Burrows-Wheeler compression with modified sort orders and exceptions to the MTF phase, and their impact on the compression rate

Marc Lehmann

Burrows-Wheeler compression

- M. Burrows and D.J. Wheeler in 1994
- lossless compression
- context-based
- 3 stages in basic form

Burrows-Wheeler compression

- M. Burrows and D.J. Wheeler in 1994
- lossless compression
- context-based
- 3 stages in basic form

Definitions:

symbol	smallest logical unit of information
string	sequence of symbols

The Burrows-Wheeler Transform

Example: `mississippi`

- Write all cyclic shifts into a table.

0	m	i	s	s	i	s	s	i	p	p	i
1	i	s	s	i	s	s	i	p	p	i	m
2	s	s	i	s	s	i	p	p	i	m	i
3	s	i	s	s	i	p	p	i	m	i	s
4	i	s	s	i	p	p	i	m	i	s	s
5	s	s	i	p	p	i	m	i	s	s	i
6	s	i	p	p	i	m	i	s	s	i	s
7	i	p	p	i	m	i	s	s	i	s	s
8	p	p	i	m	i	s	s	i	s	s	i
9	p	i	m	i	s	s	i	s	s	i	p
10	i	m	i	s	s	i	s	s	i	p	p

The BWT

- Sort the table lexicographically to get the *BW table*.
- Last column is the output of the transform (*BW code*).

0	i	m	i	s	s	i	s	s	i	p	p
1	i	p	p	i	m	i	s	s	i	s	s
2	i	s	s	i	p	p	i	m	i	s	s
3	i	s	s	i	s	s	i	p	p	i	m
4	m	i	s	s	i	s	s	i	p	p	i
5	p	i	m	i	s	s	i	s	s	i	p
6	p	p	i	m	i	s	s	i	s	s	i
7	s	i	p	p	i	m	i	s	s	i	s
8	s	i	s	s	i	p	p	i	m	i	s
9	s	s	i	p	p	i	m	i	s	s	i
10	s	s	i	s	s	i	p	p	i	m	i

The BWT

Context Blocks

- *Context block*: block of BW code corresponding to rows with a specific symbol at the beginning.

0	i	m	i	s	s	i	s	s	i	p	p
1	i	p	p	i	m	i	s	s	i	s	s
2	i	s	s	i	p	p	i	m	i	s	s
3	i	s	s	i	s	s	i	p	p	i	m
4	m	i	s	s	i	s	s	i	p	p	i
5	p	i	m	i	s	s	i	s	s	i	p
6	p	p	i	m	i	s	s	i	s	s	i
7	s	i	p	p	i	m	i	s	s	i	s
8	s	i	s	s	i	p	p	i	m	i	s
9	s	s	i	p	p	i	m	i	s	s	i
10	s	s	i	s	s	i	p	p	i	m	i

The BWT

Reversibility

- Every column is a permutation of the input
- First column can be reconstructed by sorting the last

0	i	m	i	s	s	i	s	s	i	p	p	
1	i	p	p	i	m	i	s	s	i	s	s	
2	i	s	s	i	p	p	i	m	i	s	s	
3	i	s	s	i	s	s	i	p	p	i	m	
4	m	i	s	s	i	s	s	i	p	p	i	0
5	p	i	m	i	s	s	i	s	s	i	p	
6	p	p	i	m	i	s	s	i	s	s	i	1
7	s	i	p	p	i	m	i	s	s	i	s	
8	s	i	s	s	i	p	p	i	m	i	s	
9	s	s	i	p	p	i	m	i	s	s	i	2
10	s	s	i	s	s	i	p	p	i	m	i	3

The BWT

Reversibility

- Start index is also needed
- *i*-th occurrence in the first column corresponds to *i*-th occurrence of the last column

