The BWT as a compression tool

1978 (?)

Space Invaders released

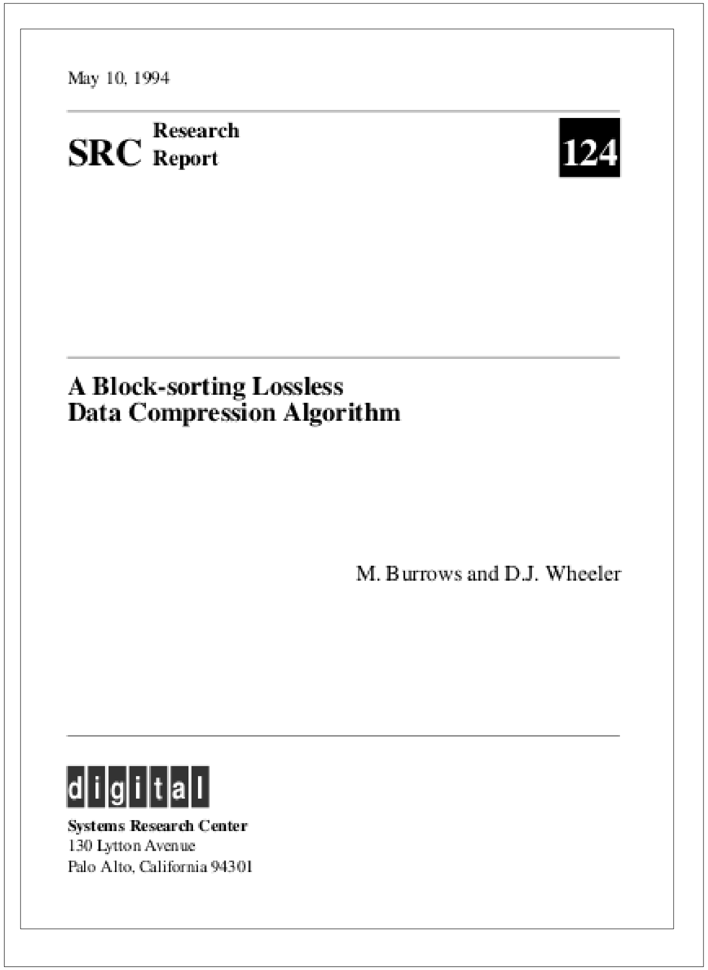
- David Wheeler conceives a data compression algorithm based on reversible transformation on the input text, but considers it too slow for practical use

1994

Mandela first black South Africa president

- Mike Burrows improves the speed of the compressor. B&W co-author the technical report describing a “block sorting” lossless data compression algorithm.
- The algorithm splits the input in blocks and computes a reversible transformation that makes the text “more compressible”
- The transformation has been later called the Burrows-Wheeler transform.

1994



May 10, 1994

SRC Research Report 124

A Block-sorting Lossless Data Compression Algorithm

M. Burrows and D.J. Wheeler

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

The BWT

```
swiss·miss·missing
```

The BWT

swiss·miss·missing

Consider all rotations of the input text

```
s wiss·miss·missin g
w iss·miss·missing s
i ss·miss·missings w
s s·miss·missingsw i
s ·miss·missingswi s
· miss·missingswis s
m iss·missingswiss ·
i ss·missingswiss· m
s s·missingswiss·m i
s ·missingswiss·mi s
· missingswiss·mis s
m issingswiss·miss ·
i ssingswiss·miss· m
s singswiss·miss·m i
s ingswiss·miss·mi s
i ngswiss·miss·mis s
n gswiss·miss·miss i
g swiss·miss·missi n
```

The BWT

```
swiss·miss·missing
```

Consider all rotations of the input text

Sort them in lexicographic order

```
· miss·missingswis s
· missingswiss·mis s
g swiss·miss·missi n
i ngswiss·miss·mis s
i ss·miss·missings w
i ss·missingswiss· m
i ssingswiss·miss· m
m iss·missingswiss ·
m issingswiss·miss ·
n gswiss·miss·miss i
s ·miss·missingswi s
s ·missingswiss·mi s
s ingswiss·miss·mi s
s s·miss·missingsw i
s s·missingswiss·m i
s singswiss·miss·m i
s wiss·miss·missin g
w iss·miss·missing s
```

The BWT

`swiss·miss·missing`

Consider all rotations of the input text

Sort them in lexicographic order

Take the last character of each rotation

`ssnswmm··isssiiigs`

		L
·	miss·missingswis	s
·	missingswiss·mis	s
g	swiss·miss·missi	n
i	ngswiss·miss·mis	s
i	ss·miss·missings	w
i	ss·missingswiss·	m
i	ssingswiss·miss·	m
m	iss·missingswiss	·
m	issingswiss·miss	·
n	gswiss·miss·miss	i
s	·miss·missingswi	s
s	·missingswiss·mi	s
s	ingswiss·miss·mi	s
s	s·miss·missingsw	i
s	s·missingswiss·m	i
s	singswiss·miss·m	i
s	wiss·miss·missin	g
w	iss·miss·missing	s

