Burrows-Wheeler Transform for Similar Sequences

Jouni Sirén
University of Chile

Burrows-Wheeler transform

- Sort the suffixes in lexicographic order, and take the preceding character for each of the suffixes.
- The combinatorial structure is similar to the suffix tree and the suffix array.
- The BWT is usually more repetitive than the text.
- Used in data compression, text indexing, and sequence analysis.

Data structures

- Algorithms: Find the occurrences of P in T using BWT(T).
- Data structures: Store BWT(T), while supporting access(), rank(), and select() efficiently.

- The encoding depends on the data.
- The interface is a model of the data structure. It can be useful, but it is certainly wrong.
- If someone uses BWT in their algorithm in a different way, we should be able to extend our interface to support it.

Compression

Given collection $C = \{ S_1, \ldots, S_r \}$ of similar sequences, store BWT(C) in small space, while supporting access(), rank(), and select() efficiently.

Run-length encoding

Text	**BWT**
GATTACA$	ACTGA$TA
GATTACA$	AAAACCCCTTTTGGGGAAAA$$$$TTTTAAAA
GATTACA$	A4C4T4G4A4$4T4A4
GATTACA$	AAAACCCCTTCTGGGGAAAAT$$$$TTTAAAA
GATTACA$	A4C4T2C1T1G4A4T1$4T3A4
GATTACA$	

Run-length encoded BWT

- Given r copies of a random sequence of length n over an alphabet of size σ, and a total of s edit operations, the number of runs in BWT is at most n + $O(s \log_\sigma(rn))$ in the expected case [Mäkinen et al., Journal of Computational Biology, 2010].

- Probably tens of bytes per edit operation for human genomes.

- What is an edit operation?

BWT construction

- We can probably store BWT-based indexes for hundreds or even thousands of genomes is main memory, if we can build them in the first place.

- The main problem is construction speed, not memory usage.

- 1 MB/s ≈ 100 GB/day is good for at most 1 TB.

- 10 MB/s is good for at most 10 TB.

The best general-purpose algorithm is [Hon et al., Algorithmica, 2007], first published in 2002.

Split the input into p parts, build BWT for each of the parts separately, and merge the parts.

