Compression: The Burrows Wheeler Transform

A classical structure: the **Burrows Wheeler transform** (BWT)

```
GATTAGATACAT$
ATTAGATACAT$G
TTAGATACAT$GA
TAGATACAT$GAT
AGATACAT$GATT
GATACAT$GATTA
ATACAT$GATTAG
TACAT$GATTAGA
ACAT$GATTAGAT
CAT$GATTAGATA
AT$GATTAGATAC
T$GATTAGATACA
$GATTAGATACAT
```

Sort lexicographically

BWT

```
$GATTAGATACAT
ACAT$GATTAGAT
AGATACAT$GATT
AT$GATTAGATAC
ATACAT$GATTAG
ATTAGATACAT$G
CAT$GATTAGATA
GATACAT$GATTA
GATTAGATACAT$
T$GATTAGATACA
TACAT$GATTAGA
TAGATACAT$GAT
TTAGATACAT$GA
```

Self Indexing: The FM-index

Searching for GAT in GATTAGATACAT:

GAT

- **Count**(P) in **O(|P|)**
- **Locate**(P) in **O(|P| + occ)**

Applied in short read aligners such as **Bowtie** and **BWA** !

	F	L
0	$GATTAGATACAT	
1	ACAT$GATTAGAT	
2	AGATACAT$GATT	
3	AT$GATTAGATAC	
4	ATACAT$GATTAG	
5	ATTAGATACAT$G	
6	CAT$GATTAGATA	
7	GATACAT$GATTA	
8	GATTAGATACAT$	
9	T$GATTAGATACA	
10	TACAT$GATTAGA	
11	TAGATACAT$GAT	
12	TTAGATACAT$GA	

Count(GAT)=2

3rd to 5th occ of A

Self Indexing: Runs

The BWT can be quite repetitive:

BWT(GATTAGATACAT$) = TTTCGGAA$AATA

A **run** is a **substring of identical letters**, for example AAABBBA has 3 runs.

BWT(GATTAGATACAT$) = TTTCGGAA$AATA => 8 runs

Method	Space	Count(P)	Locate(P)
The Run-Length FM index **[Mäkinen, Navarro, Siren, and Välimäki.]**	**O(r)**	**O(|P|log|T|)**	**/ (not in O(r))**
The r-index **[Gagie, Navarro, and Prezza.]**	**O(r)**	**O(|P|)**	**O(|P|+occ)**

Indexing of a collection: read sets

TTAGA
TAGATA
GATTAGATACAT
GATTA ATACAT
GATAC

- Could we have a similar structure with space **depending on the number of runs** for an aligned readset?
- Can we find a link between the **number of runs of that similar structure for the readsets** and the **runs in the BWT of the genome** it is aligned to?

Indexing of a collection: Naive approach

Concatenate the reads with a separator “$” and build the FM-index of the entire string.

Example:

S = GATTA$TTAGA$TAGATA$GATAC$ATACAT$

BWT(S) = CATAATGTTTTTCGG$GAAAA$$AATAATA => 20 runs.

Issues:

- Computing the BWT for such a long string is challenging (and the context before the dollars is not relevant).
- The $ break some runs as in CGG$G in the example.

