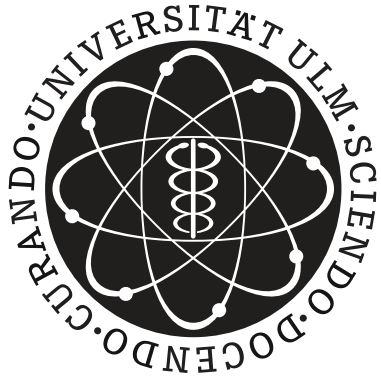
BWT Tunneling

Uwe Baier



Ulm University

Institute of Theoretical Computer Science

Ulm

June 28, 2019

Data Compression

Why should we compress our data?...

...people from informatics should know best themselves...

- age of “big data”, transfer and storage are expensive
- not every method can be implemented using streaming, memory is an even more limited resource
- some compressed representations allow methods of sequence analysis to be performed much faster

In this talk

BWT Tunneling Theory

Application in Data Compression

Application in Sequence Analysis

Burrows Wheeler Transformation

History

- reversible text transformation [Burrows and Wheeler, 1994]
- former application: data compression

Today

- rare usage in data compression, e.g. `bzip2` [Seward, 1996]
  problem: slow decompression speed
- various applications in bioinformatics,
  e.g. `BWA` for read alignment [Li and Durbin, 2010]
- usage in compressed full text indexing,
  e.g. FM-Index [Ferragina and Manzini, 2005]

Future

- advances in full text indexing [Gagie et al., 2018]
- advances in compression rates [Baier, 2018]

BWT – What is it?

"The BWT L *is a string generated by concatenating all cyclic preceding characters of the lexicographically sorted suffixes of a string S."*

BWT generation of $S =$ `easypeasy$`

prec. char.	suffixes
y	$
s	y$
a	sy$
e	asy$
p	easy$
y	peasy$
s	ypeasy$
a	sypeasy$
e	asypeasy$
$	easypeasy$

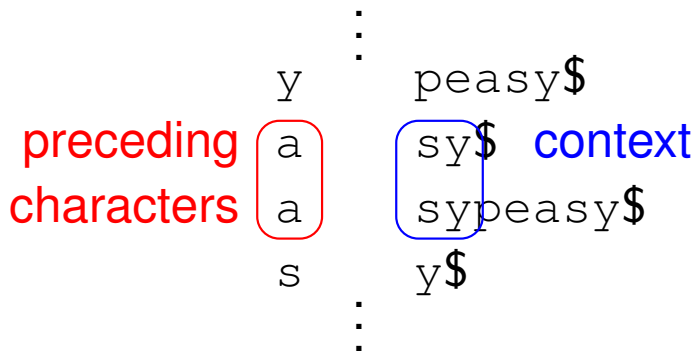
sort →

L	sorted suffixes
y	$
e	asy$
e	asypeasy$
p	easy$
$	easypeasy$
y	peasy$
a	sy$
a	sypeasy$
s	y$
s	ypeasy$

BWT - use in data compression

Similar contexts tend to be succeeded (or preceded) by similar characters

- BWT places characters preceding the same context near to each other



- character distribution of small portions of BWT is skew
- compression e.g. by transforming local to global skewness + entropy coding or by run-length-encoding

Similar contexts tend to be succeeded (or preceded) by similar strings

- compression by tunneling

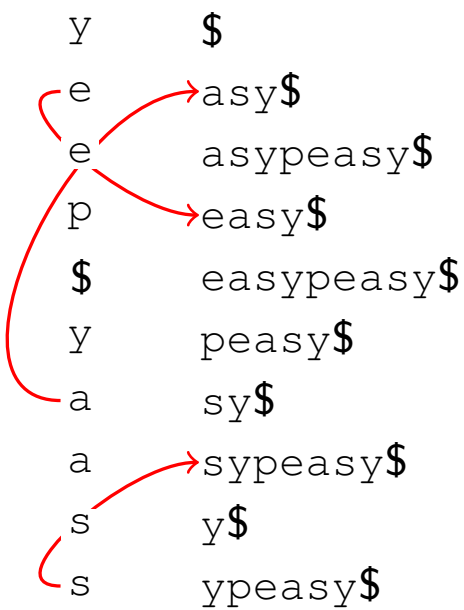
BWT - backward step

LF-mapping

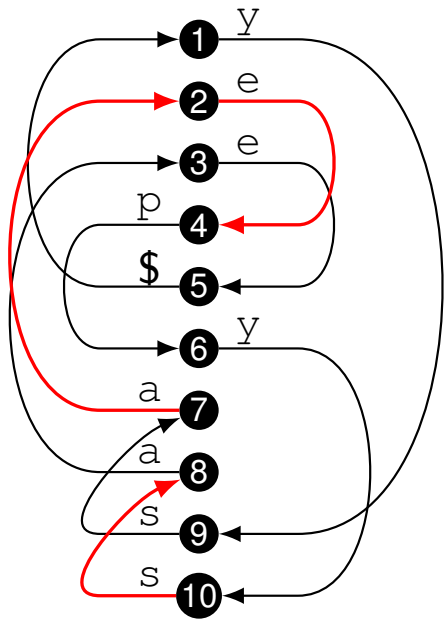
Given a position i of a sorted suffix ω with BWT entry $c = \mathrm{L}[i]$,
LF[i] points to the position j of the sorted suffix $c\omega$

- LF-mapping can solely be computed using L
- a walk through L using LF-mapping yields reverse original string
- LF-mapping and L can be visualized in a graph [Gagie et al., 2017]

