String Partition for Building Long Burrows-Wheeler Transforms

Enno Adler, Stefan Böttcher, Rita Hartel
Paderborn University

Enno Adler
Prag, 25.08.2025

Why is the BWT important?

- Main component of FM-Index:
  - text compression, indexing, pattern search
- Key to many applications in bioinformatics:
  - *de novo* assembly and read alignment
  - BWA, Bowtie2, MICA, and SOAP2

Existing Approaches

$$S \xrightarrow{\text{single-string construction}} BWT(S)$$

$$W = [W_0; \ldots; W_k] \xrightarrow{\text{multi-string construction}} BWT(W)$$

Existing Approaches

$$
\begin{tikzcd}
S \arrow[d, "\text{single-string construction}"] & W = [W_0; \ldots; W_k] \arrow[l, "S := W_0\$_0 W_1\$_1 \ldots W_k\$_k"'] \arrow[d, "\text{multi-string construction}"] \\
BWT(S) \arrow[r, "="] & BWT(W)
\end{tikzcd}
$$

- single-string construction: divsufsort, libsais, grlBWT, eGap, ropebwt3, gsufsort, BigBWT, and r-pfbwt

Existing Approaches

$$
\begin{tikzcd}
S \arrow[r, "k = 0; W_0 = S"] \arrow[d, "\text{single-string construction}"] & W = [W_0; \ldots; W_k] \arrow[d, "\text{multi-string construction}"] \\
BWT(S) & BWT(W) \arrow[l, "="']
\end{tikzcd}
$$

- single-string construction: divsufsort, libsais, grlBWT, eGap, ropebwt3, gsufsort, BigBWT, and r-pfbwt
- multi-string construction: BCR, ropebwt, ropebwt2, and IBB

Our Approach

$$
\begin{tikzcd}
S \arrow[r, "\textbf{partition (our result)}"] \arrow[d, "\text{single-string construction}"] & W = [W_0; \ldots; W_k] \arrow[d, "\text{multi-string construction}"] \\
BWT(S) & BWT(W) \arrow[l, "\text{our result}"']
\end{tikzcd}
$$

Key takeaway:

partDNA + IBB is pareto-optimal (time-RAM trade-off) for $BWT(S)$ construction.

A Second Motivation

BWT Construction using Suffix Arrays

?

BWT Construction using LF-Mapping

A Second Motivation

$k = n$, BWT Construction using Suffix Arrays

$k = 2$ as value for the following example

$k = 0$, BWT Construction using LF-Mapping

Multi-String and Single-String BWTs

BWT([AT$, ATGC$, C$])

T ← last char of 1st word AT$
C ← last char of 2nd word ATGC$
C ← last char of 3rd word C$
$
$
G
$
T
A
A

Multi-String and Single-String BWTs

BWT([AT$, ATGC$, C$])

T ← last char of 1st word AT$
C ← last char of 2nd word ATGC$
C ← last char of 3rd word C$
$
$
G
$
T
A
A

BWT(CATGCAT$)

T ← 1st smallest suffix
C ← 2nd smallest suffix
C ← 3rd smallest suffix
G
$
T
A
A

Idea of Partition

- Last chars of word i is at position i in $BWT(S)$ and $BWT(W)$
- Chars to the left until next colored char belong to the same word

BWT(C ATG C A T $)

T ← last char of 1st word
C ← last char of 2nd word
C ← last char of 3rd word
G
$
T
A
A

Idea of Partition

- Last chars of word i is at position i in $BWT(S)$ and $BWT(W)$
- Chars to the left until next colored char belong to the same word

BWT(C ATGC AT $)

T ← last char of 1st word
C ← last char of 2nd word
C ← last char of 3rd word
G
$
T
A
A

Number and Length of Words

- Last chars of word *i* is at position *i*
- Chars to the left until next belong to the word

→ Number of words is the number of smallest suffixes

→ Length of words depends on the number of smallest suffixes

BWT(C ATGC AT $)

T ← last char of 1st word
C ← last char of 2nd word
C ← last char of 3rd word
...

BWT(CATGC AT $)

T ← last char of 1st word
C ← last char of 2nd word
C
...

Equality of Construction

BWT([AT$, ATGC$, C$])

T
C
C
$
$
G
$
T
A
A

BWT(C ATGC AT $)

T ← 1st smallest suffix
C ← 2nd smallest suffix
C ← 3rd smallest suffix

G
$
T
A
A

Towards a Procedure

- Take k smallest suffixes of S
- Partition S at these positions
- W_i is the word before the ith smallest suffix

Towards a Procedure

- Take k smallest suffixes of S
- Partition S at these positions
- W_i is the word before the ith smallest suffix

- Find the position of smallest suffixes
- Sort the resulting words accordingly

On Sorting the Words

Chars before	Sorted Suffixes
C ATGC AT	$
C ATGC	AT $
C	ATGC AT $

- Suffixes are composed of words W_i
- Name words by their rank (W_1 is the smallest word $\rightarrow$ 1)

Chars before	Sorted Suffixes
W_3 W_2 W_1	$
W_3 W_2	W_1 $
W_3	W_2 W_1 $

Chars before	Sorted Suffixes
W_3 W_2 W_1	$
W_3 W_2	1 $
W_3	2 1 $

partDNA

1. Split S before runs with at least h A symbols and before $A^*\$$ into words S_j
2. recursively Bucket-sort the S_j
3. Resolve order of equal S_j by small SA construction
4. Obtain W from S_j by induced suffix sorting