The BWT

swiss·miss·missing

Consider all rotations of the input text

Sort them in lexicographic order

Take the last character of each rotation

ssnswmm··isssiiigs

F		L
·	miss·missingswis	s
·	missingswiss·mis	s
g	swiss·miss·missi	n
i	ngswiss·miss·mis	s
i	ss·miss·missings	w
i	ss·missingswiss·	m
i	ssingswiss·miss·	m
m	iss·missingswiss	·
m	issingswiss·miss	·
n	gswiss·miss·miss	i
s	·miss·missingswi	s
s	·missingswiss·mi	s
s	ingswiss·miss·mi	s
s	s·miss·missingsw	i
s	s·missingswiss·m	i
s	singswiss·miss·m	i
s	wiss·miss·missin	g
w	iss·miss·missing	s

final char (L)	sorted rotations
a	`n to decompress.  It achieves compression`
o	`n to perform only comparisons to a depth`
o	`n transformation}  This section describes`
o	`n transformation}  We use the example and`
o	`n treats the right-hand side as the most`
a	`n tree for each 16 kbyte input block, enc`
a	`n tree in the output stream, then encodes`
i	`n turn, set $L[i]$ to be the`
i	`n turn, set $R[i]$ to the`
o	`n unusual data. Like the algorithm of Man`
a	`n use a single set of probabilities table`
e	`n using the positions of the suffixes in`
i	`n value at a given point in the vector $R`
e	`n we present modifications that improve t`
e	`n when the block size is quite large.  Ho`
i	`n which codes that have not been seen in`
i	`n with $ch$ appear in the {\em same order`
i	`n with $ch$.                    In our exam`
o	`n with Huffman or arithmetic coding.  Bri`
o	`n with figures given by Bell~\cite{bell}.`

Figure 1: Example of sorted rotations. Twenty consecutive rotations from the sorted list of rotations of a version of this paper are shown, together with the final character of each rotation.

BWT inversion

It is not at all obvious that from the last column we can recover the original string…..

Burrows and Wheeler fundamental observation is that in the first and last columns equal characters are in the same relative order.

F		L
·	miss·missingswis	s
·	missingswiss·mis	s
g	swiss·miss·missi	n
i	ngswiss·miss·mis	s
i	ss·miss·missings	w
i	ss·missingswiss·	m
i	ssingswiss·miss·	m
m	iss·missingswiss	·
m	issingswiss·miss	·
n	gswiss·miss·miss	i
s	·miss·missingswi	s
s	·missingswiss·mi	s
s	ingswiss·miss·mi	s
s	s·miss·missingsw	i
s	s·missingswiss·m	i
s	singswiss·miss·m	i
s	wiss·miss·missin	g
w	iss·miss·missing	s

BWT inversion: demo

BWT **swiss·miss·missing**

= **ssnswmm··isssiiigs**

·miss·missingswis	s
·missingswiss·mis	s
gswiss·miss·missi	n
ingswiss·miss·mis	s
iss·miss·missings	w
iss·missingswiss·	m
issingswiss·miss·	m
miss·missingswiss	·
missingswiss·miss	·
ngswiss·miss·miss	i
s·miss·missingswi	s
s·missingswiss·mi	s
singswiss·miss·mi	s
ss·miss·missingsw	i
ss·missingswiss·m	i
ssingswiss·miss·m	i
swiss·miss·missin	**g**
wiss·miss·missing	s

BWT vs $H_k(s)$ (1)

Let **s** = *ippississim*,

s^R = *mississippi*

$$BWT(s^R) = \begin{array}{l} \text{imississipp} \\ \text{ippimississ} \\ \text{issippimiss} \\ \text{ississippim} \\ \text{mississippi} \\ \text{pimississip} \\ \text{ppimissisi} \\ \text{sippimissis} \\ \text{sissippimis} \\ \text{ssippimissi} \\ \text{ssissippimi} \end{array}$$

BWT vs $H_k(s)$ (1)

Let s = *ippississim*,

s^R = *mississippi*

BWT(s^R) =

F		L
i	mississip	p
i	ppimissis	s
i	ssippimis	s
i	ssissippi	m
m	ississipp	i
p	imississi	p
p	pimississ	i
s	ippimissi	s
s	issippimi	s
s	sippimiss	i
s	sissippim	i

$H_1(s) = (4/11)\ H_0(pssm) + (1/11)\ H_0(i)$
$\qquad + (2/11)\ H_0(pi) + (4/11)\ H_0(ssii)$

To compress up to $H_1(s)$ it suffices to compress each segment up to H_0

BWT vs $H_k(s)$ (2)

Let s = *ippississim*,

s^R = *mississippi*

BWT(s^R) =

```
imississipp
ippimississ
issippimiss
ississippim
mississippi
pimississip
ppimississi
sippimissis
sissippimis
ssippimissi
ssissippimi
```

To compress up to $H_k(s)$ it suffices to compress each segment up to H_0

Summing up

To compress a string up to $H_k(s)$ it suffices to compress the corresponding partition of $BWT(s^R)$ up to H_0 (compare with PPM)

In the first BWT-based compressors this was done implicitly using Move-to-Front followed by an Order0 encoder (Huffman or Arithmetic coding)