Improving FM Index

Juha Kärkkäinen[1]

[1]University of Helsinki, Finland

Algodan Seminar
October 28, 2011

FM Index (Ferragina, Manzini)

Practical succinct self-index of a text

- Pattern counting

space (bits)	query time
$n \log \sigma + o(n \log \sigma)$	$\mathcal{O}(m \log \sigma)$

$n =$ text length
$\sigma =$ alphabet size
$m =$ pattern length

Main components

- Burrows–Wheeler transform (BWT)
- Wavelet tree
- Binary rank index

Text Compression

Text T: $n = |T|$ $\sigma =$ alphabet size

Three levels of compression

1. No compression: $n \log \sigma$ bits
2. Zeroth order compression: $nH_0(T)$ bits
3. Higher order compression: $nH_k(T)$ bits

$H_k(T)$ is the kth order empirical entropy of T

- $\log \sigma \geq H_0(T) \geq H_k(T)$

Zeroth Order Compression of FM Index

- Huffman-shaped wavelet tree (HWT)
- RRR binary rank index

Compression technique	space	query time
None	$n \log \sigma + o(n \log \sigma)$	$\mathcal{O}(m \log \sigma)$
HWT	$nH_0(T) + n + o(nH_0(T) + n)$	$\mathcal{O}(mH_0(T))$ on average
RRR	$nH_0(T) + o(n \log \sigma)$	slower by a constant factor
HWT+RRR	$nH_0(T) + o(nH_0(T) + n)$	

Higher Order Compression by Compression Boosting

- Partition $\mathrm{BWT}(T)$ into blocks suitably
- Compress each block using zeroth order compressor
- The result is higher order compression of T

Theorem (Manzini)

For any text T and any k, there exists a partitioning $L_1 L_2 \ldots L_\ell$ of $\mathrm{BWT}(T)$ into $\ell \leq \sigma^k$ blocks so that

$$\sum_i |L_i| H_0(L_i) = |T| H_k(T)$$

Compression Boosting of FM Index

- Context block boosting (Ferragina et al.)
  - Optimal partitioning of BWT into varying size context blocks
- Implicit boosting (Mäkinen & Navarro)
  - Using RRR achieves the boosting effect
- FM index with either boosting method needs

$$nH_k(T) + o(n \log \sigma)$$

  bits of space

Fixed Block Boosting (Kärkkäinen & Puglisi, SPIRE 2011)

- Partition BWT into blocks of fixed size
- Never much worse than any other partitioning

Theorem

Let $X_1 X_2 \ldots X_\ell$ be any partitioning of S.
Let $Y_1 Y_2 \ldots Y_{n/b}$ be a partitioning of S into blocks of size b.
Then