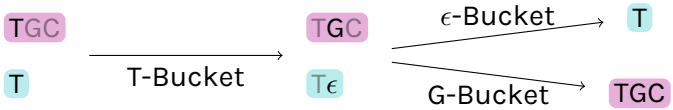
- Find the position of smallest suffixes
- Sort the resulting words accordingly

1. Split S

1. Split S before runs with at least h A symbols and before $A^*\$$ into words S_j

- $S = \$\,C\,A\,TGC\,A\,T$
- $h = 1$
- $S_0 = C$, $S_1 = TGC$, and $S_2 = T$

2. The Recursive Bucket Sort



Stop if leave is of size 1 or if ϵ-bucket

Sort within ϵ-bucket according to number of following As in $S = \$\,C\,A\,TGC\,A\,T$

The order is top-down the leaves: $\$C < AT < ATGC$

Assign names to words: $\$C \mapsto 1$, $AT \mapsto 2$, and $ATGC \mapsto 3$

3. Small SA Construction

$$S = \$\ S_0\ \ A\ S_1\ \ A\ S_2$$

$$R = 1\ \ 2\ \ 3$$

$$SA(R) = 123$$

$$\$C \mapsto 1$$

$$AT \mapsto 2$$

$$ATGC \mapsto 3$$

4. Induce the Words

$S = \$\ S_0\ \ A\ S_1\ \ A\ S_2\ \$$

$SA(R) = 123$

Smallest suffixes:

1. $\$$ / $\$\ S_0$ S_1 $A\ S_2$

Word before the suffix:

- $A\ S_2$

4. Induce the Words

$S = \$\ S_0\ \ A\ S_1\ \ A\ S_2\ \$$

$SA(R) = 123$

Smallest suffixes:

1. $\$ / $\$\ S_0$ S_1 $A\ S_2$

3. $A\ S_2$

Word before the suffix:

- $A\ S_2$
- $A\ S_1$

4. Induce the Words

$S =$ \$ S_0 A S_1 A S_2 \$

$SA(R) = 123$

Smallest suffixes:

1. \$ / \$ S_0 S_1 A S_2
2. A S_1 A S_2
3. A S_2

Word before the suffix:

- A S_2
- \$ S_0
- A S_1

4*. Induce a Complex Word

Smallest suffixes:

1. $

Word before the suffix:

- A S_2

$S = \$\ S_0\ AA\ A\ S_1\ A\ A\ S_2\ \$$

$SA(R) = 123$

4*. Induce a Complex Word

$S = \$\ S_0\ AA\ A\ S_1\ A\ A\ S_2\ \$$
$SA(R) = 123$

Smallest suffixes:

1. $\$$
5. $A\ S_1\ A\ A\ S_2$
6. $A\ S_2$

Word before the suffix:

- $A\ S_2$
- A
- A

4*. Induce a Complex Word

$S = \$\ S_0\ AA\ A\ S_1\ A\ A\ S_2\ \$$
$SA(R) = 123$

Smallest suffixes:

1. $\$$
3. A $A\ S_1$ A $A\ S_2$
4. A $A\ S_2$
5. $A\ S_1$ A $A\ S_2$
6. $A\ S_2$

Word before the suffix:

- $A\ S_2$
- A
- $A\ S_1$
- A
- A

4*. Induce a Complex Word

$S =$ \$ S_0 AA A S_1 A A S_2 \$

$SA(R) = 123$

Smallest suffixes:

1. \$
2. AA A S_1 A A S_2
3. A A S_1 A A S_2
4. A A S_2
5. A S_1 A A S_2
6. A S_2

Word before the suffix:

- A S_2
- \$ S_0
- A
- A S_1
- A
- A

partDNA

1. Split S before runs with at least h A symbols and before $A^*\$$ into words S_j
2. recursively Bucket-sort the S_j
3. Resolve order of equal S_j by small SA construction
4. Obtain W from S_j by induced suffix sorting

Datasets

datasets	GRCh38		JAGHKL01	
	word count	word length	word count	word length
original	1	> 3 billion	1	> 14 billion
partition at *AAAAA*	> 20 million	151	> 39 million	364
partition at *AAAA*	> 46 million	67	> 118 million	121
partition at *AAA*	> 116 million	26	> 356 million	40

Approaches

S

single-string construction (black bullet)

- libsais
- divsufsort

...

$BWT(S)$

S

partition (shape)

△ AAA

□ $AAAA$

W

multi-string construction (color)

IBB

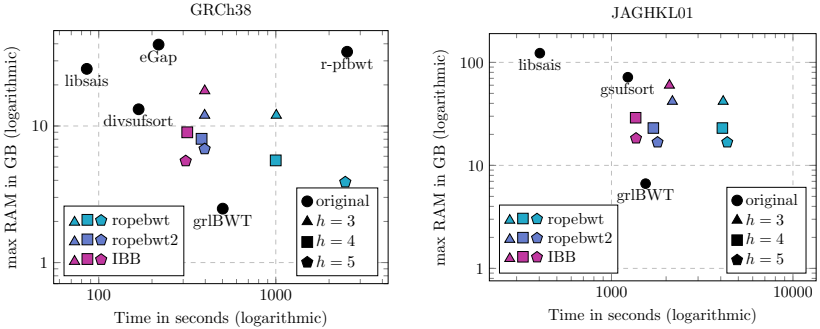
...

$BWT(W)$

delete \$

$BWT(S)$

Construction Time and RAM Usage



Key takeaway:

partDNA + IBB is pareto-optimal (time-RAM trade-off) for $BWT(S)$ construction.

PADERBORN UNIVERSITY

Thank you very much for your attention.

I look forward to an exciting discussion.

Enno Adler | eadler@mail.upb.de