Indexing of a collection: The EBWT

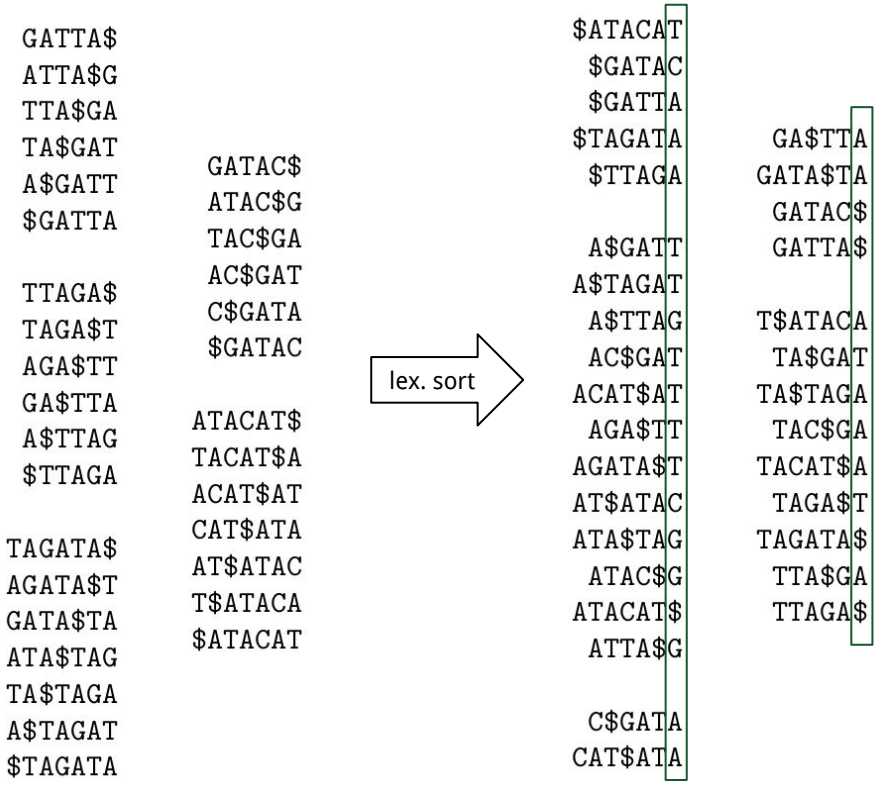
Extended BWT: lexicographically sort all rotations of each string.

[Mantaci, Restivo, Rosone, and Sciortino, An extension of the Burrows–Wheeler Transform, 2007]

Example:

EBWT(S) =
TCAAATTGTTTTCGG$GAAAA$$ATAAATA
=> 19 runs.

- Easier to build and update than the naive approach.
- Still has more than double the number of run than BWT(GATTAGATACAT).



Indexing of a collection: the heuristics on order

RLO: Reorganize the reads in the reversed lexicographic order (co-lexicographic order)

Example:

RLO(S) = {GATTA$, TTAGA$, TAGATA$, GATAC$, ATACAT$}
EBWT(RLO(S)) = AAACTGTTTTTTCGG$GAAAA$$AATAATA => 19 runs

SPRING:

[SPRING: a next-generation compressor for FASTQ data, Chandak, Tatwawadi, Ochoa, Hernaez, and Weissman, 2018]

Attempts to reorder reads according to their position in the genome.

EBWT(SPRING(S))= ACATATTGTTTTCGG$GAAAA$$ATAAATA => 22 runs

Indexing of a tree: The XBWT

eXtended BWT: Generalization of the BWT for labeled trees where nodes are sorted by their label from the node to the root and we output the outgoing edges.

XBWT(T) = ab abc $c $ $ $ a ac a a $ $

(Can be seen as a sub-case of a Wheeler graph.)