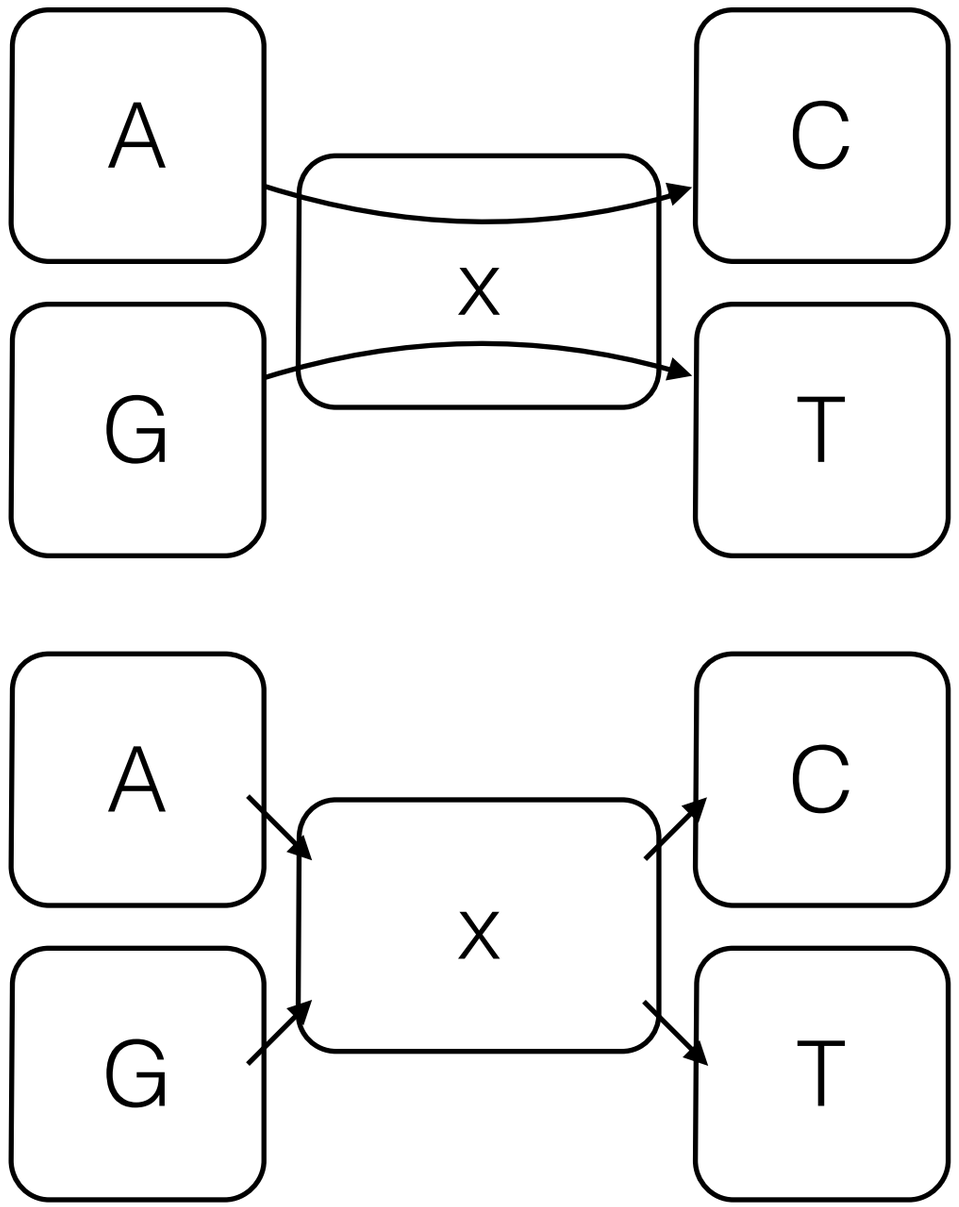
Requires $O(n + p \cdot |RLE(BWT)|)$ time and $|RLE(BWT)| + O((n/p) \log n)$ bits of space.

Sequence S can be inserted to / deleted from the BWT in $O(|S| + |RLE(BWT)|)$ time.

My proof-of-concept implementation indexed ~100 GB/day in 2009. ~1 TB/day could be possible.

Generalization

The collection contains AxC and GxT, but not AxT or GxC.



We can replace the implicit graph structure in the run-length encoded BWT with an explicit graph structure.