$$\sum |Y_i| H_0(Y_i) \leq \sum |X_i| H_0(X_i) + (\ell - 1) b$$

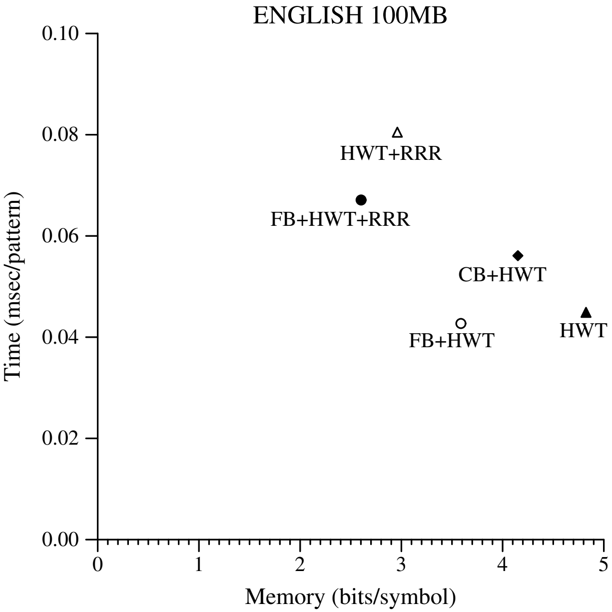
- Achieves the same space of

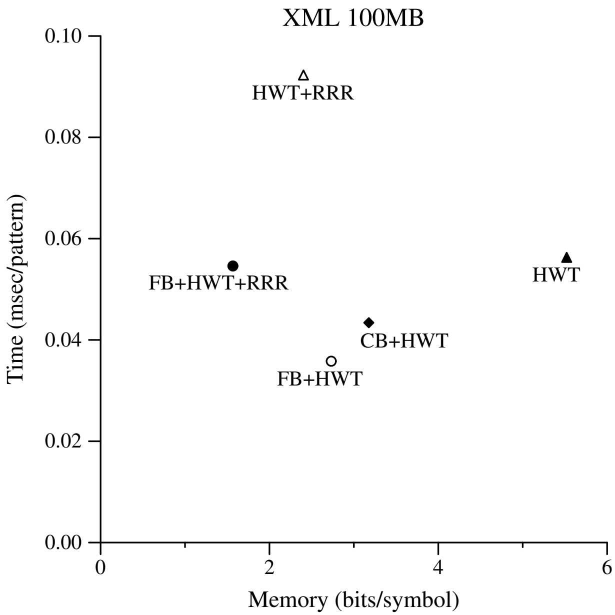
$$nH_k(T) + o(n \log \sigma)$$

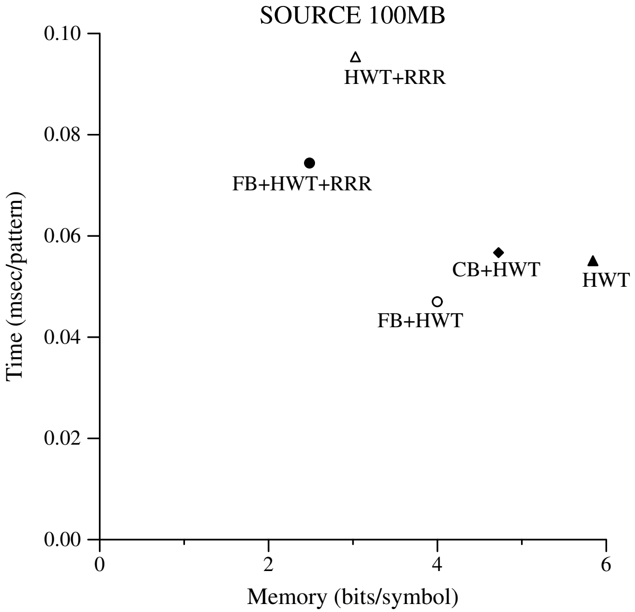
- Simpler, faster and smaller in practice

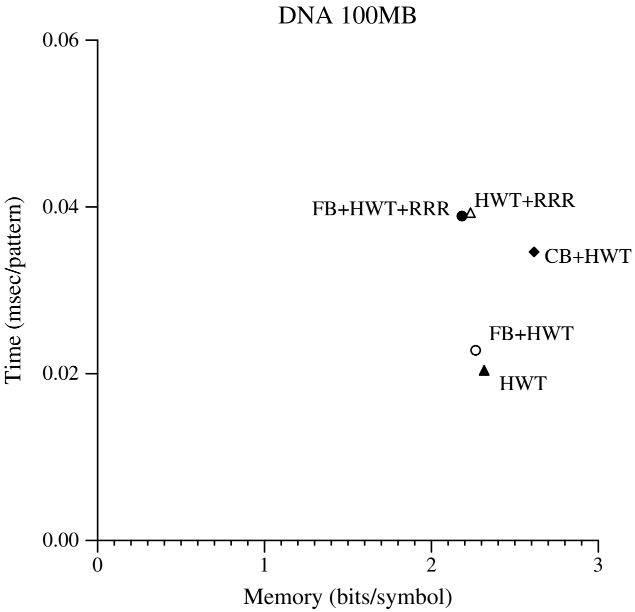
Experimental Results

Previous Implementations (from Pizza & Chili)	New implementations (using the same components)
HWT • No boosting	
CB+HWT • Context block boosting	FB+HWT • Fixed block boosting
HWT+RRR • Implicit boosting	FB+HWT+RRR • Fixed block boosting


ENGLISH 100MB
Time (msec/pattern)
0.10
0.08
0.06
0.04
0.02
0.00
0
1
2
3
4
5
Memory (bits/symbol)
HWT+RRR
FB+HWT+RRR
CB+HWT
FB+HWT
HWT


XML 100MB
Time (msec/pattern)
0.10
0.08
0.06
0.04
0.02
0.00
HWT+RRR
FB+HWT+RRR
HWT
CB+HWT
FB+HWT
0
2
4
6
Memory (bits/symbol)


SOURCE 100MB
Time (msec/pattern)
0.10
0.08
0.06
0.04
0.02
0.00
HWT+RRR
FB+HWT+RRR
CB+HWT
HWT
FB+HWT
0
2
4
6
Memory (bits/symbol)



DNA 100MB
0.06
0.04
0.02
0.00
Time (msec/pattern)
FB+HWT+RRR
HWT+RRR
CB+HWT
FB+HWT
HWT
0
1
2
3
Memory (bits/symbol)

Summary

- Compression boosting improves FM index compression from zeroth order to higher order
- Better compression boosting using blocks of fixed size
  - Same theoretical properties
  - Simple to implement
  - Smaller and faster in practice

Further Improvements

- Algorithm engineering
  - Better boosting
  - Engineering components
  - Supporting other queries
- Getting below $nH_k(T)$
  - Highly repetitive sequences (DNA data, version databases, ...) can be compressed to less than $nH_k(T)$ bits
  - Run-length encoding: RLFM, RLWT, RLFM+
  - Measures of compressibility?