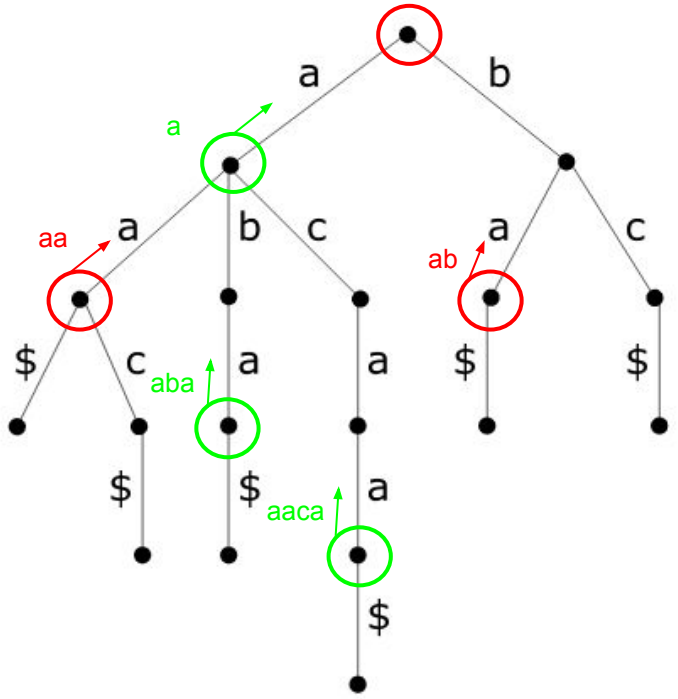


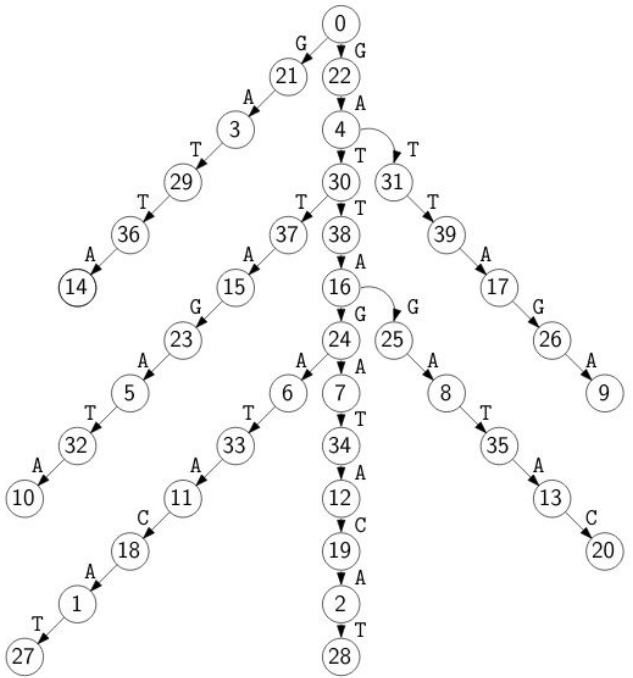
Figure: XBWT Tricks, *Giovanni Manzini*

Our approach: using alignment and XBWT

If the reads correspond to a known assembled genome, use the genome as additional context for better compression.

Example:

TTAGA
TAGATA
GATTAGATACAT
GATTA ATACAT
GATAC

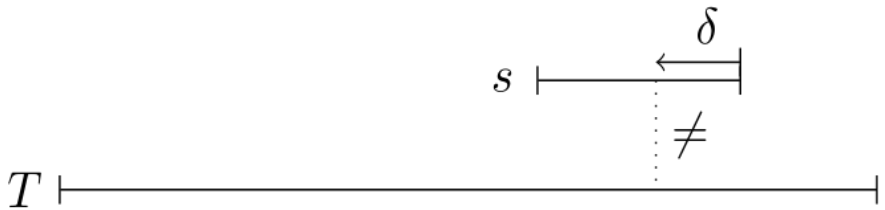


XBWT(T)=GGTTTTTTTTTCCCGGGGAAAAAAAATTTTAAAAAAAA => 7 runs.

Our approach: Theoretical guaranties

If we create such a tree, where the **reads are errorlessly sampled and aligned to the reference**, then the XBWT of the tree has the **same number of runs** as the BWT of the reverse of the reference.

But **in reality** the reads are **not perfectly matching** the assembled genome ..?



If the reads differ from the reference string and that the average **distance from first difference** (insertion, deletion, or substitution) **to the end** of the read is **δ**, then, the XBWT of the tree will have at most **2δ additional runs** per reads.

Our approach: Theoretical guaranties

If we create a labeled tree T as explained, let:
r be the number of runs in the XBWT,
t be the number of reads,
Then in **O(r+t)** words of space,
We can:
Count(P) in **O(|P|log log(|T|))**
Locate(P) in **O((|P|+occ)log log(|T|))**

What are the runs in the XBWT in practice ? And how do we build the XBWT ?

For scalability: prefix-free parsing construction

The reference and genomes are typically tens of Gb of data…
Consequently, **algorithms designed to work in RAM may not be practical**.
We chose to adapt a previous technique **Prefix free parsing.**

Prefix free construction takes advantage of the **highly repetitive** nature of genomic databases using **context-triggered piecewise hashing**.
[Boucher, Gagie, Kuhnle, Langmead, Manzini and Mun, Prefix-free parsing for building big BWTs, WABI 2017]

[PFP Data Structure, Boucher, Cvacho, Gagie, Holub, Manzini, Navarro, Rossi, 2020]
54 GB peak memory for 1000 variants of human chromosome 19, initially occupying roughly **56 GB**.

To aim for a scalable structure **we adapted Prefix free parsing to build the XBWT**.