BWT



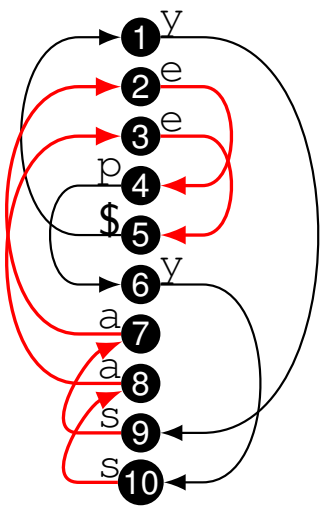
Wheeler graph



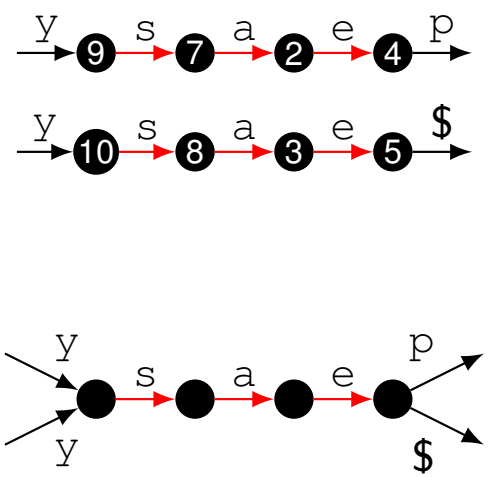
Tunneling

- parallel equally labeled paths (called prefix interval) can be contracted to a tunnel
- original Wheeler graph can be emulated
  - when entering a tunnel, save offset to uppermost entry edge
  - when leaving a tunnel, use offset to jump back to “correct lane”
- can be performed iteratively

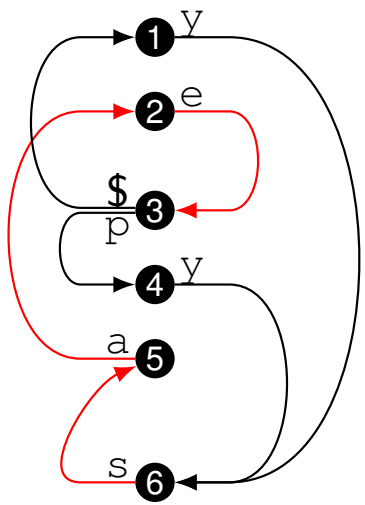
Wheeler graph



prefix interval



Tunneled graph

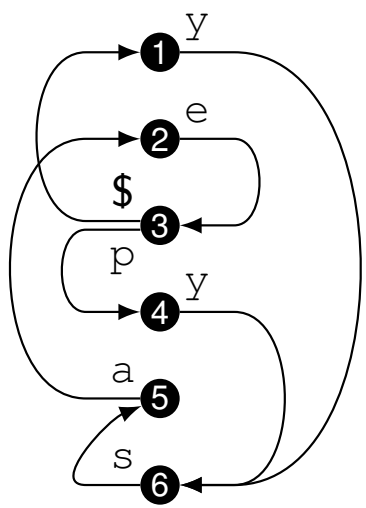


Succinct graph representation

- concatenate unary encoded $\begin{smallmatrix}\text{indegree}\\ \text{outdegree}\end{smallmatrix}$ of each node to bitvectors $\begin{smallmatrix}D_{\text{in}}\\ D_{\text{out}}\end{smallmatrix}$
- L ist obtained by concatenating edge labels from top to bottom

Tunneled graph

D_{out}	outdeg.
1	1
1	1
0	2
1	1
1	1
1	1
1	



indeg.	D_{in}
1	1
1	1
1	1
1	1
1	1
2	0
	1

Succinct representation

D_{out}	L	D_{in}
1	y	1
1	e	1
0	$	1
1	p	1
1	y	1
1	a	0
1	s	1

Tunneling recap

- tunneling contracts prefix intervals to just one prefix

Suffixes and prefixes

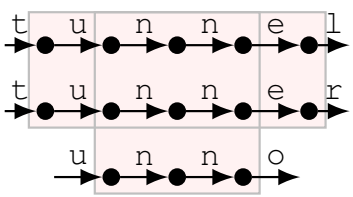
prefixes	sorted suffixes
easypeasy	$
easype	asy$
~~e~~	asypeasy$
easyp	easy$
$	easypeasy$
easy	peasy$
easypea	sy$
~~ea~~	sypeasy$
easypeas	y$
~~eas~~	ypeasy$

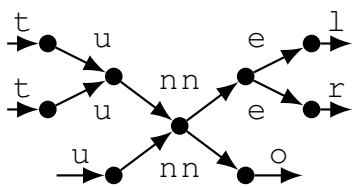
Tunneled BWT

L	D_{out}	D_{in}
y	1	1
e	1	1
$	0	1
p	1	1
y	1	1
a	1	0
s	1	1

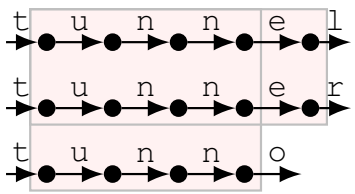
- prefix intervals are allowed to overlap (“cross-overlay”)

Compensable collision





Critical collision



Theory overview

- tunneling discovery for normal BWT [Baier, 2018]
  - correctness, overlappings, computation of parallel paths
  - use in data compression, hints for sequence analysis
- extension to Wheeler graphs [Alanko et al., 2019]
  - other BWT-based data structures can be tunneled (de Bruijn graphs, tries, wavelet trees, . . .)
  - generalization of prefix intervals, sampling scheme for tunneled FM index
  - Capocelli prize for best student paper

... end of story, let's dig tunnels? ...

- tunneling planning is difficult [Baier and Dede, 2019]
  - hard to compute exactly
  - hard to approximate

