Edge minimization in de Bruijn graphs

Uwe Baier, Thomas Büchler,
Enno Ohlebusch, Pascal Weber

Ulm University

Institute of Theoretical Computer Science

Data Compression Conference
2020

In this talk

Motivation

Tunneling

- compression scheme for the Burrows Wheeler Transform
- applicable to lossless data compression [Baier, CPM 18]
- applicable to FM-index compression [Alanko et al, DCC 19]

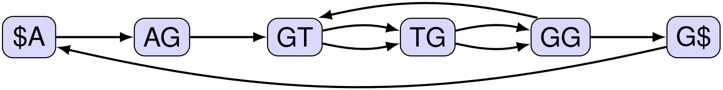
Tunnel planning

- NP-complete for overlapping tunnels [Baier and Dede, DCC 19]
- open problem for non-overlapping tunnels [Alanko et al, DCC 19]

Open problems are: How to find the optimal blocks that minimize space? Can we still support path searching if blocks are overlapping?

De Bruijn graphs

- invented in 1946 for solving combinatorial problems
- usage in bioinformatics, e.g. genome assembly or variation



De Bruijn graph $G_2(\text{AGTGGTGG\$})$ of order $k = 2$.

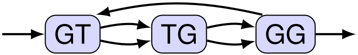
De Bruijn graph

Let S be a string of length n and $k \in [1, n]$.

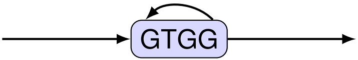
- k-cyclic string of S: $Z_k(S) := S[1..n]S[1..k]$.
- de Bruijn graph $G_k(S) = (\mathcal{K}, E)$ of order k:
  - $\mathcal{K} := \{ Z_k(S)[i..i+k-1] \mid i \in [1, n] \}$
  - $E := \{ (x[1..k], x[2..k+1])^m \mid$
    $x \in \Sigma^{k+1}$ occurs exactly m times in $Z_k(S)\}$

De Bruijn graph compression

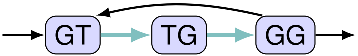
- basis of compression:
  nodes x and y s.t.
  - x is the only predecessor of y
  - y is the only successor of x



- typical compression:
  merge nodes



- our concept:
  fuse multi-edges to one edge

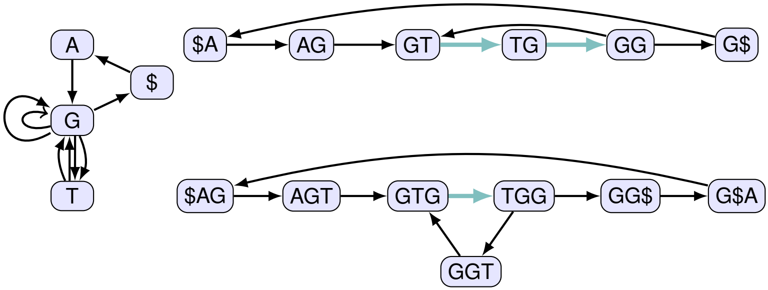


Edge-reduced de Bruijn graphs

- edge-reduced de Bruijn graph $\widetilde{G}_k(S)$: fuse all multi-edges where nodes fulfill the predecessor-successor relationship

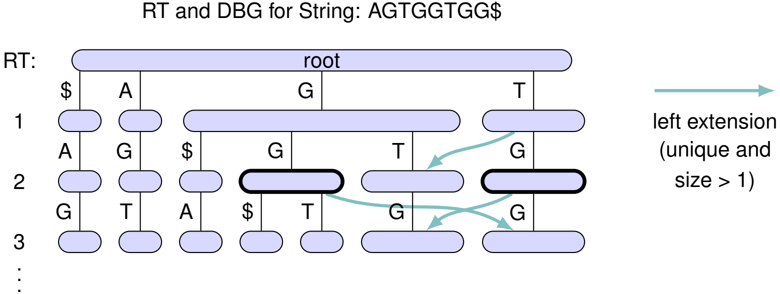
De Bruijn graph edge minimization problem

Let S be a string of length n, find the order $k \in [1, n]$ minimizing the number of edges in the edge-reduced graph $\widetilde{G}_k(S)$.



Edge-reduced DBGs with $k \in \{1, 2, 3\}$ for $S =$AGTGGTGG$.