Experiments: comparison to the state of the art

A preliminary comparison only on the number of runs for now.

We compared to:

- **EBWT** (using the ropebwt2 implemetation), with and without $.
- **SPRING + EBWT,** with and without $.
- **RLO + EBWT,** with and without $.

Removing the $ reduced the number of runs between **2.7%** and **29.2%**.

Experiments: datasets and protocol

Only reads matched to the genome, not to the reverse complement.

Reads and corresponding genome:

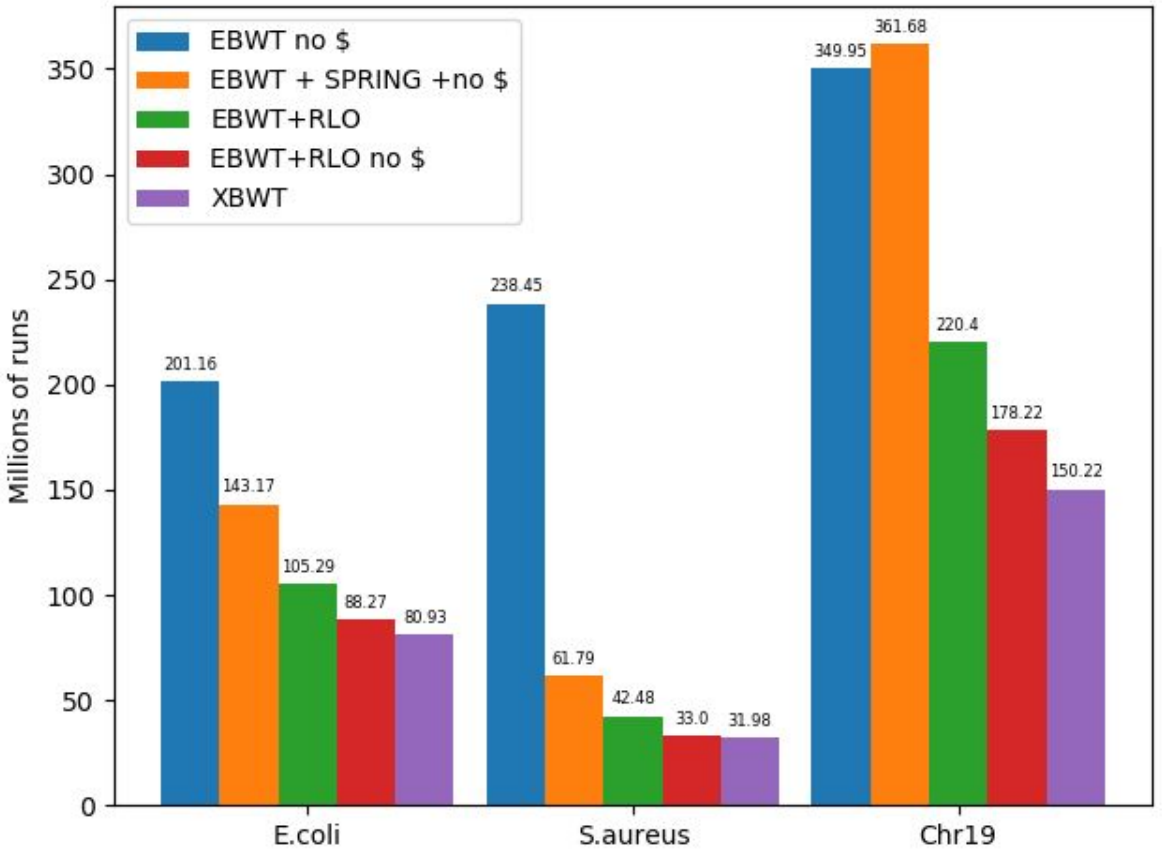
- **E.coli**: from the single cell dataset
- **S.aureus**: from the single cell dataset
- **R.sphaeroides**: from the Gage-b dataset

Reads aligned to a reference genome:

- **Chr19**: a reference genome and the reads of a HiSeq 2000 readsets that aligned

bwa-mem used to align the reads to the genomes.

