Extension and Faster Implementation of the GRP Transform for Lossless Compression

Hidetoshi Yokoo

Gunma University
Japan
Outline

1. **What is the GRP Transform**
   - Sort-based Transforms for Lossless Data Compression
   - Existing Transforms -> Parametric Generalization

2. **Proposed Extension**
   - Forward Transformation
   - Faster Implementation of Inverse Trans.

3. **Compression Experiments**

4. **Conclusions and Future Work**
Lossless Compression with Sort-Based Transforms

Transform: Conversion of one string to another
Lossless Compression with Sort-Based Transforms

- Burrows-Wheeler Transform (BWT)
- Sort Transform (ST)
- RadixZip

[Second-Step Encoders for Transformed Strings]
Actual encoders for transformed text (LGT: Local to Global Transform, Adjeroh-Bell-Murkherjee ‘08)
## Sort-based Transforms for Lossless Data Compression

<table>
<thead>
<tr>
<th>Transforms</th>
<th>Context Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWT (Burrows &amp; Wheeler 1994)</td>
<td>Unbounded</td>
</tr>
<tr>
<td>ST (Schindler 1997)</td>
<td>Constant</td>
</tr>
<tr>
<td><em>Permute</em> in RadixZip (Vo &amp; Manku 2008)</td>
<td>0, 1, ..., Constant (Cyclically Changing)</td>
</tr>
</tbody>
</table>

– Sorting = Context-Gathering
## Sort-based Transforms for Lossless Data Compression

<table>
<thead>
<tr>
<th>Transforms</th>
<th>Context Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWT (Burrows &amp; Wheeler 1994)</td>
<td>Unbounded</td>
</tr>
<tr>
<td>ST (Schindler 1997)</td>
<td>Constant</td>
</tr>
<tr>
<td>Permute in RadixZip (Vo &amp; Manku 2008)</td>
<td>0, 1, ..., Constant (Cyclically Changing)</td>
</tr>
<tr>
<td>Our Transform (Generalized Radix Permutation) (ITY, SPIRE2009)</td>
<td>$d$, $d + 1$, ..., $d + \ell - 1$</td>
</tr>
</tbody>
</table>

- Parameterized Generalization
Aims of Generalization

– Proper Generalization
– For Data Consisting of Subunits
  – Genetic Codes (Triplet Codons), Multi-byte Data, 2D Data
– Explicit Use of Context Information
  – Towards a Generalization of Second-Step Encoders
## Proposed Extension

*(Generalized Radix Permutation)*

<table>
<thead>
<tr>
<th>Transforms</th>
<th>Context Lengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWT (Burrows &amp; Wheeler 1994)</td>
<td>Unbounded</td>
</tr>
<tr>
<td>ST (Schindler 1997)</td>
<td>Constant</td>
</tr>
<tr>
<td><em>Permute</em> in RadixZip (Vo &amp; Manku 2008)</td>
<td>0, 1, ..., Constant (Cyclically Changing)</td>
</tr>
<tr>
<td>Our Transform <em>(Generalized Radix Permutation)</em></td>
<td>$d, d + 1, ..., d + \ell - 1$</td>
</tr>
</tbody>
</table>

- Parameterized Generalization
Proposed Extension
(Generalized Radix Permutation)

<table>
<thead>
<tr>
<th>Transform</th>
<th>Context Lengths</th>
</tr>
</thead>
</table>

- Arbitrary Parameter Values

\[ 1 \leq \ell \leq n, \quad 0 \leq d \leq n \] for text length \( n \)

- Linear Time Algorithm for Inverse Transformation

\[ d, d+1, \ldots, d + \ell - 1 \]

- Parameterized Generalization

- Modification of MTF for the Transform
Outline

1. What is the GRP Transform
   - Sort-based Transforms for Lossless Data Compression
   - Existing Transforms -> Parametric Generalization

2. Proposed Transform
   - Forward Transformation
   - Faster Implementation of Inverse Trans.

3. Compression Experiments

4. Conclusions and Future Work
Forward Transformation

Divide into $\ell$ - symbol blocks ($\ell = 3$)

