Error-Resilient LZW data compression

Yonghui Wu       Stefano Lonardi
University of California, Riverside

Wojciech Szpankowski
Purdue University, West Lafayette

Problem definition

• How to achieve joint source and channel coding in LZW (i.e., by adding error resiliency)
  – by keeping backward-compatibility with the original LZW?
  – and without significantly degrading the compression performance
Decoding (with errors)

- Corrupted Lena.gif → GIF decoder (LZW std) → ?
- Corrupted Lena.gif → GIF decoder (LZW+RS) →?

Roadmap

- We will show how to embed extra redundant bits in LZW
- We will show how to achieve error resiliency in LZW
Some related works

- Storer and Reif, “Error-resilient optimal data compression”, SICOMP, 1997
- Lonardi and Szpankowski, “Joint source-channel LZ’77 coding”, DCC’03
- Shim, Ahn and Jeon, “DH-LZW: lossless data hiding in LZW compression”, ICIP’04

Greedy-LZW vs. relaxed-LZW
Is relaxed-LZW backward-compatible?

• We tested the decoding of non-greedy phrases
  – in the GIF format using MS paint, IE, and Mozilla
  – in the ZIP format using Winzip
  – in the .Z format using Unix Compress

• All LZW decoders we tested uses hash tables for the dictionary, so multiple identical entries in the dictionary do not cause any problem

Embedding extra bits in LZW

• Relax *some* of the phrases in the parsing (do not relax *too many* otherwise compression degrades)

• The pattern of occurrence of non-greedy phrases encodes for the extra information being embedded
Embedding extra bits in LZW

\[ M = \begin{array}{cccccc}
K & L & K & L & K & L \\
k_1 & l_1 & k_2 & l_2 & k_3 & l_3
\end{array} \]

- greedy phrases
- relaxed phrases

Count phrases longer than \(2^L\)

Selection of \(K\) and \(L\)

- \(K\) and \(L\) controls the capacity of the message-embedding channel
- Generally, compression ratio degrades as the channel capacity increases
- Need to determine the best trade-off, such that the channel capacity is sufficient for the parity bits, but not much more than that
Channel capacity estimation

- Want to estimate the capacity of the message-embedding channel, given $K$, $L$, $n$, and $H$, where $n$ is the length of the text $T$ to be compressed and $H$ is the entropy of $T$

- To simplify the model, we assume
  - The length of the phrases are always greater than $2^L$
  - The message $M$ to be embedded is generated by an i.i.d. source with 0 and 1 having equal probabilities

- The text $T$ can be logically decomposed into $T_1$ and $T_2$, where $T_1$ is encoded by the greedy phrases and $T_2$ is encoded by non-greedy phrases. Let $n_1 = |T_1|$, $n_2 = |T_2|$

- The average length of greedy phrases is equal to $\log n_1/H$

- Solving a set of equations for $|M|$ gives the estimated channel capacity (next slide)

- Estimation is fairly accurate
Channel capacity estimation

\[
\hat{k} = \frac{2^K + 1}{2}, \hat{l} = \frac{2^{L} + 1}{2} \\
l_1 = \frac{\log n_1}{H}, l_2 = l_1 - \hat{l} \\
B = \frac{n_1}{l_1\hat{k}}, n_2 = B(l_1 - \hat{l}) \\
n = n_1 + n_2 = \ldots = n_1 + \frac{n_1}{\hat{k}} - \frac{n_1H\hat{l}}{k\log n_1} \\
m = B(K + L) = \ldots = \frac{n_1H}{k\log n_1}(\hat{k} + \hat{l})
\]

Towards error-resiliency

- Typical LZW implementation uses a fixed size dictionary (usually 4,096)
- As soon as the dictionary is full, it is flushed and refreshed, and a special EOD symbol is inserted into the LZW file
- Those EOD symbols logically break the text into self-contained *chunks*
Error-resilient encoding/decoding

Implementation

• We are still working on a full implementation of the error-resilient LZW
• We have implemented a new GIF encoder that is capable of embedding the bits of another file
• The “augmented” GIF is decodable by any standard programs, but if given to our decoder the bits of the second file are recovered
• Available at http://www.cs.ucr.edu/~yonghui/
Experimental results (GIF)

<table>
<thead>
<tr>
<th>size of the compressed image with $M$ embedded</th>
<th>estimated message length</th>
</tr>
</thead>
<tbody>
<tr>
<td>size of the compressed image</td>
<td>size of the message $M$ embedded</td>
</tr>
</tbody>
</table>

| Image   | $T'$ | $I$ | $T_{se}$ | $|M|$ | $|M|$ | $|T'|-|M|$ | $|M|/|T'|$ | estim. $|M|$ |
|---------|------|-----|----------|------|------|-----------|----------|----------|
| airplane | 64908 | 5.77 | 64468    | 1706 | 5.63 | -146      | 0.02628  | 1980     |
| baboon  | 149414 | 2.49 | 151804   | 2169 | 2.45 | 221       | 0.01451  | 4678     |
| couple  | 19664  | 4.88 | 20088    | 595  | 4.77 | -21       | 0.02576  | 587      |
| girl    | 23573  | 4.04 | 24127    | 566  | 3.94 | -12       | 0.02401  | 712      |
| lena    | 96373  | 3.87 | 98770    | 2396 | 3.78 | 1         | 0.02486  | 2973     |
| peppers | 105262 | 3.54 | 10792    | 2372 | 3.46 | 158       | 0.02253  | 3258     |

 averag phrase length | average phrase length after embedding

$K = 5$, $L = 1$

Findings

- Method to recover extra redundant bits from LZW
- Extra bits allow to incorporate error-resiliency in LZW
  - backward-compatible (deployment without disrupting service)
  - compression degradation due to the extra bits is minimal