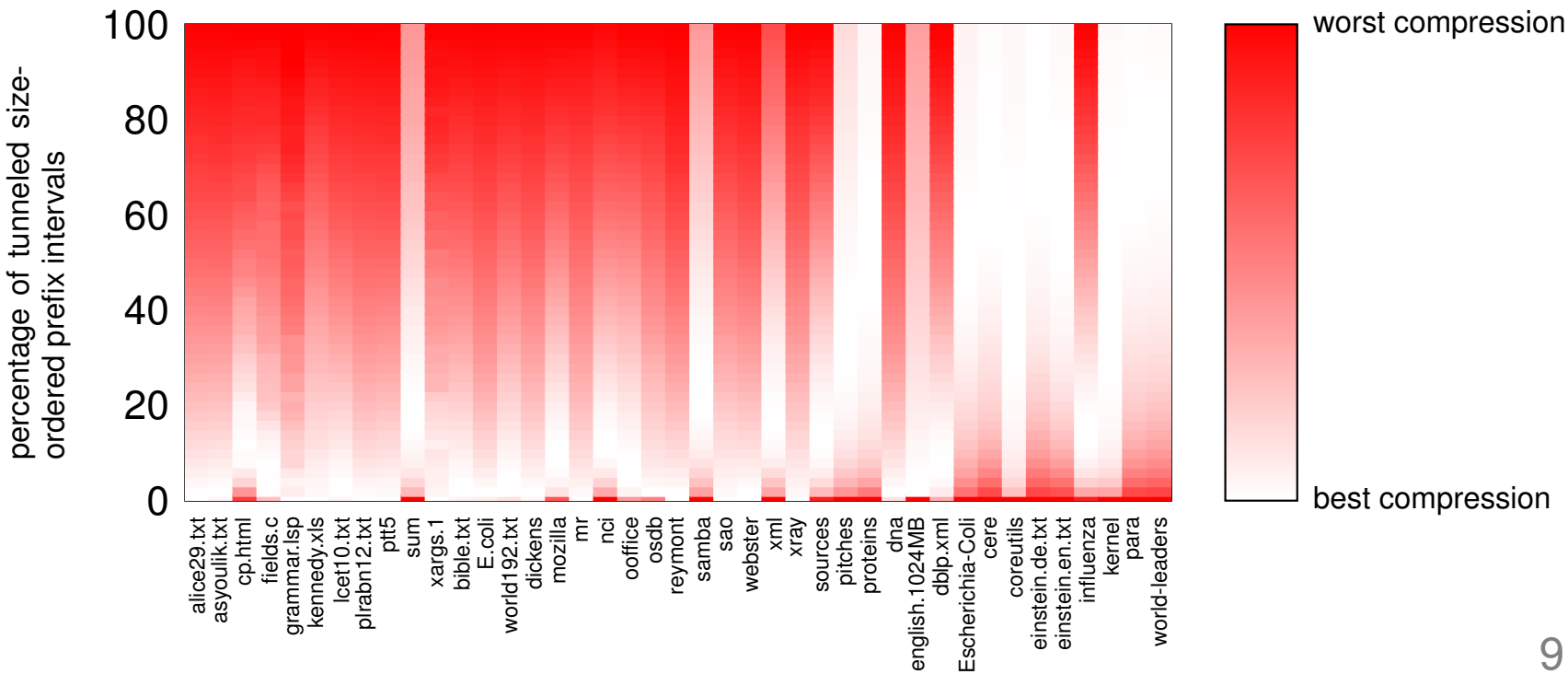
... so how to do it? ...

BWT data compression challenges

- BWT compression research since 25 years with more or less advance
- oldest approach by Burrows & Wheeler still is very competitive
- unlikely that BWT compressors can be improved much

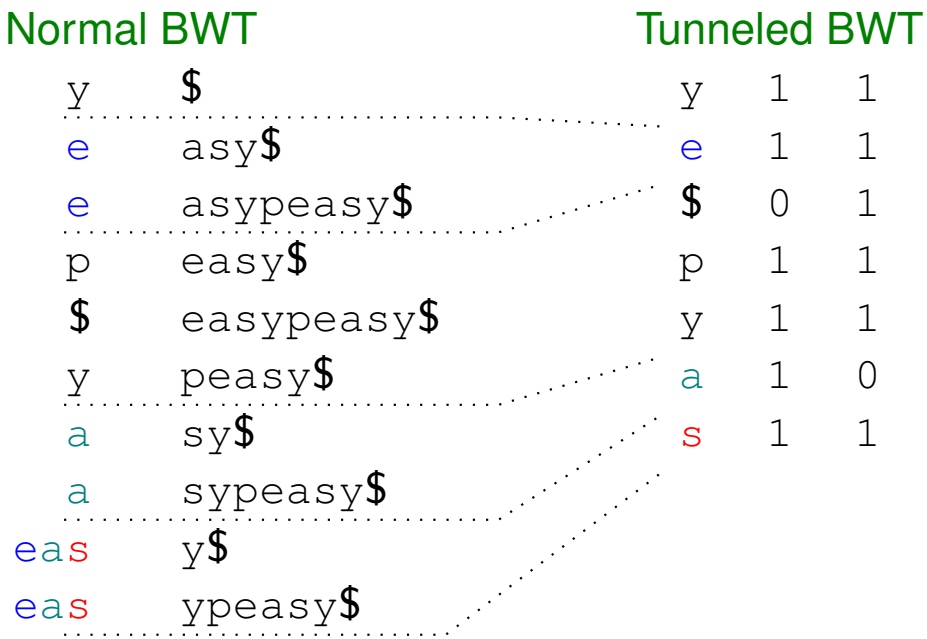
How to integrate tunneling for improvements?

- tunneling should be an interim stage
- tunneling should enhance compressors, not replace them
- target is to **minimize encoding size, not BWT length**



Tunneling and BWT runs

- compressibility of BWT heavily depends on BWT runs
  $\Rightarrow$ run-length-encoding $\cdots \underbrace{\texttt{aaaaa}}_{5 \text{ times}} \cdots \Rightarrow \cdots \underbrace{\texttt{a01}}_{5 = (101)_2} \cdots$
- columns of a prefix interval are subsequences of BWT runs
- tunneling shortens length of BWT runs



Tunnel engineering

Run-terminated prefix intervals

- leftmost and rightmost column of interval is a full run
- overlappings are always compensable
- additional bitvectors can be compactified to number of runs

Abstract cost model

- uses run-length coding and entropy coding as basis
- prefix intervals have an individual score but uniform cost
- validated with 3 different BWT backend encoders

Difficulties with run-length encoding

- logarithmic scale
- negative side effects (overlapping prefix intervals)
- positive side effects (non-overlapping prefix intervals passing through same run)

Tunnel planning strategies

Greedy strategy

- chooses prefix intervals in a greedy fashion
- considers negative side effects by updating benefits of not-yet chosen intervals
- final choice: intervals whose benefit overcomes their costs
- optimal without run-length encoding

Hirsch strategy

- the more tunnels are used, the less is the cost per tunnel
- compute for each prefix interval the number of tunnels required so that benefit of interval overcomes tunnel costs
- choose as many intervals as possible such that each intervals benefit overcomes the costs
- no side effects are considered

