KDS-transformation for data compression

IONUȚ POPA
Faculty of Computer Science,
“Al.I.Cuza” University of Iași,
700483 Iași,
ROMANIA

Abstract:- In this paper we present a new simple method to rewrite a given input string into another one which can be efficiently compressed (in some cases) by traditional compression methods.

Key-Words:- Data compression, Reversible transformation

1. Introduction

It is an well known observation in data compression theory that, given an input source, symbols with low probabilities are responsible for the high entropy of the input source. Lets call those symbols rare symbols and the others dominant symbols.

At least empirically, an input source with rare symbols would be more efficiently compressed if those symbols can be encoded somehow using dominant symbols. Obviously, this encoding should be completely reversible. We present here such a method, similar somehow with difference encoding and other transformation techniques used in image or sound compression. The fundamentals of this method can also be found in Huffman compression algorithm.

The paper is structured as follows: in the next section the basic idea of our transformation is presented. In Section 3, an analysis of this technique is presented. In the last sections some conclusions and remarks are also given.

2. KDS - transformation

KDS -transformation stands for Keep the Dominant Symbols transformation. The advantage of this method is that those statistical properties of the input stream that can be efficiently used by any compression algorithm are not dramatically changed.

Consider the following procedure to build a Huffman code for an alphabet $V$, where $p_a$ is the probability of appearance of symbol $a$ in an input source:

- if $|V| = 2$ then $V = \{a, b\}$ and $\text{code}(a) = 0$ and $\text{code}(b) = 1$.
- if $|V| > 2$ then select from $V$ two symbols with smallest probabilities (call them $a$ and $b$). Then $\text{code}(a) = \text{code}(\#)0$, $\text{code}(b) = \text{code}(\#)1$, where $\#$ is a new symbol with $p_{\#} = p_a + p_b$, and $\text{code}(\#)$ is computed applying the same procedure with the alphabet $V \setminus \{a, b\} \cup \{\#\}$.

The same is the essence of our transformation, to replace symbols with low probabilities with a new symbol.

Further we present formally this transformation.

Consider $A$ is an alphabet and $w$ is a word over the alphabet $A$. Consider $A = R \cup D$ with $|R| = |D|$, where $R$ is a subset of the alphabet $A$ containing the ”rare” symbols and $D$ a subset containing the ”dominant” symbols in the word $w$.

It is easy to see that there is an unique decomposition of $w$:

$$w = r_1d_1r_2d_2 \ldots r_kd_k,$$
The length (in bits) of the Lampel-Ziv[2] code of a word \( w \in V^* \) is:

\[
LZ_w = \sum_{i=1}^{b} \lceil \log_2(k_i) \rceil,
\]

where \( b \) is the number of blocks resulting from LZ-parsing and \( k \) is the size of the alphabet of the string \( w \).

Consider now the case of the alphabet \( A \) where \( |A| = 2^p, p > 1 \). Let \( u \) and \( v \) be two words, \( u \in A^* \) and \( v \in (A \cup \{\#\})^* \), \( v \) is obtained form \( u \) using the above transformation.

\[
LZ_u = \sum_{i=1}^{b} \lceil \log_2(2^p i) \rceil = bp + \sum_{i=1}^{b} \lceil \log_2(i) \rceil
\]

which telescope[1] into

\[
LZ_u = bp + b(|\log_2 b| + 1) + b2^{1-\log_2 b + |\log_2 b|} - 2,
\]

where \( b \) is the number of blocks resulting from LZ parsing of the string \( u \).

\[
LZ_v = b'(p - 1) + b'(|\log_2 b'| + 1) + b'2^{1-\log_2 b' + |\log_2 b'|} - 2,
\]

where \( b' \) is the number of blocks resulting from LZ parsing of the string \( v \).

If \( \frac{LZ_v}{LZ_u} > 1 \) compression is improved by transformation presented above.

In Figure 1 the dark spot代表 values for \( b \) and \( b' \) when compression can be improved.

Moreover consider \( A = R \cup D \), where \( |R| = |D| \), and the probabilities of appearance of symbols form \( R \) in \( u \) is \( p_1 \), and the probabilities of appearance of symbols from \( D \) in \( u \) is \( p_2 \). In this case we can estimate the number \( b' \) using \( b, p_1 \) and \( p_2 \) and determinate the cases when \( LZ_v \leq LZ_u \).
4. Conclusion

The idea of this transformation can be also used to encode pointers/references in dictionary compression schemes.

An implementation of this algorithm improves compression from 5% up to 10% in combination with software products like gzip or WinRAR 3.30. In the following table some experimental results are presented:

<table>
<thead>
<tr>
<th>Original size</th>
<th>WinRAR</th>
<th>KDS+WinRAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>104448</td>
<td>19926</td>
<td>19756</td>
</tr>
<tr>
<td>263680</td>
<td>68084</td>
<td>68527</td>
</tr>
<tr>
<td>285696</td>
<td>62008</td>
<td>60657</td>
</tr>
<tr>
<td>829440</td>
<td>211659</td>
<td>202124</td>
</tr>
<tr>
<td>913755</td>
<td>293157</td>
<td>287293</td>
</tr>
<tr>
<td>1016320</td>
<td>244126</td>
<td>232476</td>
</tr>
<tr>
<td>1150976</td>
<td>229790</td>
<td>229350</td>
</tr>
<tr>
<td>1691136</td>
<td>424372</td>
<td>407384</td>
</tr>
<tr>
<td>2156032</td>
<td>511538</td>
<td>489263</td>
</tr>
<tr>
<td>3259904</td>
<td>597233</td>
<td>579094</td>
</tr>
<tr>
<td>5751296</td>
<td>1321138</td>
<td>1278913</td>
</tr>
<tr>
<td>6125056</td>
<td>1168909</td>
<td>1126271</td>
</tr>
<tr>
<td>28844032</td>
<td>6409980</td>
<td>6228254</td>
</tr>
</tbody>
</table>

Table 1: Experimental results

The first column of Table 1 contains the size (in bytes) of original files. Tests were performed using Java bytecode files. The second file contains the size of the files compressed using WinRAR 3.30. Finally, the last column contains files encoded first using KDS transformation and then WinRAR 3.30.

In most of the cases KDS transformation improved compression efficiency. It also works well on images, sounds or other files containing "rare" symbols.

References