0	i	m	i	s	s	i	s	s	i	p	p	
1	i	p	p	i	m	i	s	s	i	s	s	
2	i	s	s	i	p	p	i	m	i	s	s	
3	i	s	s	i	s	s	i	p	p	i	m	
4	m	i	s	s	i	s	s	i	p	p	i	0
5	p	i	m	i	s	s	i	s	s	i	p	
6	p	p	i	m	i	s	s	i	s	s	i	1
7	s	i	p	p	i	m	i	s	s	i	s	
8	s	i	s	s	i	p	p	i	m	i	s	
9	s	s	i	p	p	i	m	i	s	s	i	2
10	s	s	i	s	s	i	p	p	i	m	i	3

The BWT

The Effect

- Substrings of the input beginning with the same symbols are sorted one below the other
- BW code are the symbols immediately preceding them
- Usually only a few distinct symbols in a context block, many runs of the same symbol

The BWT

The Effect

- Substrings of the input beginning with the same symbols are sorted one below the other
- BW code are the symbols immediately preceding them
- Usually only a few distinct symbols in a context block, many runs of the same symbol

For example, context block corresponding to "`nd `" in book1 (first 100 symbols):

```
eaeaaAaaiaaaaaaaaauaaaaoaaaaiaaaiiaauaauaauaiaaaaiuaa
    aaaaaaaaaaaaaaaaaaaiaaaaaaAaaaaaaaaaaaaaaaaaaaa
```

Move-To-Front Coding

- Alphabet initialized with all possible symbols (e.g. `[0x00, 0x01, ..., 0xff]` for bytes)
- Symbols encoded as index in the coder's alphabet
- Encoded symbol is moved to the front of the alphabet

Move-To-Front Coding

- Alphabet initialized with all possible symbols (e.g. `[0x00, 0x01, ..., 0xff]` for bytes)
- Symbols encoded as index in the coder's alphabet
- Encoded symbol is moved to the front of the alphabet

For example, encode `aaabacccba`:
`[97, 0, 0, 98, 1, 99, 0, 0, 2, 2]`

- Generally small numbers
- Runs of same symbols are runs of zeros
- book1: `0`: 49.8%, `1`: 15.4%, `2`: 8%, `3`: 5.3%

Entropy Coding

- Output of MTF has very skewed probabilities, suitable for entropy coding
- Symbols with high probability are encoded with short codes
- Huffman coding, arithmetic coding

Modifying the Sort Order

- B. Chapin, 1998
- Instead of `a` → `b` → `c` → . . ., sort differently
- Transitions between “similar” context blocks means lower MTF codes in the beginning
- Less symbols have to be “fetched from the back” of the alphabet
- Chapin: handpicked order `aeioubcd...`
- Overhead: $\lceil log_2 256! \rceil = 1684$ bits

Computing Orders

- Assign a cost to each transitions between symbols (i.e. What if x was sorted before y?)
- Run Traveling Salesman Heuristic on the costs
- Best tour is the best sort order
- (according to the metric that computed the costs)

Metrics

- Chapin: based on BW code
- Analyze similarities in symbol frequencies of context blocks
- Badness metric: based on the effect on the MTF code
- Attempts to put a number to how “bad” a transition is (for compression)

The Badness Metric

Partial MTF

- Like regular MTF, but start with empty alphabet
- Encode symbols that aren't in it with a special code (−1)

Example `aaabacccba` from earlier:

```
        MTF:   [97, 0, 0, 98, 1, 99, 0, 0, 2, 2]
Partial MTF:   [-1, 0, 0, -1, 1, -1, 0, 0, 2, 2]
```

- Special codes only difference from regular MTF

The Badness Metric

- Want to determine badness value for transition from x to y
- Do BWT with natural order and get the context blocks corresponding to x and y
- Create the partial MTF for context block y and for the concatenation of both

The Badness Metric

- Want to determine badness value for transition from x to y
- Do BWT with natural order and get the context blocks corresponding to x and y
- Create the partial MTF for context block y and for the concatenation of both

For example, `abc` and `ccaadb`:

```
right side: [                -1, 0, -1, 0, -1,  -1]
combined:   [-1, -1, -1,   0, 0,   2, 0, -1,   3]
```