Experiments overview

BWT compressors enhanced with tunneling

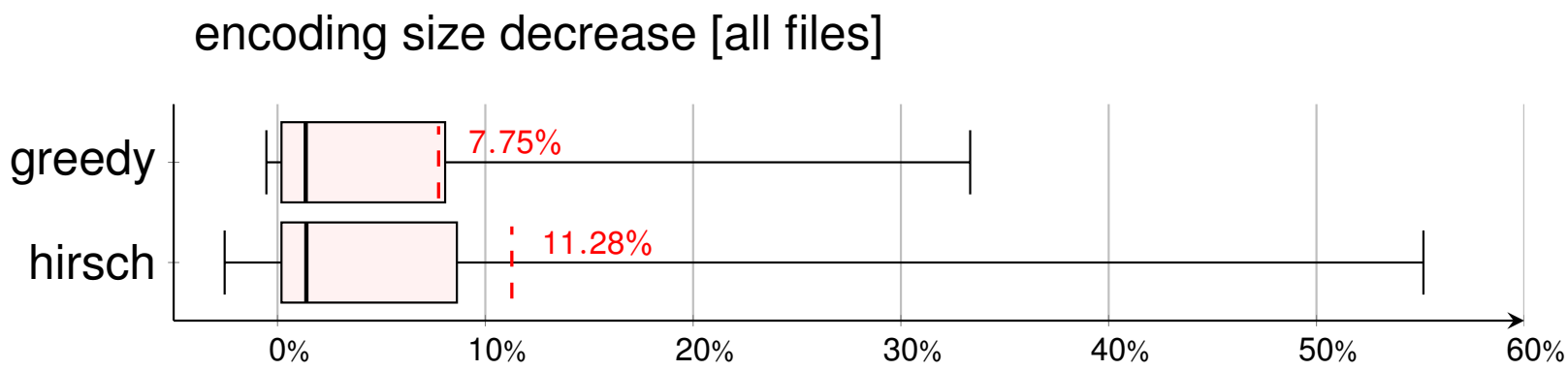
- `bwz`: original scheme by Burrows & Wheeler (≈ `bzip2`)
- `bcm`: one of the best open-source BWT compressors
- `wt`: wavelet tree using hybrid bitvectors

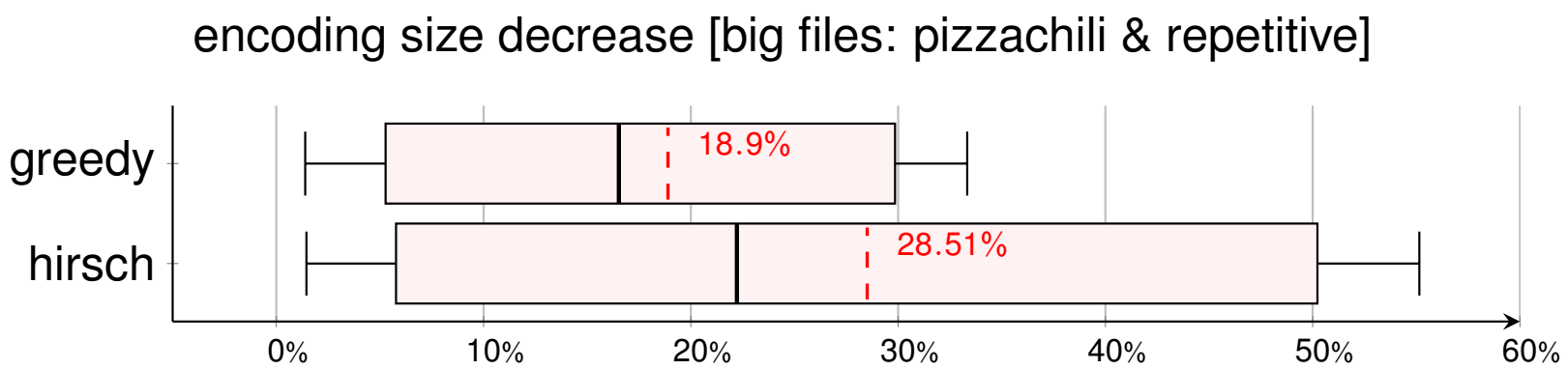
Test data

Corpus	#Files	Filesizes (MB)
Canterbury	11	0.003 - 1
Large Canterbury	3	2 - 5
Silesia	12	6 - 49
Pizza & Chili	6	54 - 1130
Repetitive	9	45 - 446

Tunneling improvements

- Comparison with normal BWT compression
- BWT backend encoder: `bcm`





Comparison with other data compressors

- `xz`: uses LZMA, similar to `7-zip`
- `zpaq`: uses context mixing
- all values are measured in bits per symbol

Compressor	Silesia Corpus			Pizza & Chili Corpus			Repetitive Corpus		
	dickens (10 MB)	samba (21 MB)	webster (40 MB)	proteins (1130 MB)	dna (386 MB)	english (1024 MB)	cere (440 MB)	coreutils (196 MB)	worldleaders (45 MB)
`bcm`	1.76	1.49	1.24	2.33	1.72	1.56	0.24	0.23	0.13
`tbcm-greedy`	1.75	1.42	1.24	1.95	1.70	1.34	0.17	0.16	0.11
`tbcm-hirsch`	**1.75**	1.41	1.24	**1.89**	**1.70**	**1.33**	0.11	**0.13**	0.09
`xz`	2.22	1.38	1.61	2.22	1.78	1.93	**0.09**	0.14	**0.09**
`zpaq`	1.78	**1.20**	**1.21**	2.61	1.86	1.64	1.77	0.62	0.09

Tunneling and data compression overview

- initial compression results [Baier, 2018]
  - run-terminated prefix intervals and compensable overlappings
  - abstract cost model, greedy strategy
  - mentioned side effects
- compression improvements
  - predecessor of hirsch strategy
    [Dede, 2018, student project], [Baier and Dede, 2019]
  - hirsch strategy [Räther, 2019, student project]
- no further compression results published so far

Tunneling and sequence analysis

- various text-based data structures can be described by Wheeler graphs
  - de Bruijn graphs
  - Tries
  - Wavelet trees
  - ...

