytes of the sunset image to the right is compressed. The compression ratio is defined as the average number of bits in the compressed file needed to represent each byte of the original file. So, for an unaltered, uncompressed file, the compression ratio is 8. Anything below that for compressed files is considered a successful compression.

This first graph only considers bits of the compressed file that directly stand for data. Overhead is entirely eliminated in this display; therefore, this in practice is not a useful graph when measuring the compression. However, it allows study of points in the data where notable activity occurs in the compression. For example, sudden inclines in the graph (such as around 1700 bytes) indicate a frequency shift of the file’s alphabet to which the AHT must now adapt. Red indicates points that are a minimum of the previous 64 points.

As expected, the general trend is up and to the right, steeply at first, because with only few bytes processed at the beginning, the AHT tree has relatively shallow depths for all characters, thus few bits are needed to represent them. As the alphabet expands via encountering new characters, additional depth is needed to make them accessible. Still, the compression ratio manages to stay below 4.6, offering a compaction of at least 42%.

The next graph adds in the overhead imposed by the AHT. This is a much more realistic metric to tell the compression ratio, and it paints a much smoother curve that follows the intuition of overheads in compressed files: at the start, the AHT takes up most of the space and thus the compression is very poor. As the amount of data compressed dwarfs the AHT, compression starts to become valuable, crossing the magic 8 number at about byte 43. This example shows a final compression ratio of 4.7. Red has the same meaning as before.

The purpose of identifying moving minimum/maximum/average trends is to attempt to find a forking point in the data at which to begin a new chunk. The difficulty of this is of course speculating how the data will compress afterwards whether a new chunk is created or not. The third graph shows two forks on compression of the sunrise image. Using orange on the blue curve to indicate moving minima, a new chunk was made around byte 35000 to now follow the gray curve. Then, before the steep incline at about byte 103000, the yellow curve breaks off into its own chunk. These points were chosen visually from this graph, which is a liberty an efficient compressor does not possess.

The challenge of optimizing these forking points is indeed immense. It is likely best done with a moderate amount of lookahead before making the decision. Even then, it may be best to create multiple AHTs running alongside each other representing different forks that may or may not tailor out well. Looking at the gray curve, benefits from forking do not come until several thousand bytes later due to the need for a new AHT, which adds immediate overhead. Yet, despite being forked later and thus having less time to show its advantage, the yellow curve lowers the compression ratio more.

The large amount of analysis, speculation, and extra computation through trials may not be worth the small gain of roughly 2% in compression, especially considering how this image was chosen specifically to emphasize the utility of multiple chunks: there is a clear breaking point in the frequency of characters corresponding to the horizon in the image. Most files will not have this fortuitous arrangement.