Data: \[ b \ a \ c \ a \ c \ a \ b \ a \ c \ a \$

$\ell = 3$

\[ b \ a \ c \  a \ c \ a \  b \ a \ c \  a \$\$ \]
Forward Transformation

Data: bac ac ac ab ac a$

bac ac ac bac ac a$$
Forward Transformation

Generate all cyclic shifts of the blocks

Data: $\begin{array}{cccc}bac & aca & bac & a$$\$
    
    aca & bac & a$$\$ & bac
    
    bac & a$$\$ & bac & aca
    
    a$$\$ & bac & aca & bac
\end{array}$
Generate all cyclic shifts of the blocks

Data: \[ \text{bac} \quad \text{aca} \quad \text{bac} \quad \text{a}\$\$ \]

\[ \text{aca} \quad \text{bac} \quad \text{a}\$\$ \quad \text{bac} \]

\[ \text{bac} \quad \text{a}\$\$ \quad \text{bac} \quad \text{aca} \]

\[ \text{a}\$\$ \quad \text{bac} \quad \text{aca} \quad \text{bac} \]

\[ d = 4 \]

Sort the row vectors according to the left \( d \) symbols

\( \$ \) is supposed to be the alphabetically largest symbol.)
Forward Transformation

Sorted row vectors according to the left $d$ symbols

Data:

(Sort key)
**Forward Transformation**

Output the symbols in the rightmost column

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>$</th>
<th>$</th>
<th>b</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$</td>
<td>$</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

Data:

<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>$</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>$</td>
<td>$</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>

流向：

| c | c | $ | a |
Forward Transformation

Output the symbols in the rightmost column

```
acac  bac  a$$$  bac

acac  bac  aca  bac
```

Data: bac  aca  bac  a$$

```
bac  a$$  bac  aca
```

```
cc$a
```

Sort again the row vectors according to the rightmost symbols. (Perform a stable sort.)
Forward Transformation

Perform a **stable** sort according to the rightmost symbols.

Data: 

```
<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>
```
Output the symbols in the column, second from right.

```
<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>
```

Data:  

```
| b | a | c | a | c | a | a | a | a | $ |
```

Output:  

```
| c | c | $ | a | c | a | a | $ |
```
Forward Transformation

Output the symbols in the column, second from right.

\[
\begin{array}{ccc}
  b & a & c \\
  a & c & a \\
  a & c & a
\end{array}
\]

\[
\begin{array}{ccc}
  b & a & c \\
  a & c & a \\
  a & c & a
\end{array}
\]

Data: \[
\begin{array}{ccc}
  b & a & c \\
  a & c & a \\
  b & a & c \\
  a & c & a \\
\end{array}
\]

\[
\begin{array}{ccc}
  c & c & a \\
  c & c & a \\
  c & c & a \\
  c & c & a \\
\end{array}
\]

Perform a stable sort again on the row vectors according to the outputted symbols.
Forward Transformation

Perform a stable sort according to the outputted symbols.

Data:

\[
\begin{array}{cccc}
\text{a} & \text{c} & \text{a} & \text{b} & \text{a} & \text{c} \\
\text{a} & \text{a} & \text{c} & \text{b} & \text{a} & \text{c} \\
\text{b} & \text{a} & \text{c} & \text{a} & \text{a} & \text{c} \\
\text{b} & \text{a} & \text{c} & \text{c} & \text{a} & \text{c} \\
\text{b} & \text{a} & \text{c} & \text{a} & \text{a} & \text{c} \\
\text{b} & \text{a} & \text{c} & \text{a} & \text{a} & \text{c} \\
\end{array}
\]
Forward Transformation

Output the symbols in the column, third from right.

```
acac bac a$$ b ac
a$$ bac aca bac
bac a$$ bac aca
```

Data: bac aca bac a$$

```
c c$ a c a a b b a a
```
Forward Transformation

Output the symbols in the column, third from right.

```
\ | | | |
\ | | | |
\ | | | |
```

Data:  

```
\ | | | |
\ | | | |
\ | | | |
```

Transformation completed!
Inverse Transformation

1. Main contribution of the present work
2. Linear time/space implementation
   – Application of the technique of Nong-Zhang-Chan for the ST [CPM’08]
   – Context switch vector and cycles
3. We need not reconstruct the matrix representation, instead
4. We use only auxiliary quantities all of size $O(n/\ell)$
**Inverse Transformation**

**Actual Procedure** *(Conceptually quite simple)*

1. Reconstruction of the output part

2. Computation of four auxiliary vectors
   - Permutation vector $Q[1..n/\ell]$
   - Context switch vector $D[1..n/\ell]$
   - Index vector $T_d[1..n/\ell]$
   - Counter vector $C_d[1..n/\ell]$

3. Restoring an original string
Inverse Transformation

Reconstruction of the output part

\[
\begin{array}{ccc}
\text{a} & \text{c} & \text{a} \\
\text{b} & \text{a} & \text{c} \\
\text{b} & \text{a} & \text{c} \\
\text{a} & \text{c} & \text{a} \\
\end{array}
\]