  $\Rightarrow$ data structures can be represented succinct using BWT variants
- tunneling reduces the size of BWT variants $\Rightarrow$ higher succinctness level

Challenges

- size of BWT variants should be minimized
- considered prefix-intervals should not overlap
- tunneling should be integrated into BWT variants and not being added on top
- tunneled data structures should keep their full functionality

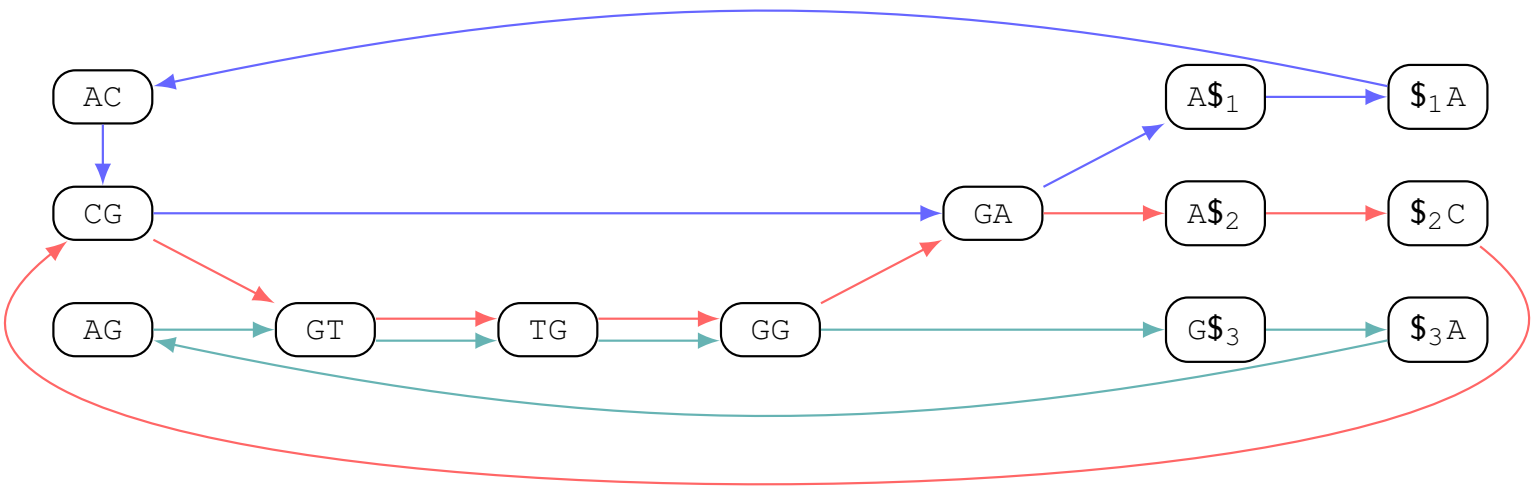
De Bruijn graphs

A de Bruijn graph is a multigraph

- nodes are all k-mers (length k substrings) of a cyclic string S
- nodes are joint by an edge if the two k-mers overlap by $k-1$ characters in the cyclic string S
- multiple de Bruijn graphs can be combined

Combined de Bruijn graph ($k = 2$)

$S_1 = \texttt{ACGA\$}_1$, $S_2 = \texttt{CGTGGA\$}_2$, $S_3 = \texttt{AGTGG\$}_3$



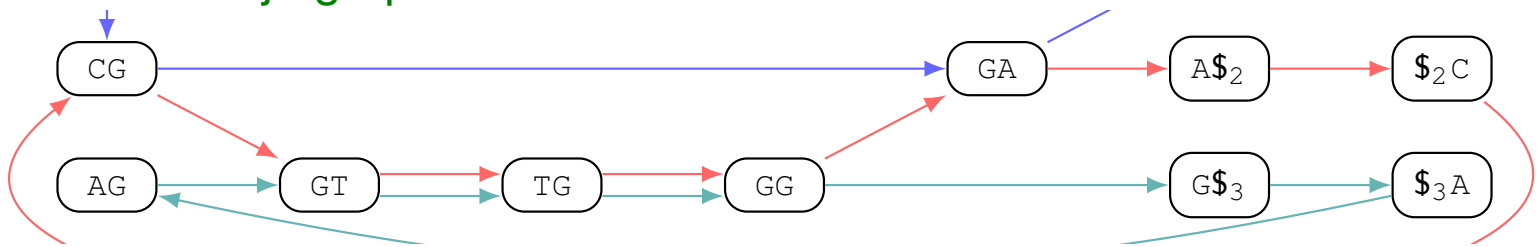
Edge reductions

Let G be a de Bruijn graph and u and v be two nodes fulfilling

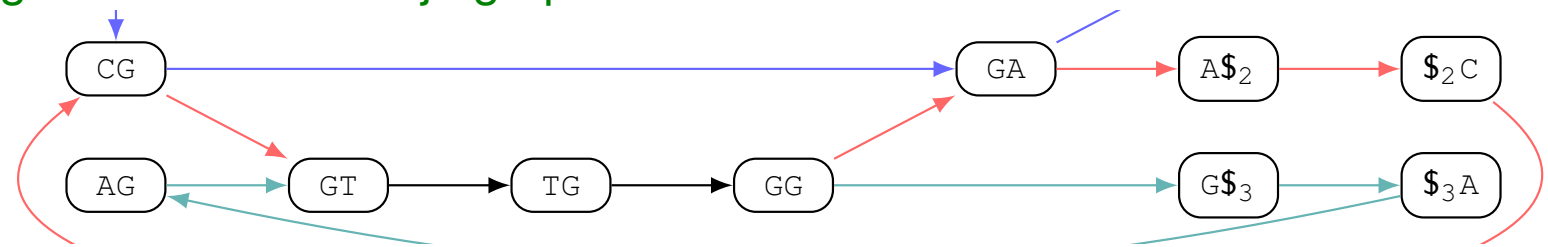
1. u is the only predecessor of v
2. v is the only successor of u

