Abstract :

- The Problem is De Bruijn graphs Consume Large memory.
- We have a solution to reduce this complexity by compacting paths with single vertices
- On the other hand, this way has a problem because it requires the graph to be in the memory.
- Bifrost features is a parallel and algorithm which saves the memory ,also.
- Our Parallel algorithm enable the direct construction of the compacted graphs.
- Bifrost has some added features such as indexing, editing, querying and the graph coloring method.

Introduction :

- De Bruijn graph is an abstract data structure used in computational biology as a tool for genome assembly.
- Since 2008, the de Bruijn graph genome assemblers range have been released.
- De Bruijn graph-based methods are nonetheless used to assemble and correct long reads.
- De Bruijn graphs have found widespread by solving a variety of problems such as de novo transcriptome assembly, variant calling, short read compression, short read correction, long read correction, and short read mapping to name a few.
- There is a colored de Bruijn graph which is a variant of the de Bruijn graph.
- The colored de Bruijn graph keeps track of the source of each vertex in the graph.
- The initial application was for assembly, genotyping, pan-genomics, variant calling, and transcript quantification methods.

Related Works:

- **An Eulerian path approach to DNA fragment assembly**

_ The reduction of the fragment assembly to a variation of the classical Eulerian path problem that allows one to generate accurate solutions of large-scale sequencing problems.

_ we do a very counterintuitive (some would say childish) thing: we cut the existing pieces of a puzzle into even smaller pieces of regular shape.

https://www.pnas.org/content/98/17/9748.short

• TwoPaCo: an efficient algorithm to build the compacted de Bruijn graph from many complete genomes

_ de Bruijn graphs have been proposed as a data structure to facilitate the analysis of related whole genome sequences, in both a population and comparative genomic settings.

_ However, current approaches do not scale well to many genomes of large size (such as mammalian genomes).

https://academic.oup.com/bioinformatics/article/33/24/4024/2725383?login=true

• From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy

_ A major limitation of nanopore sequencing is its high error rate, which despite recent improvements to the nanopore chemistry and computational tools still ranges between 5% and 15%.

_ Here, we review computational approaches determining the nanopore sequencing error rate. Furthermore, we outline strategies for translation of raw sequencing data into base calls for detection of base modifications and for obtaining consensus sequences.

https://link.springer.com/article/10.1186/s13059-018-1462-9

• Disk-based compression of data from genome sequencing

_ We propose overlapping reads compression with minimizers, a compression algorithm dedicated to

sequencing reads (DNA only).

_ Our method makes use of a conceptually simple and easily parallelizable idea of minimizers, to obtain 0.317 bits per base as the compression ratio, allowing to fit the 134.0 Gbp dataset into only 5.31 GB of space.

https://academic.oup.com/bioinformatics/article/31/9/1389/200464?login=true

Methods :

- The concepts and data structures used throughout this paper section describes how the uncompacted de Bruijn graph is built from a set of sequencing reads.

- Constructing the compacted de Bruijn graph section shows how the approximate compacted de Bruijn graph is built from its uncompacted counterpart and subsequently converted to an exact compacted de Bruijn graph.
- Coloring section presents how the graph coloring is built efficiently on top of the compacted de Bruijn graph.

Results :

cdBG construction

- We constructed the cdBG of the NA12878 human genome short read dataset from the Genome In A Bottle consortium [55]. The dataset is downsampled from 300-fold to 30-fold coverage to reflect normal sequencing depth, resulting in about 696 million 150-bp paired-end sequences.
- We compared Bifrost to BCALM2 as it can build a cdBG from short read data or assembled genomes.
- BCALM2 can be configured for different memory usage where a lower memory usage results in a longer running time
- Additionally, BCALM2 uses by default up to 5 GB of disk space while Bifrost does not use any disk except for the final output.

cdBG querying

- We compared Bifrost to two tools for querying dBGs based on the k-mer composition of the queries.
- For querying, Bifrost takes as input the graph it constructed and builds an index for querying k-mers.
- To query the graph, we used 30 million single-end reads from the NA12878 short read dataset that was used to construct the reference graph.
- Note that both Bifrost and Mantis return query hits for every query while Blight only returns the total number of k-mers found in the graph from all input queries.
- Overall, the results show that Bifrost is the fastest at querying, while using 26.8 GB of memory, whereas Blight uses less memory at the expense of speed.

- The low memory usage of Blight is partially explained by the fact that Blight maintains its index in main memory but stores subsequences of the graph on disk.

cdBG coloring

- We constructed ccdBGs with k = 31 for a maximum of 117,913 assembled genomes of Salmonella, The input represents all publicly available Salmonella.
- This is a 7.3× increase in the number of colors compared who reported the ccdBG construction for 16,000 Salmonella strains.
- We compared Bifrost to VARI-merge as both tools can construct the colored de Bruijn graph and update it without reconstructing the graph entirely.
- The main differences between the two tools is VARI-merge is mainly a disk-based method that produces a non-compacted colored.
- We only benchmarked VARI-merge as it is currently the state-of-the-art for colored de Bruijn graph construction.