Lightweight Reference-Free Variation Detection using the Burrows-Wheeler Transform

Nicola Prezza[1], Nadia Pisanti[1], Marinella Sciortino[2], and Giovanna Rosone[1]*

[1] Dipartimento di Informatica, Università di Pisa, Pisa, Italy.
[2] Dipartimento di Matematica e Informatica, Università di Palermo, Palermo, Italy.
* Corresponding author: `giovanna.rosone@unipi.it`.

Motivation We study the problem of identifying SNPs and INDELs within a reads set without aligning it against a reference sequence. Most existing tools for this problem are based on de-Bruijn graphs and share some limitations as their order k is usually small ($\approx$ 30 bases) and they do not store k-mer coverage and k-mer adjacency in reads.

Methods We describe a new approach based on the extension of the Burrows-Wheeler transform (BWT) to a collection of strings. We show that the the output of such a transformation can be partitioned in clusters (substrings), each associated with a position of the underlying (unknown) genome; if that position exhibits a variant, then the cluster will contain more than one distinct character.

Results We compared the performance of our tool eBWT2SNP with the state-of-the-art tool DiscoSnp++ on real and simulated Human datasets. Already at 10x coverage, our tool discovers 80% of existing SNPs and 59% of the INDELs (versus 55% and 32% of DiscoSnp++). At 48x, these percentages increase to 96% and 87% (versus 75% and 46% of DiscoSnp++). DiscoSnp++ , on the other hand, is more precise: on average, 89% of its output SNPs and 94% of its output INDELs are correct (versus 80% and 90% of our tool). Due to the fact that we use compressed data structures, DiscoSnp++ is also faster. We are currently developing a parallel version our algorithms to become competitive also under this metric.

DEFINITIONS

Our tool relies on the *extended Burrows-Wheeler transform* of the input reads:

Definition: eBWT

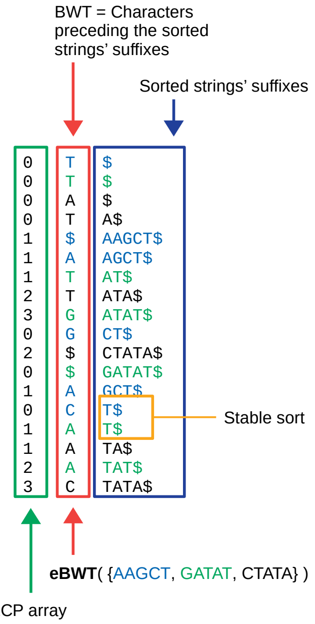
`eBWT` is the string containing the characters preceding the lexicographically-sorted reads' suffixes. Ties are broken by input order.

We also use the *Longest Common Prefix* array:

Definition: LCP

`LCP` is the array containing the lengths of the longest common prefixes between adjacent suffixes in lexicographic order.

In the following example, we show `eBWT` and `LCP` on the reads set {`AAGCT`, `GATAT`, `CTATA`}. Also the sorted reads' suffixes are shown (blue box). Note that these are shown only for illustrative purposes and are not stored in practice.



BWT = Characters preceding the sorted strings' suffixes

Sorted strings' suffixes

LCP array	eBWT	Sorted strings' suffixes
0	T	$
0	T	$
0	A	$
0	T	A$
1	$	AAGCT$
1	A	AGCT$
1	T	AT$
2	T	ATA$
3	G	ATAT$
0	G	CT$
2	$	CTATA$
0	$	GATAT$
1	A	GCT$
0	C	T$
1	A	T$
1	A	TA$
2	A	TAT$
3	C	TATA$

Stable sort

eBWT({AAGCT, GATAT, CTATA})

LCP array

PRE-PROCESSING

`eBWT` can be computed with lightweight tools such as `BCR` (`github.com/BEETL/BEETL`) and `EGSA` (`github.com/felipelouza/egsa`). Currently, this is the bottleneck of our method: these tools process data at a rate of $\approx$ 5GB per hour. We are currently looking into parallel algorithms to speed this step up.