- Metric assumes that context blocks remain unchanged
- Only positions where the right side has special codes are relevant

The Badness Metric

- Compare values at the relevant positions to ideal values

```
abc → ccaadb
 right side: [            -1,  0, -1,  0, -1, -1]
   combined: [-1, -1, -1,  0,  0,  2,  0, -1,  3]
      ideal: [             0,      1,      2,  3]
```

The Badness Metric

- Compare values at the relevant positions to ideal values

```
abc → ccaadb
right side: [            -1,  0, -1,  0, -1, -1]
  combined: [-1, -1, -1,  0,  0,  2,  0, -1,  3]
     ideal: [             0,      1,      2,  3]
```

- Special code in the combined code: symbol only appears in the right side
- No information: assume ideal
- Badness value is the sum of the differences between actual and ideal value

The Badness Metric

Variants

- Weighting: divide value by the number of special codes in the right side
- So (long) blocks with many different symbols aren't punished
- MTF prediction: instead of assuming ideal code, make a guess

  generic mean of all MTF codes greater or equal the ideal code

  specific mean of all MTF codes greater or equal the ideal code, that are encoding the same underlying symbol

The Badness Metric

First Column Only

- Metric assumes context blocks remain unchanged, but this is not true
- When a different order is used, the blocks will also be sorted differently
- More specific problem: only look for order for first column, rest is ordered with the natural order
- This way, context blocks stay as they were, metric's assumption holds

Performance of the Metrics

book1

Metric	weighted	MTF prediction	all columns	first column
Badness	✗	✗	-0.04	0.01
	✗	generic	-0.02	0.02
	✗	specific	0.04	0.01
	✓	✗	0.03	0.02
	✓	generic	0.06	0.02
	✓	specific	0.08	0.02
	“aeiou. . .”		0.08	0.01
histogram differences			0.04	0.01
number of inversions			0.04	0.01
number of inversions log			0.04	0.01

File: book1, size 6150168 bits.

Performance of the Metrics

paper1

Metric	weighted	MTF prediction	all columns	first column
Badness	✗	✗	-0.40	0.02
	✗	generic	-0.21	0.10
	✗	specific	-0.15	0.10
	✓	✗	-0.24	0.12
	✓	generic	-0.10	0.14
	✓	specific	-0.04	0.13
	“aeiou. . .”		0.17	0.02
histogram differences			0.05	0.05
number of inversions			0.11	0.11
number of inversions log			-0.04	0.09

File: paper1, size 425288 bits.

Performance of the Metrics

Observations

- Not much of a difference
- More relative improvement with smaller file size (number of transitions)
- Badness is good for first column, not always for all columns
- Considering overhead, paper1 actually gets bigger

Multiple Sort Orders

- Make separate orders for the first two (or more) columns
- One order: for every distinct symbol in the first column, there's one context block
- Two orders: for every distinct symbol in the first column, for every distinct symbol in the second that follows it, there's one context block
- More transitions to optimize, hopefully more compression gains

Multiple Sort Orders

The BWT

0	i	s	s	i	s	s	i	p	p	i	m
1	i	s	s	i	p	p	i	m	i	s	s
2	i	p	p	i	m	i	s	s	i	s	s
3	i	m	i	s	s	i	s	s	i	p	p
4	m	i	s	s	i	s	s	i	p	p	i
5	p	p	i	m	i	s	s	i	s	s	i
6	p	i	m	i	s	s	i	s	s	i	p
7	s	s	i	s	s	i	p	p	i	m	i
8	s	s	i	p	p	i	m	i	s	s	i
9	s	i	s	s	i	p	p	i	m	i	s
10	s	i	p	p	i	m	i	s	s	i	s

- In general, different orders for all different symbols in the first column

Reversibility

- Can't show reversibility with arbitrary number of orders
- Can show for two orders
- Problem: *i*-th occurrence in first column doesn't correspond to *i*-th occurence in last column anymore
- Solution: "Look ahead" and reorder according to the second column