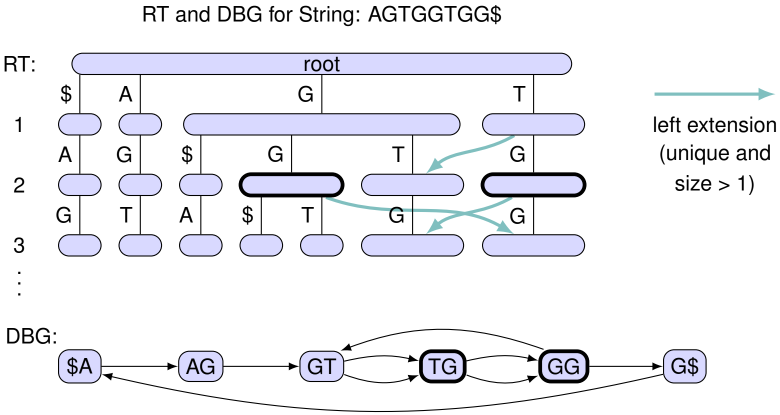
Connection between De Bruijn graph and Rotations Trie



A Rotations Trie is a rooted tree with each node representing a (cyclic) substring of a text:

- *label*(*root*) $= \varepsilon$
- *v* is a child of *u*, iff *label*(*v*) is a right extensions of *label*(*u*)
- The size of node *u* euqals the number of (cyclic) occurences of *label*(*u*) in the text.

Connection between De Bruijn graph and Rotations Trie



- Node in DBG has one predecessor iff the analog node in RT has only one left extension.
- Node in DBG is a single successor iff the target of left extension is a single child.

A naive approach

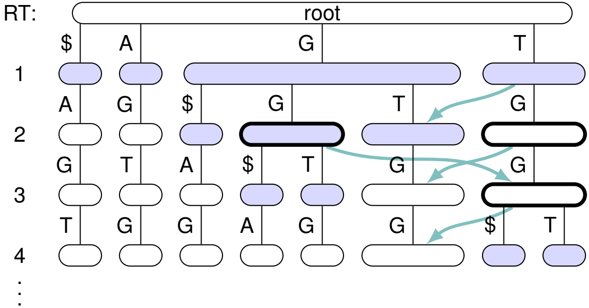
Given the Rotations Trie:

- Iterate over all nodes of a level.
- For each node check if the conditions for edge reduction is given.
- Count how many edges can be removed in this level.
- Go to next level.
- Return level with the lowest edge count.

This method has a quadratic runtime. We can do better!

The efficient algorithm

Information about reductions of edges are passed down from level to level. Not all nodes of a level will be visited, but only nodes that change the structure of RT (blue).



Top down precedure is terminated, when number of nodes of a level is greater than current minimum edge count.

The efficient algorithm

Implementation details:

- A FM-Index instead of a Rotations Trie is used
- Left extensions are backward steps
- Given the FM-Index, the runtime is $\mathcal{O}(m^* \log(\sigma))$
- Given the text, the runtime is $\mathcal{O}(n \log(\sigma))$
- k^*, m^* and vectors marking the fusions are calculated

Burrows Wheeler Transform

- sort list of prefixes and rotations by the rotations

prefixes		sorted rotations
AGTGGTGG	←	$AGTGGTGG
$	←	AGTGGTGG$
AGTGGTG	←	G$AGTGGTG
AGTGGT	←	GG$AGTGGT
AGT	←	GGTGG$AGT
AGTG	←	GTGG$AGTG
A	←	GTGGTGG$A
AGTGG	←	TGG$AGTGG
AG	←	TGGTGG$AG

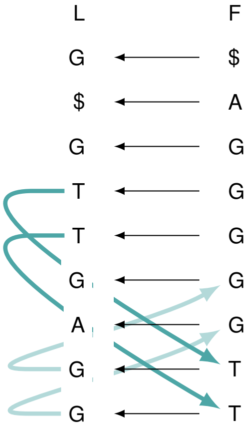
Burrows Wheeler Transform

- sort list of prefixes and rotations by the rotations
- BWT L: last character of prefixes; F: first character of rotations

L		F
AGTGGTGG	←	$AGTGGTGG
$	←	AGTGGTGG$
AGTGGTG	←	G$AGTGGTG
AGTGGT	←	GG$AGTGGT
AGT	←	GGTGG$AGT
AGTG	←	GTGG$AGTG
A	←	GTGGTGG$A
AGTGG	←	TGG$AGTGG
AG	←	TGGTGG$AG

Burrows Wheeler Transform

- sort list of prefixes and rotations by the rotations
- BWT L: last character of prefixes; F: first character of rotations
- backward step: k-th c in L corresponds to k-th c in F



- final BWT: string L (string F is given implicitely)

BWT Tunneling

- basis: equal suffixes of adjacent prefixes (prefix intervals)

prefixes		sorted rotations
AGTGGTGG	←	\$AGTGGTGG
\$	←	AGTGGTGG\$
AGTGGTG	←	G\$AGTGGTG
AGTGGT	←	GG\$AGTGGT
AGT	←	GGTGG\$AGT
AGTG	←	GTGG\$AGTG
A	←	GTGGTGG\$A
AGTGG	←	TGG\$AGTGG
AG	←	TGGTGG\$AG