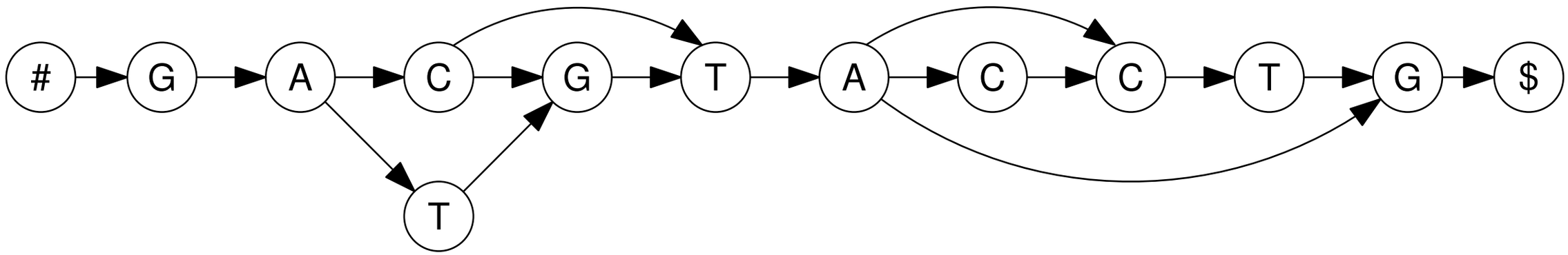
Generalized BWT

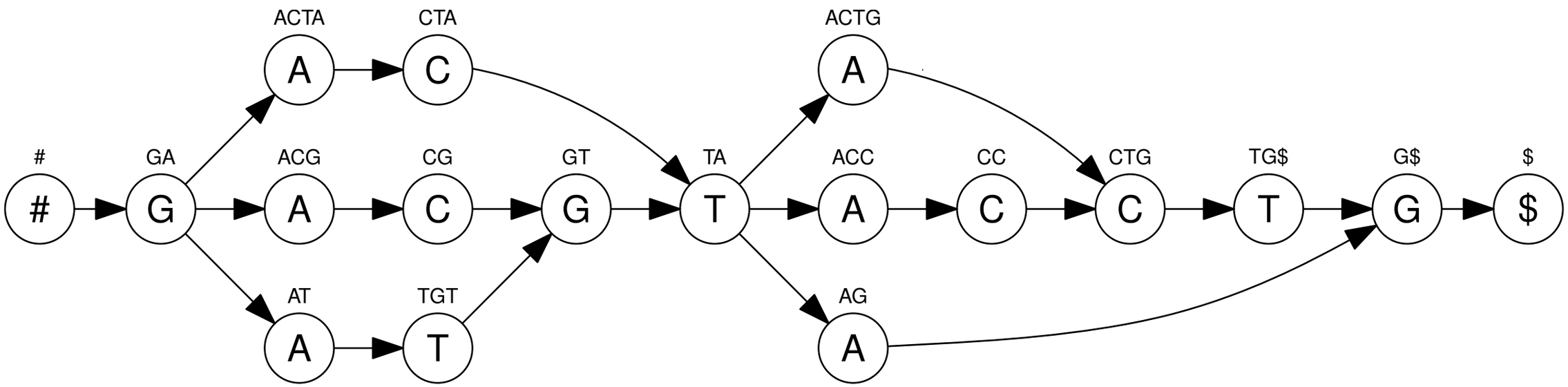
- Key property of BWT: Suffixes starting with c are in same order as suffixes preceded by c.

- de Bruijn graphs have this property, while DAGs can be transformed into equivalent automata having the property.

- Transformation is exponential in the worst case, but linear with a reasonable input.

- Human genomes are almost reasonable.





	$	ACC	ACG	ACTA	ACTG	AG	AT	CC	CG	CTA	CTG	G$	GA	GT	TA	TG$	TGT	#
BWT	G	T	G	G	T	T	G	A	A	A	AC	AT	#	CT	CG	C	A	$
Edges	1	1	1	1	1	1	1	1	1	1	1	1	100	1	100	1	1	1

GCSA

- With some heuristics, we can build BWT-based indexes for entire populations in ~10 hours, ~200 GB [Sirén et al., TCBB, 2014].

- Better construction algorithms may exist.

- In principle, the GCSA is a black-box replacement for BWT-based indexes.

- In practice, sequence analysis algorithms usually assume that the input is a sequence.

Relative compression

If sequence S is similar to reference sequence R, BWT(S) is also similar to BWT(R). With BWT(S | R) and BWT(R), we can simulate BWT(S) directly or decompress it.

Relative FM-index

- Find the LCS of BWT(S) and BWT(R). This can be approximated quickly with backward searching and the greedy LCS algorithm.
- BWT(S | R) contains the alignment of BWT(S) and BWT(R), and the complements BWT(S) – LCS and BWT(R) – LCS.
- BWT(S).rank = BWT(R).rank + (BWT(S) – LCS).rank – (BWT(R) – LCS)
- More flexible than building BWT for the entire collection, but compression is worse.
- [Belazzougui et al., SPIRE 2014]

NA12878 vs. reference

Index	Construction	Size	Queries
FM-index, fast	–	1090 MB	57.31 s
FM-index, small	–	636 MB	325.49 s
Relative FM, fast	589 s	218 MB	619.81 s
Relative FM, medium	575 s	199 MB	1058.75 s
Relative FM, small	5412 s	173 MB	1921.92 s

Conclusions

- We can probably store a BWT-based index for hundreds or even thousands of genomes in main memory. The bottleneck is construction speed.

- The BWTs of individual genomes can be encoded relative to the BWT of the reference. By doing this, we trade compression for flexibility.

- BWT can also be built for graph-based representations of multiple genomes. Heuristics are often needed to avoid exponential growth.