Reversibility

Looking Ahead

0	i	s	s	i	s	s	i	p	p	i	m	
1	i	s	s	i	p	p	i	m	i	s	s	
2	i	p	p	i	m	i	s	s	i	s	s	
3	i	m	i	s	s	i	s	s	i	p	p	
4	m	i	s	s	i	s	s	i	p	p	i	3
5	p	p	i	m	i	s	s	i	s	s	i	2
6	p	i	m	i	s	s	i	s	s	i	p	
7	s	s	i	s	s	i	p	p	i	m	i	0
8	s	s	i	p	p	i	m	i	s	s	i	1
9	s	i	s	s	i	p	p	i	m	i	s	
10	s	i	p	p	i	m	i	s	s	i	s	

- Get possible indices and sort based on following symbols according to the second order

Performance

Metric	weighted	MTF prediction	all columns	first columns
Badness	✗	✗	-0.06	0.07
	✗	generic	-0.02	0.09
	✗	specific	-0.03	0.09
	✓	✗	-0.06	0.09
	✓	generic	-0.03	0.10
	✓	specific	-0.04	0.09
“aeiou. . . ”			0.08	0.01
histogram differences			-0.04	0.03
number of inversions			0.02	0.06
number of inversions log			0.01	0.06

File: book1, size 6150168 bits.

Performance

Observations

- Better compression than one order if only the first columns are reordered
- Orders for the second column even less suitable as default order
- Overhead for storing the orders ($83 \cdot 1684$ bits) outweighs compression gain
- First columns actually require three orders (natural order as default), not sure if reversible

Exceptions to the MTF

- Context block “nd ” in book1:

```
eaeaaAaaiaaaaaaaaauaaaaoaaaaiaaaiiaauaauaauaiaaaaiuaa
    aaaaaaaaaaaaaaaaaaaiaaaaaaAaaaaaaaaaaaaaaaaaaa
```

- Context block “. ”:

```
leyyytetylnyyykrnytehnnyadyyyrkesyyydalsyyyddlednyyyd
    trgkryesendydnekyayswnregsyrmdeycs.nntyhkdegyd!
```

- Next sentence doesn’t give indication about last letter of last word of previous one
- Many different symbols, no long runs

Exceptions to the MTF

- Symbols are the last letters of words, different probabilities
- But no information is taken from the further context
- Encoding with MTF doesn't make sense in this case
- "Pollutes" the statistics of a static entropy coder

Exceptions to the MTF

- Symbols are the last letters of words, different probabilities
- But no information is taken from the further context
- Encoding with MTF doesn't make sense in this case
- "Pollutes" the statistics of a static entropy coder
- Exclude certain blocks from the MTF phase
- Put a special code in the MTF to signalize a missing block
- Encode with Huffman directly and append

Exceptions to the MTF

Selecting Exceptions

- Encode with MTF as usual
- For every context block: if the mean of all the MTF values is above a threshold, exclude it from the MTF phase
- Also require a certain length of the block, so the compression gains aren't destroyed by the overhead

Exceptions to the MTF

Performance, book1

min length	threshold	gain	excepted blocks
0	3	-0.39	*many*
0	4	0.59	0x00, \n, 0x1a, space, !, &,), *, +, “,”, ., 0, 5, 7, :, ;, =, >, ?, E, U, V, X
100	4	0.59	\n, space, !, +, “,”, ., :, ;, >, ?, E, U
100	4.5	0.60	\n, space, !, +, “,”, ., :, ;, >, ?
100	5	0.59	\n, space, !, +, “,”, ., :, ;, >
100	6	0.26	\n, “,”, .

File: book1, size 6150168 bits.

Exceptions to the MTF

Observations

- Works much better than the reordering stuff
- Good choice for threshold seems to be between 4 and 5

Exceptions to the MTF

Observations

- Works much better than the reordering stuff
- Good choice for threshold seems to be between 4 and 5

Caveats

- Effective with static (two-pass) Huffman coder I use
- Adaptive coder could adapt to temporarily higher MTF codes
- Huffman coder with multiple tables could have one for these bad cases (bzip2)