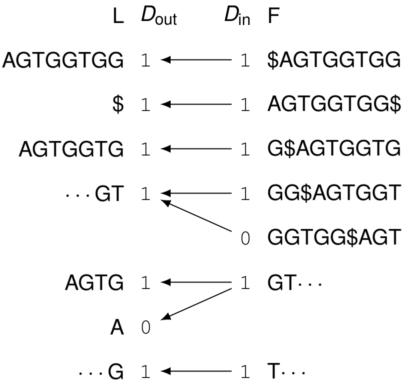
BWT Tunneling

- basis: equal suffixes of adjacent prefixes (prefix intervals)
- mark rotations and prefixes ending in the prefix interval

prefixes		sorted rotations
AGTGGTGG	←	\$AGTGGTGG
\$	←	AGTGGTGG\$
AGTGGTG	←	G\$AGTGGTG
AGTGGT	←	GG\$AGTGGT
AGT	←	GGTGG\$AGT
AGTG	←	GTGG\$AGTG
A	←	GTGGTGG\$A
AGTGG	←	TGG\$AGTGG
AG	←	TGGTGG\$AG

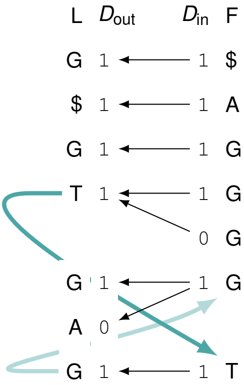
BWT Tunneling

- basis: equal suffixes of adjacent prefixes (prefix intervals)
- mark rotations and prefixes ending in the prefix interval
- fuse adjacent marked entries; mark tunnel ends



BWT Tunneling

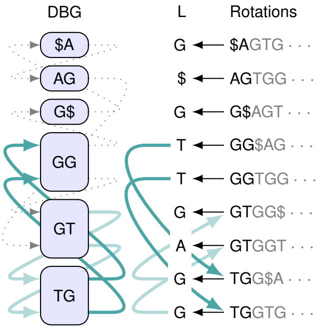
- basis: equal suffixes of adjacent prefixes (prefix intervals)
- mark rotations and prefixes ending in the prefix interval
- fuse adjacent marked entries; mark tunnel ends

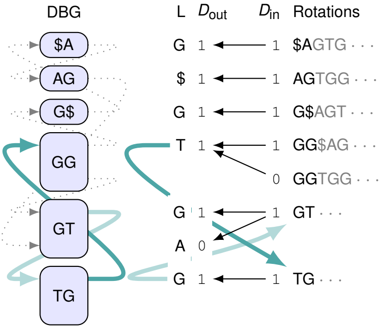


- final tunneled BWT: string L and bit-vectors D_{out} and D_{in}

Tunneling and Edge reductions

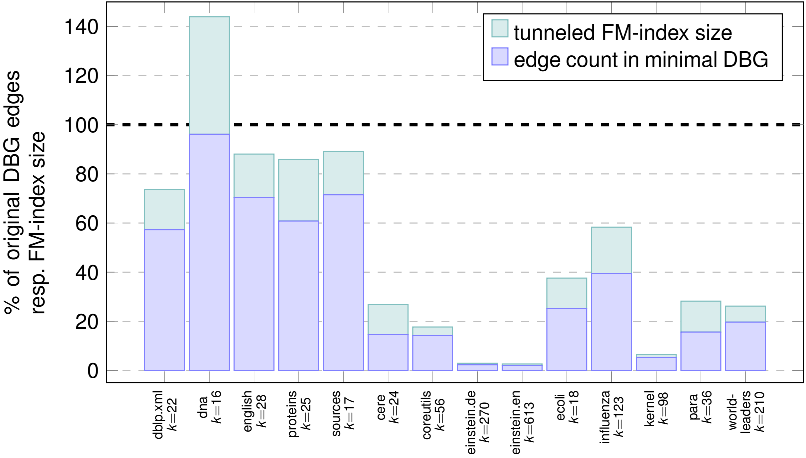
- fusible paths in DBG correspond to special non-overlapping prefix intervals in the BWT
- number of edges in an edge-reduced DBG is equal to length of corresponding tunneled BWT
- minimizing edges minimizes tunneled BWT length using a restricted class of prefix intervals





Experimental Results

- test data: files from the Pizza & Chili and Repetitive corpus
- tunneled FM-index: tunneled BWT as a wavelet tree with overhead of two additional bit-vectors D_{out} and D_{in}



Conclusion

- introduction of the DBG edge minimization problem
  $\rightarrow$ efficient problem-solving algorithm
- deep connection to tunneling of non-overlapping prefix intervals
  $\rightarrow$ major progress in the open problem of Alanko et al.
  $\rightarrow$ FM-index size reduction of about 80 % for repetitive files
- edge-minimal DBGs are interesting in their own right
  $\rightarrow$ constitution of graphs with minimum redundancy

Thanks for your Attention!