DCC IEEE

Presented at the 2007 IEEE
Data Compression Conference
Snowbird, Utah — March 27–29

Bit Recycling with Prefix Codes

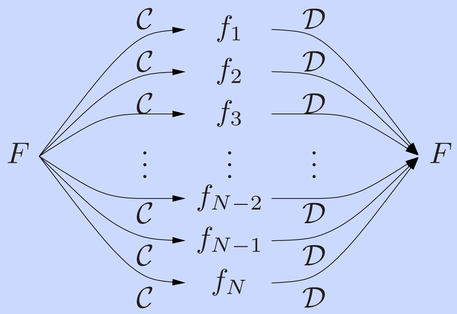
Danny Dubé
Danny.Dube@ift.ulaval.ca

Vincent Beaudoin
Vincent.Beaudoin.1@ulaval.ca

Université Laval
Canada

This work was funded by the Natural Science and Engineering Research Council of Canada.

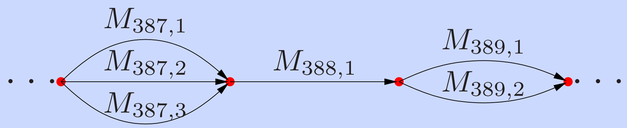
Multiplicity of encodings



$\mathcal{C} \equiv$ Compression $\quad \mathcal{D} \equiv$ Decompression

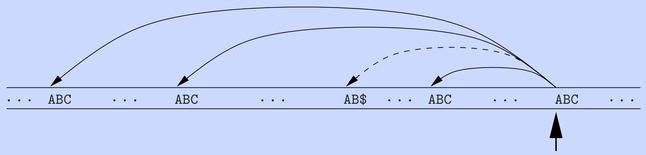
Many lossless compression methods allow a file F to be encoded as one of many equivalent compressed files.

Equivalent messages



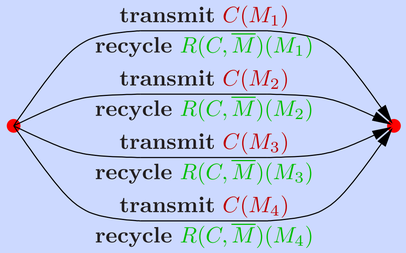
Let us view a compressed file as a concatenation of messages.
Redundancy from equivalent messages *only* means that:
1. The number N of messages depends only of the original data.
2. The ith message can be chosen among n_i *equivalent* messages, $M_{i,1}, \ldots M_{i,n_i}$, for $1 \leq i \leq N$.

Example: LZ77



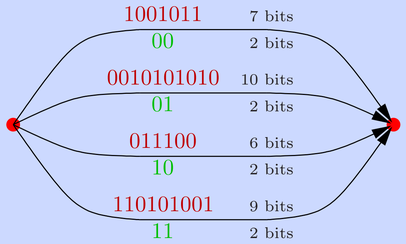
There are 3 longest matches for `ABC`; these are considered to be *equivalent messages*.
The match to `AB` (the dashed arrow) would not be an equivalent message.

Recycling in general



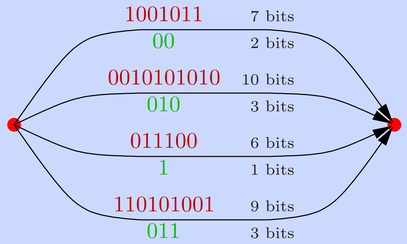
where C encodes messages,
$\overline{M} = \{M_1, M_2, M_3, M_4\}$ is the selected equivalence class,
and R is the recycling function.

Flat recycling



Each of the n_i options causes about $\log_2 n_i$ bits to be recycled.
Average cost: $\frac{1}{4}(7-2) + \frac{1}{4}(10-2) + \frac{1}{4}(6-2) + \frac{1}{4}(9-2) = 6$ bits.

Proportional recycling



The recycled bits associated with option $M_{i,j}$ are determined using Huffman algorithm by assigning frequency $2^{-|M_{i,j}|}$ to the option.
Average cost: $\frac{1}{4}(7-2) + \frac{1}{8}(10-3) + \frac{1}{2}(6-1) + \frac{1}{8}(9-3) = 5.375$ bits.
Proportional recycled leads to an average improvement of 2.1% [2].
We believe proportional recycling to be close to optimal. In some cases, it is more profitable to drop some costly options than to keep them all.