Experiments: results



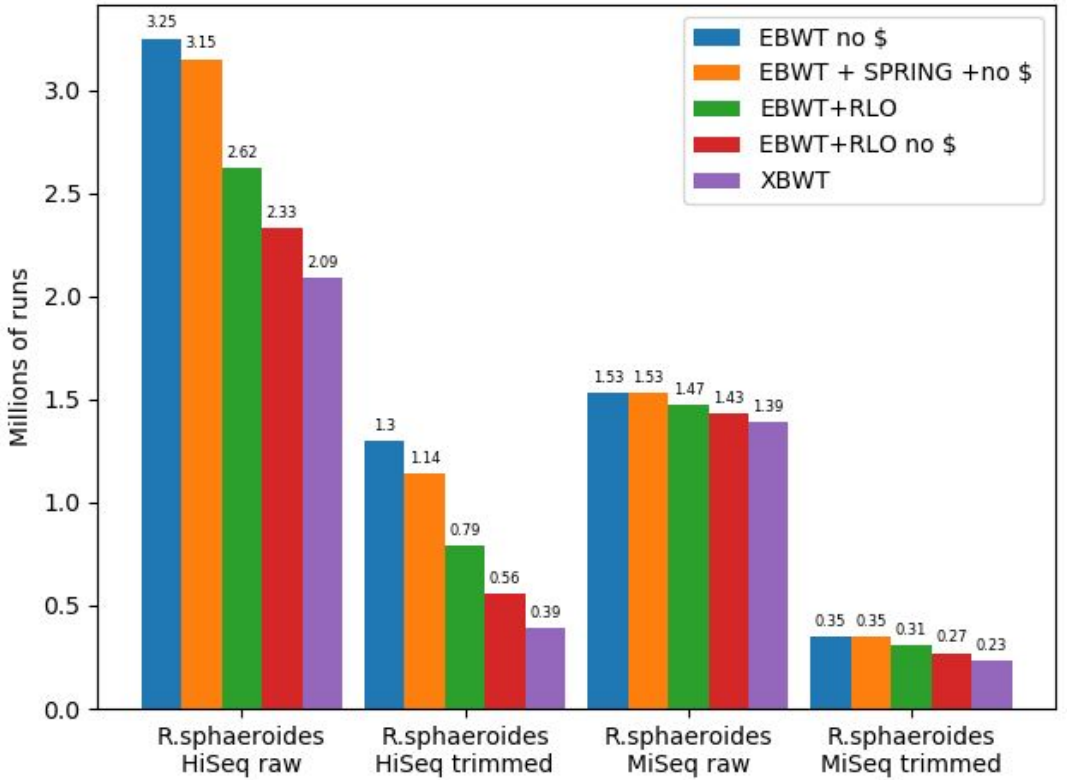
Dataset	Nb. reads	Coverage	δ	errorless reads
E.coli	14M	304×	13	57%
S.aureus	26M	927×	7	88%
Chr19	34M	57×	15	71%

- RLO+EBWT has much less runs than the plain EBWT.
- **XBWT performs best (but RLO is close)**
- On S.aureus, RLO, RLO no $ and XBWT are very close.

Experiments: Gage-b R.sphaeroides

Dataset	Number of reads	Read length	Coverage
HiSeq			
raw	166 820	101	46×
trimmed	134 207	up to 101	37×
MiSeq			
raw	23 102	251	24×
trimmed	20 046	up to 251	20×

Dataset	δ	Errorless reads	Error rate
HiSeq			
raw	27	31.34%	0.04%
trimmed	6	83.26%	0.01%
MiSeq			
raw	122	0.25%	0.15%
trimmed	29	63.55%	0.03%



Summary of our contributions:

- Looking at the genome for additional context for better compression is worth investigating !
  - We provide theoretical time and space guaranties depending on the number of reads and the number of runs.
  - We show an upper bound on the number of runs depending on the errors in the reads compared to the genome.
  - The experimental number of runs is comparatively small.
- Prefix-free construction of the BWT can be adapted for the XBWT.
- A similar approach could be used to improve the space usage of the hybrid index. (Not explored in this talk)

//TODO:

- Larger scale analysis (on human genome, on long reads)
- FM-index, implementation and time analysis
- Time comparison of PFP construction of the XBWT compared to other construction

  [BWT-tunneling by Uwe Baier, Wheeler sort by Jarno Alanko]