Transformed string:

\[
\text{c c a c a a b b a a}
\]
Inverse Transformation

Reconstruction of the output part

\[
\begin{array}{ccc}
\text{b} & \text{a} & \text{c} \\
\text{b} & \text{a} & \text{c} \\
\text{a} & \$ & \$ \\
\text{a} & \text{c} & \text{a} \\
\end{array}
\]

Sort the obtained output part lexicographically
Inverse Transformation

Computation of auxiliary vectors

1  a c a  4  b a c  1
2  a $ $  3  b a c  2
3  b a c  1  a $ $  3
4  b a c  2  a c a  4
## Inverse Transformation

<table>
<thead>
<tr>
<th>$i$</th>
<th>$Q[i]$</th>
<th>$T_d[i]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>aca</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>a$$</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>bac</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>bac</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>aca</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>bac</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>a$$</td>
<td>3</td>
</tr>
</tbody>
</table>
### Inverse Transformation

<table>
<thead>
<tr>
<th>$i$</th>
<th>$Q$</th>
<th>$D$</th>
<th>$C_d$</th>
<th>$T_d$</th>
<th>$C_d[i]$</th>
<th>$D[i]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>aca</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>bac</td>
</tr>
<tr>
<td>2</td>
<td>a$</td>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>bac</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>a$</td>
</tr>
<tr>
<td>4</td>
<td>bac</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>aca</td>
</tr>
</tbody>
</table>

$C_d[i]$: Counter vector

$D[i]$: Context switch vector
**Inverse Transformation**

<table>
<thead>
<tr>
<th></th>
<th>Q</th>
<th>D</th>
<th>C_d</th>
<th>T_d</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>bac</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>bac</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>a $$</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>aca</td>
</tr>
</tbody>
</table>

\[ i := \text{index of the row that includes } \$; \]

**Repeat:**

Stack the \( i \)th row;

\[ i := T_d[i]; \]

\[ C_d[i] := C_d[i] - 1; \]

\[ i := i + C_d[i]; \]
### Inverse Transformation

<table>
<thead>
<tr>
<th>i</th>
<th>Q</th>
<th>D</th>
<th>C_d</th>
<th>T_d</th>
<th>Original String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>b a c</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>b a c</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>a $ $</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>a c a</td>
</tr>
</tbody>
</table>

\[
i := \text{index of the row that includes \$};
\]

Repeat:

- Stack the i-th row;
- \(i := T_d[i] ;\)
- \(C_d[i] := C_d[i] - 1 ;\)
- \(i := i + C_d[i] ;\)

Original string recovered!
Outline

1. What is the GRP Transform
   - Sort-based Transforms for Lossless Data Compression
   - Existing Transforms -> Parametric Generalization

2. Proposed Transform
   - Forward Transformation
   - Faster Implementation of Inverse Trans.

3. Compression Experiments

4. Conclusions and Future Work
Compression Experiments

- Generalized Radix Permutation Transform (GRP)
- MTF
- Run Length Huffman Coding
- Distance Coding
- Inversion Frequencies
- etc.

[Second-Step Encoders for Transformed String]
Actual encoders for transformed text (LGT: Local to Global Transform, Adjero-Bell-Murkherjee ‘08)
We have modified the MTF (Move-to-Front) heuristics so that it takes advantage of the output characteristics of the GRP transform. (AMTF: Adaptive MTF)
Output Characteristics

The symbols in the rightmost column:

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>$</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>$</td>
<td>$</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>c</td>
<td>b</td>
</tr>
</tbody>
</table>

Data:

<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>$</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>$</td>
<td>$</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>

The situation is the same as in the BWT in the sense that their contexts are not observable.
Output Characteristics

The symbols in the second column, second from right:

Data:

However, the situation is changed. We can observe their immediate right symbols.
When we encode the symbols in the $k$th column, $k$th from right,

we can use their $k-1$ right symbols as their immediate contexts.