Then all edges from u to v can be contracted to just one edge

Normal de Bruijn graph



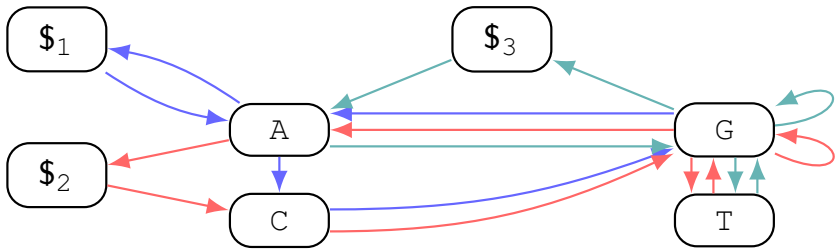
Edge-reduced de Bruijn graph

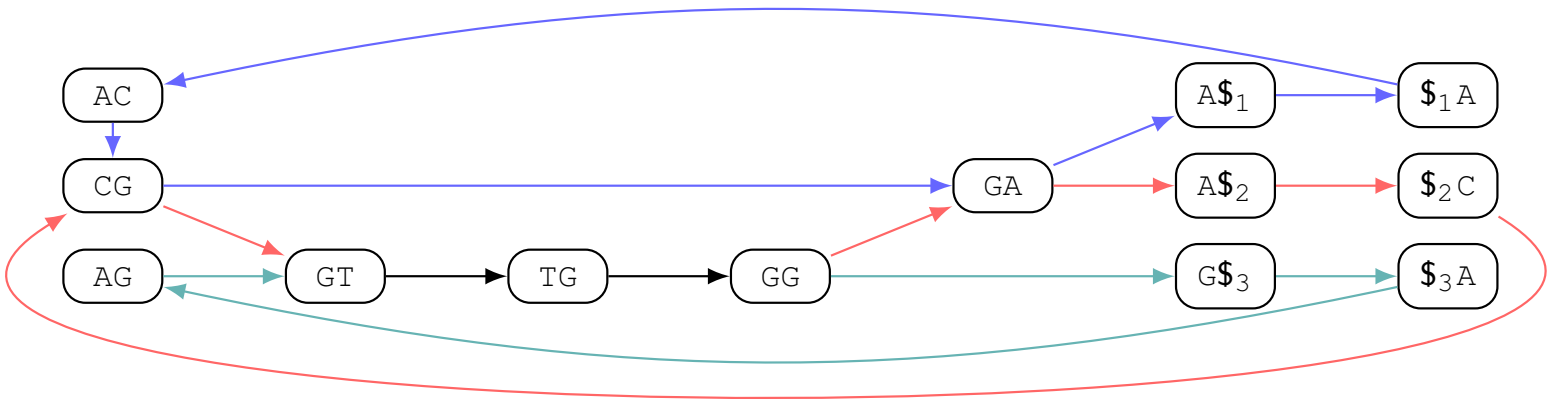


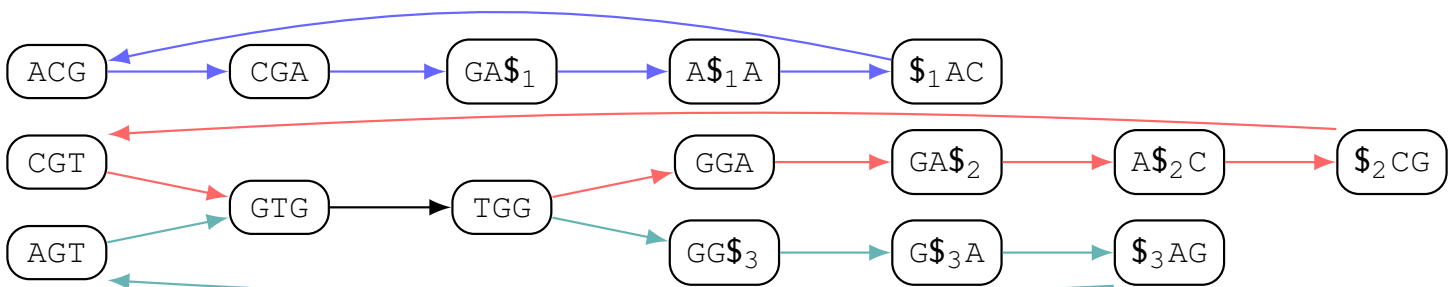
$\Rightarrow$ edge reduction corresponds to tunneling of k-mer prefix intervals

Edge minimization problem

Find order k s.t. edge reduced de Bruijn graph has minimal number of edges
$\Rightarrow$ minimizes size of tunneled BWT w.r.t k-mer prefix intervals