POSITIONAL CLUSTERING

Our method relies on the *clustering* property of the `eBWT`. In [1] we prove the following theorem:

Theorem [1]

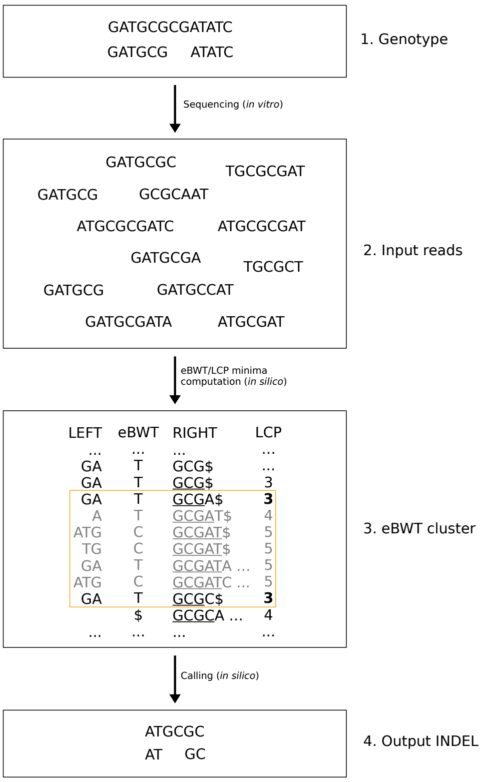
Let i and j be local minima in `LCP`. With high-enough probability, the *cluster* `eBWT`$[i, \ldots, j]$ contains the sequenced copies of a single position in the genome.

According to the previous theorem, a genome position contains a variation if and only if the corresponding `eBWT` cluster contains two different letters. This suggests the following strategy:

Our strategy: eBWT2SNP

1. Compute `eBWT` and `LCP`.
2. Detect `LCP` minima ⇒ `eBWT` clusters.
3. If the cluster contains 2 distinct letters ⇒ variation found.
4. Extract context, output variation.

We add INDELs detection w.r.t. the preliminary version [1]. See the following example; in bold: `LCP` minima. Inside orange box: `eBWT` cluster.



GATGCGCGATATC
GATGCG ATATC

1. Genotype

Sequencing (*in vitro*)

GATGCGC TGCGCGAT
GATGCG GCGCAAT
ATGCGCGATC ATGCGCGAT
GATGCGA TGCGCT
GATGCG GATGCCAT
GATGCGATA ATGCGAT

2. Input reads

eBWT/LCP minima computation (*in silico*)

LEFT	eBWT	RIGHT	LCP
...	...	...	...
GA	T	GCG$	...
GA	T	GCG$	3
GA	T	GCGA$	**3**
A	T	GCGAT$	4
ATG	C	GCGAT$	5
TG	C	GCGAT$	5
GA	T	GCGATA ...	5
ATG	C	GCGATC ...	5
GA	T	GCGC$	**3**
	$	GCGCA ...	4
...	...	...	...

3. eBWT cluster

Calling (*in silico*)

ATGCGC
AT GC

4. Output INDEL

SPACE-EFFICIENT COMPUTATION

Our tool takes as input just `eBWT`. How do we find `LCP` minima? In [2], we show:

Theorem [2]

Given `eBWT`, we can find all `LCP` minima in linear time using just 1 Byte per base in RAM.

We moreover build a compressed index on top of the `eBWT` to extract the context surrounding the variations. This strategy uses 8 times less space than the preliminary version [1].

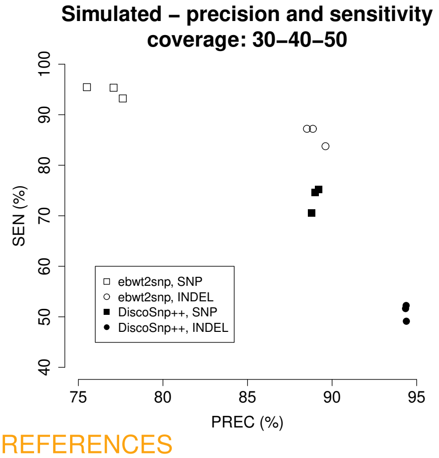
RESULTS

We compared eBWT2SNP with DiscoSnp++ , the state-of-the-art tool for reference-free variation detection. All datasets have been downloaded from the `1000genomes` website. We processed reads — both real and simulated — from a single individual in order to reconstruct its genotype:

Experiments

1. **Simulated**. Heterozygous reads simulated from `HG00096`, Chr1, cov. 50x. Variations taken from the real `VCF` file.
2. **Real**. Reads sequenced from `HG00096`, Chr1, cov. 48x.

Results on simulated data:



REFERENCES

[1] N. Prezza, N. Pisanti, M. Sciortino, and G. Rosone. SNPs detection by eBWT positional clustering. *Algorithms for Molecular Biology*, 14(1):3, 2019.

[2] N. Prezza and G. Rosone. Space-Efficient Computation of the LCP Array from the Burrows-Wheeler Transform. In *CPM 2019*, Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.