Related work

The idea of exploiting the redundancy of some data compression methods, namely the LZ77 derivatives, was presented earlier. It has been used for information hiding (steganography), authentication, and error correction.
Bit recycling based on LZ77 was presented by Dubé and Beaudoin [1, 2] and, later, by Yokoo [3]. In the latter technique, recycling is not done on a per-message basis. Instead, a compressed file is split in two parts: a first (long) part that is effectively transmitted and a second (short) one that is *embedded* inside the first part using recycling.

Future work

- Developing an efficient technique to perform optimal proportional recycling.
- Adapting bit recycling to arithmetic coding.
- Extending bit recycling to contexts in which redundancy does not come solely from equivalent messages, e.g. in LZ77 compression, taking shorter matches into account.

Pseudo-code of algorithms

Regular Algorithms

```
1. while description incomplete do
2.     let C := curr. coding funct.;
3.     let M := select message;
4.     emit C(M);
where
5. procedure emit w:
6.     σ := σ · w;
7.     return;
```

Compressor

```
1. while description incomplete do
2.     let C := curr. coding funct.;
3.     let M := receive C;
4.     interpret M;
where
5. procedure receive C:
6.     let M, σ' s.t. C(M) · σ' = σ;
7.     σ := σ';
8.     return M;
```

Decompressor

Algorithms with Recycling

```
1. while description incomplete do
2.     let C := curr. coding funct.;
3.     let M̄ := possible messages;
4.     let M := ND-select in M̄;
5.     emit C(M);
6.     recycle R(C, M̄)(M);
where
7. procedure emit w:
8.     if w = ε or ρ = ε then
9.         σ := σ · w;
10.    else if w = b · w' and ρ = b · ρ'
11.    /* where b ∈ {0,1} */ then
12.        ρ := ρ';
13.        emit w';
14.    else
15.        error;
16.    return;
17. procedure recycle w:
18.    ρ := w · ρ;
19.    return;
```

Compressor

```
1. while description incomplete do
2.     let C := curr. coding funct.;
3.     let M := receive C;
4.     interpret M;
5.     let M̄ := equiv. class of M;
6.     recycle R(C, M̄)(M);
where
7. procedure receive C:
8.     let M, σ' s.t. C(M) · σ' = σ;
9.     σ := σ';
10.    return M;
11.
12.
13.
14.
15.
16.
17. procedure recycle w:
18.    σ := w · σ;
19.    return;
```

Decompressor

Resolution algorithm

Non-determinism is not required in order to build the compressed file.
Let σ_i be the bit stream that describes messages $M_i, \ldots, M_N$, for $1 \leq i \leq N+1$.
We can compute σ_{N+1} down to σ_1 the following way:

- Let σ_{N+1} be the encoding of "end of description". We presume that σ_N can be artificially extended by appending dummy bits, if necessary.
- We can obtain σ_i from σ_{i+1} the following way:
  1. We define:
     σ_i to be the stream *before* transmission;
     σ'_i to be the stream *after* transmission but *before* recycling;
     σ''_i to be the stream *after* recycling.
  2. Let $\sigma''_i = \sigma_{i+1}$.
  3. Let C_i be the coding function used at step i. Let $\overline{M}_i$ be the set of candidate messages at step i. Let $M_{k_i} \in \overline{M}_i$ such that there is a stream σ'_i such that $R(C_i, \overline{M}_i)(M_{k_i}) \cdot \sigma'_i = \sigma''_i$.
  4. Let $\sigma_i = C_i(M_{k_i}) \cdot \sigma'_i$.
- σ_1 *is* the compressed file (maybe with some header prepended).

The resolution algorithm can be made greedy.

References

[1] D. Dubé and V. Beaudoin. Recycling bits in LZ77-based compression. In *Proceedings of the Conférence des Sciences Électroniques, Technologies de l'Information et des Télécommunications (SETIT 2005)*, Sousse, Tunisia, mar 2005.

[2] D. Dubé and V. Beaudoin. Improving LZ77 data compression using bit recycling. In *Proceedings of the International Symposium on Information Theory and Applications (ISITA)*, Seoul, South Korea, oct 2006.

[3] H. Yokoo. Lossless data compression and lossless data embedding. In *Proceedings of the Asia-Europe Workshop on Concepts in Information Theory*, Jeju, South Korea, oct 2006.