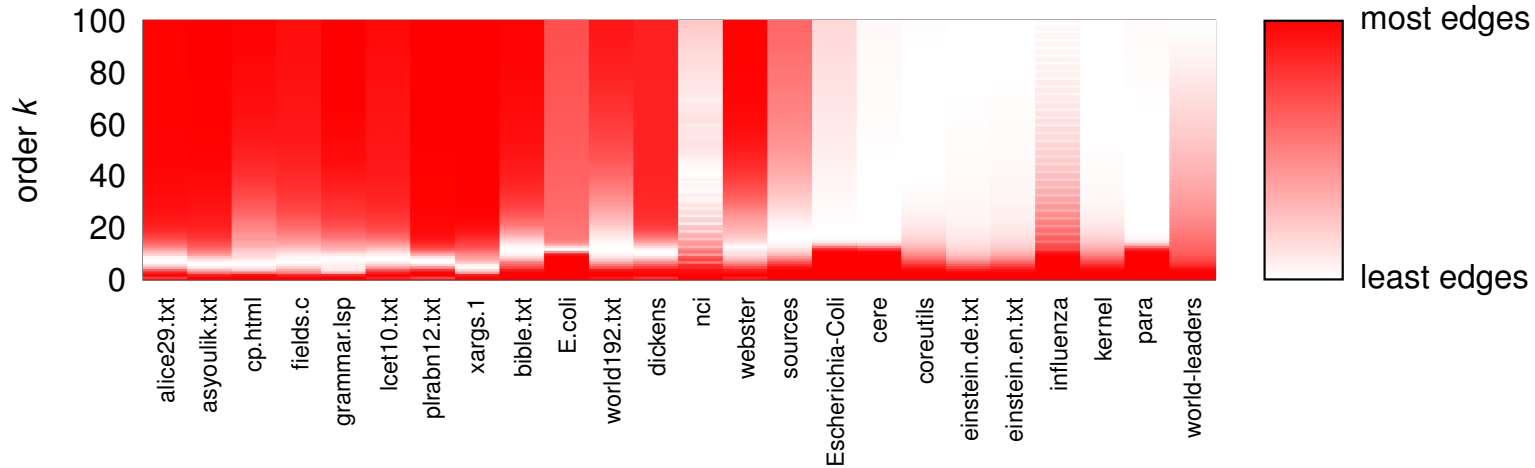




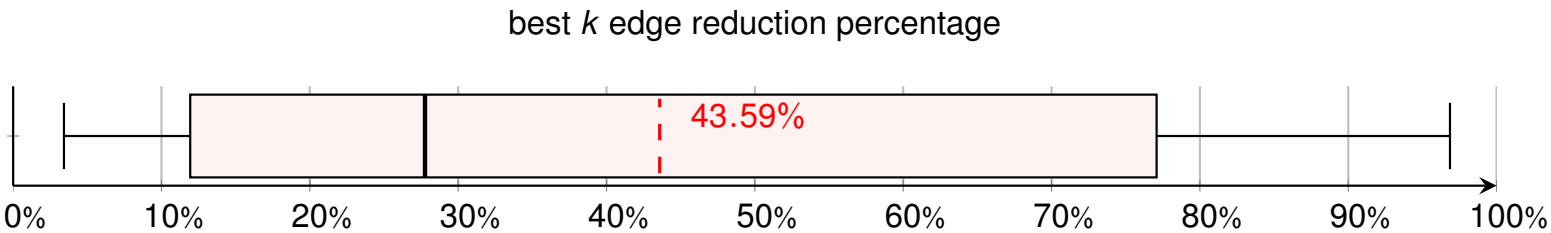


Edge minimization problem

- problem is not trivial



- problem can be soluted using an outputsensitive linear-time algorithm
- amount of edge reductions is big $\Rightarrow$ good BWT compactification

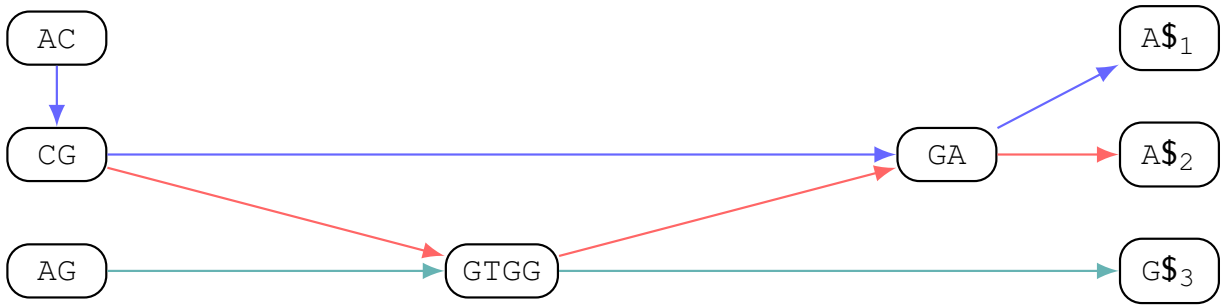


Purpose #1: Compressed de Bruijn graphs

- acyclic de Bruijn graph
- merges nodes instead of edge reduction

Compressed de Bruijn graph ($k = 2$)

$S_1 = \texttt{ACGA\$}_1$, $S_2 = \texttt{CGTGGA\$}_2$, $S_3 = \texttt{AGTGG\$}_3$



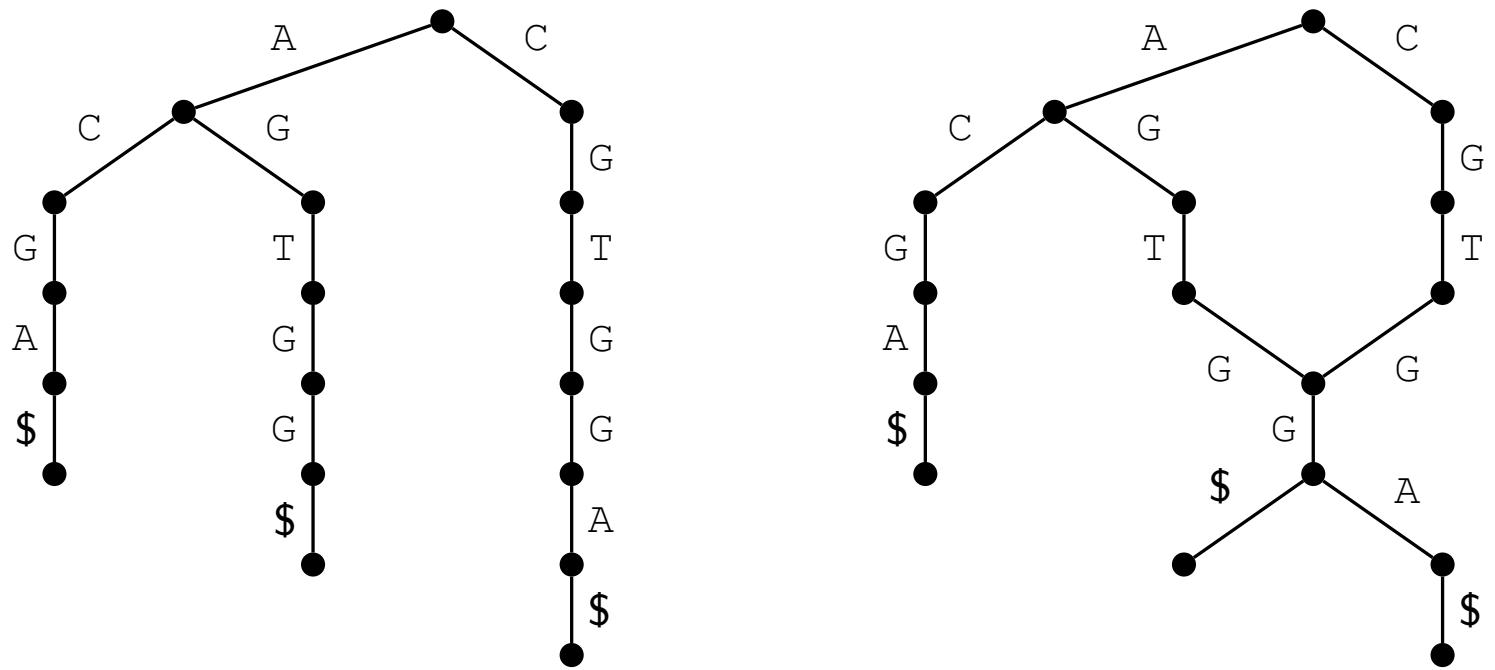
edge minimization leads to

- compactification of representation (tunneling)
- better illustration of string similarities and differences (few edges $\Rightarrow$ many merged nodes)

Purpose #2: Tries

- treelike representation of a set of strings
- can be enhanced with failure links, similar to jump table in Knuth-Morris-Pratt algorithm
  $\Rightarrow$ allows multi-pattern search in linear time
- by using k-mer prefix intervals for tunneling, failure links can be retained