**AMTF = MTF + Contextual information**

adaptively changes the MTF list when the context changes according to:

When the context changes completely, the MTF list of symbols is initialized so that the order of symbols corresponds to the frequency of symbols occurred in shorter similar contexts.
AMTF: Adaptive MTF

The output part

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>b</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>
```
Encoding the $k$th column

The MTF list of symbols is initialized using the symbol frequencies in shorter similar contexts.
# Compression Results

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\ell$</th>
<th>File</th>
<th>cp.html</th>
<th>alice29</th>
<th>lct10</th>
<th>plabn12</th>
<th>ptt5</th>
<th>kennedy</th>
<th>world192</th>
<th>bible</th>
<th>E.coli</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>MTF</td>
<td>2.68</td>
<td>2.80</td>
<td>2.76</td>
<td>3.16</td>
<td>1.00</td>
<td>0.88</td>
<td>2.73</td>
<td>2.46</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\infty$</td>
<td>1</td>
<td>MTF</td>
<td>2.68</td>
<td>2.80</td>
<td>2.75</td>
<td>3.15</td>
<td>0.90</td>
<td>1.22</td>
<td>2.69</td>
<td>2.45</td>
<td>2.27</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>MTF</td>
<td>3.22</td>
<td>2.96</td>
<td>2.98</td>
<td>3.24</td>
<td>1.04</td>
<td>0.70</td>
<td>3.00</td>
<td>2.68</td>
<td>2.20</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>AMTF</td>
<td>3.18</td>
<td>2.95</td>
<td>2.97</td>
<td>3.24</td>
<td>1.04</td>
<td>0.70</td>
<td>2.99</td>
<td>2.68</td>
<td>2.20</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>MTF</td>
<td>3.20</td>
<td>2.89</td>
<td>2.90</td>
<td>3.20</td>
<td>1.01</td>
<td>0.90</td>
<td>2.82</td>
<td>2.58</td>
<td>2.18</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>AMTF</td>
<td>3.17</td>
<td>2.89</td>
<td>2.89</td>
<td>3.20</td>
<td>1.00</td>
<td>0.90</td>
<td>2.82</td>
<td>2.57</td>
<td>2.18</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>MTF</td>
<td>3.33</td>
<td>3.03</td>
<td>2.99</td>
<td>3.23</td>
<td>1.03</td>
<td>0.88</td>
<td>2.98</td>
<td>2.69</td>
<td>2.16</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>AMTF</td>
<td>3.26</td>
<td>3.01</td>
<td>2.98</td>
<td>3.22</td>
<td>1.03</td>
<td>0.87</td>
<td>2.97</td>
<td>2.68</td>
<td>2.16</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>MTF</td>
<td>3.32</td>
<td>2.98</td>
<td>2.94</td>
<td>3.19</td>
<td>1.00</td>
<td>0.99</td>
<td>2.85</td>
<td>2.60</td>
<td>2.17</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>AMTF</td>
<td>3.25</td>
<td>2.97</td>
<td>2.93</td>
<td>3.18</td>
<td>1.00</td>
<td>0.98</td>
<td>2.83</td>
<td>2.60</td>
<td>2.17</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>AMTF</td>
<td>4.69</td>
<td>4.31</td>
<td>4.30</td>
<td>4.33</td>
<td>1.11</td>
<td>2.38</td>
<td>4.58</td>
<td>4.10</td>
<td>2.14</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>AMTF</td>
<td>3.74</td>
<td>3.53</td>
<td>3.58</td>
<td>3.67</td>
<td>1.04</td>
<td>1.33</td>
<td>3.75</td>
<td>3.34</td>
<td>2.18</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>AMTF</td>
<td>3.17</td>
<td>2.88</td>
<td>2.89</td>
<td>3.20</td>
<td>0.97</td>
<td>1.16</td>
<td>2.79</td>
<td>2.57</td>
<td>2.18</td>
</tr>
</tbody>
</table>
Compression Results

File: world192.txt

Compression Rate (bit/byte)

$d = 0$

$d = 3$

$d = \infty$ : BWT

$d = 0$; cumulative

$d = 0$; final column

$d = 3$; cumulative

$d = 3$; final column
Conclusions and Future Work

<table>
<thead>
<tr>
<th>Transform</th>
<th>Context Lengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>We developed the Generalized Radix</td>
<td>$d, d + 1, \ldots, d + \ell - 1$</td>
</tr>
<tr>
<td>Permutation transform [2009]</td>
<td></td>
</tr>
</tbody>
</table>

- We have extended it to a more general version (applicable to arbitrary parameter values).
- Linear time implementation of the inverse transformation.
- Development of a new second-step encoder (AMTF).
- Its compression performance is slightly better than the conventional MTF when combined with the GRP transform.
- Development of other second-step encoders.
Lossless Compression with Sort-Based Transforms

- Burrows-Wheeler Transform (BWT)
- Sort Transform (ST)
- RadixZip

MTF → Run Length Huffman
Distance Coding
Inversion Frequencies
etc.

[Generalized Radix Permutation Transform]
Parametric generalization of BWT, ST, and RadixZip
(0 ≤ d ≤ ℓ)
Compression Experiments

- GRP Transform
- Adaptive MTF
- Run Length Encoding
- Range Coder