Normal and tunneled trie of $S_1 = \texttt{ACGA\$}$, $S_2 = \texttt{CGTGGA\$}$, $S_3 = \texttt{AGTGG\$}$



Tunneling and sequence analysis overview

- k-mer prefix intervals
  - unpublished
  - (so far) only idea for non-overlapping prefix intervals
- edge minimization
  - output-sensitive algorithm [Weber, 2019, student project]
  - publication planned
- compressed de Bruijn graphs
  - results for construction and usage [Baier et al., 2016]
  - tunneling [Koch, 2019, bachelor thesis]
- tries
  - construction improvements [Ohlebusch et al., 2018]
  - tunneling [Reyes Häusler and Reyes Häusler, 2019, student project]

Summary

Tunneling theory

- contraction of parallel equally labeled paths
- tunnel planning is hard

Application in data compression

- size minimization of run-length-encoded BWT
- heuristical approaches reduce encoding size by about 11 % and about 28 % for bigger files

Application in sequence analysis

- edge minimization in de Bruijn graphs
- tunneling applied to compressed de Bruijn graphs and tries
- decrease of data structure size is about 43 %

Publications

Uwe Baier and Kadir Dede.

BWT Tunnel Planning is Hard But Manageable.
In Proceedings of the 2019 Data Compression Conference, DCC '19, pages 142–151, 2019.

Uwe Baier, Timo Beller, and Enno Ohlebusch.

Parallel Construction of Succinct Representations of Suffix Tree Topologies.
In String Processing and Information Retrieval, SPIRE '15, pages 234–245, 2015.

Uwe Baier, Timo Beller, and Enno Ohlebusch.

Graphical pan-genome analysis with compressed suffix trees and the Burrows-Wheeler transform.
Bioinformatics, 32(4):497–504, 2016.

Uwe Baier, Timo Beller, and Enno Ohlebusch.

Space-Efficient Parallel Construction of Succinct Representations of Suffix Tree Topologies.
Journal of Experimental Algorithmics, 22(1):1.1:1–1.1:26, 2017.

Uwe Baier.

Linear-time Suffix Sorting - A New Approach for Suffix Array Construction.
In Annual Symposium on Combinatorial Pattern Matching, CPM '16, pages 23:1–23:12, 2016.

Uwe Baier.

On Undetected Redundancy in the Burrows-Wheeler Transform.
In Annual Symposium on Combinatorial Pattern Matching, CPM '18, pages 3:1–3:15, 2018.

Enno Ohlebusch, Stefan Stauß, and Uwe Baier.

Trickier XBWT Tricks.
In String Processing and Information Retrieval, SPIRE '18, pages 325–333, 2018.

Student projects

Lisa Arnold.

Patternsuche in einer getunnelten BWT.
Bachelor's thesis, Ulm University, 2019.
not yet finished (June 21, 2019).

Kadir Dede.

Blockwahl beim Tunneln von Burrows-Wheeler-Transformationen.
Elaboration of Project Algorithm Engineering 2018 (draft by Uwe Baier), 2018.

Matthias Koch.

Tunneling compressed DeBruijn Graphs.
Bachelor's thesis, Ulm University, 2019.
not yet finished (May 14, 2019).

Caroline Räther.

Heuristic for Tunneled BWT Block Choice.
Elaboration of Project Algorithm Engineering 2018 (draft by Uwe Baier), 2019.

Sebastian Reyes Häusler and Valentin Reyes Häusler.

Tunneling eXtended BWT's.
Elaboration of Project Algorithm Engineering 2019 (draft by Uwe Baier), not yet finished (May 14, 2019), 2019.

Pascal Weber.

Kantenminimierung in DeBruijn Graphen.
Elaboration of Project Algorithm Engineering 2019 (draft by Uwe Baier), not yet finished (May 14, 2019), 2019.

References I

Alfred V. Aho and Margaret J. Corasick.

Efficient String Matching: An Aid to Bibliographic Search.
Communications of the ACM, 18(6):333–340, 1975.

Jarno Alanko, Travis Gagie, Gonzalo Navarro, and Louisa Seelbach Benkner.

Tunneling on Wheeler Graphs.
In Proceedings of the 2019 Data Compression Conference, DCC '19, pages 122–131, 2019.

Michael Burrows and David J. Wheeler.

A block-sorting lossless data compression algorithm.
Technical Report 124, Digital Equipment Corporation, 1994.

Paolo Ferragina and Giovanni Manzini.

Indexing Compressed Text.
Journal of the ACM, 52(4):552–581, 2005.

Luca Foschini, Roberto Grossi, Ankur Gupta, and Jeffrey Scott Vitter.

When Indexing Equals Compression: Experiments with Compressing Suffix Arrays and Applications.
ACM Transactions on Algorithms, 2(4):611–639, 2006.

Edward Fredkin.

Trie Memory.
Communications of the ACM, 3(9):490–499, 1960.

Travis Gagie, Giovanni Manzini, and Jouni Sirén.

Wheeler graphs: A framework for BWT-based data structures.
Theoretical Computer Science, 698:67–78, 2017.

References II

Travis Gagie, Gonzalo Navarro, and Nicola Prezza.

Optimal-Time Text Indexing in BWT-runs Bounded Space.
In Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '18, pages 1459–1477, 2018.

Jorge E. Hirsch.

An Index to Quantify An Individual's Scientific Research Output.
Proceedings of the National Academy of Sciences of the United States of America, 102(46):16569–16572, 2005.

Juha Kärkkainen, Dominik Kempa, and Simon J. Puglisi.

Hybrid Compression of Bitvectors for the FM-Index.
In Proceedings of the 2014 Data Compression Conference, DCC '14, pages 302–311, 2014.

Donald Knuth, James H. Morris, and Vaughan Pratt.

Fast Pattern Matching in Strings.
SIAM Journal on Computing, 6(2):323–350, 1977.

Heng Li and Richard Durbin.

Fast and accurate long-read alignment with Burrows–Wheeler transform.
Bioinformatics, 26(5):589–595, 2010.

Ilya Muravyov.

`bcm` file compressor.
`https://github.com/encode84/bcm`.
last visited January 2018.

References III

Julian Seward.

`bzip2` file compressor.
`http://bzip.org/`, 1996.
last visited May 2019.