Techniques for de novo genome and metagenome assembly

Rayan Chikhi

Univ. Lille, CNRS

séminaire INRA MIAT, 24 novembre 2017

short bio

@RayanChikhi
http://rayan.chikhi.name

- compsci/math background
- algorithms and data structures related to DNA sequencing
- software tools around *de novo* assembly:
 - Minia, DSK, Bcalm, KmerGenie, GATB
- actual assemblies: giraffe, gorilla Y

Genome assembly



Why assemble

Create genome/transcriptome

- Novel insertions
- make sense of un-mapped reads
- SNPs in non-model organisms
- Find SVs
- Regions of interest same techniques as:
 - DNA variants detection
 - Transcript quantification
 - Alternative splicing detection



Graphs for assembly

Overlaps between reads is the basic information used to assemble.

Graphs are used to represent reads (or k-mers) and overlaps.

2 types:

- de Bruijn graphs
- string graphs

for Illumina data for PacBio/Nanopore data

Overlap graphs

First, agree on some definition of "what is an overlap".

- 1. Nodes = reads.
- 2. Edges = overlap between two reads.

In this example, let's say that an overlap needs to be:

exact

- and over at least 3 characters.

ACTGCT

CTGCT (overlap of length 5) GCTAA (overlap of length 3)



String graphs

A **string graph** is obtained from an overlap graph by removing redundancy:

- redundant reads (those fully contained in another read)
- transitively redundant edges (if $a \rightarrow c$ and $a \rightarrow b \rightarrow c$, then remove $a \rightarrow c$)

Example: ACTGCT CTGCT (overlap length 5) GCTAA (overlap length 3)	Let's have inexact overlaps here ACTGCT CTACT GCTAA	
		.T A

de Bruijn graphs

A **de Bruijn** graph for a fixed integer *k*:

- 1. **Nodes** = all k-mers (substrings of length k) in the reads.
- 2. There is an **edge** between x and y if the (k 1)-mer prefix of y matches exactly the (k 1)-mer suffix of x.

TGA



de Bruijn graphs

A **de Bruijn** graph for a fixed integer *k*:

- 1. **Nodes** = all k-mers (substrings of length k) in the reads.
- 2. There is an **edge** between x and y if the (k 1)-mer prefix of y matches exactly the (k 1)-mer suffix of x.

ACTG CTGC TGCT GCTG CTGA TGAT dBG, k = 3:



Data is messy



dBG of S. aureus, uncleaned (SRR022865) - it's a SMALL genome

Salmonella genome, cleaned assembly (Velvet), 100 bp Illumina reads. k = 51



Salmonella genome, cleaned assembly (Velvet), 100 bp Illumina reads. k = 61



Salmonella genome, cleaned assembly (Velvet), 100 bp Illumina reads. k = 71



Salmonella genome, cleaned assembly (Velvet), 100 bp Illumina reads. k = 81



Salmonella genome, cleaned assembly (Velvet), 100 bp Illumina reads. k = 91

[™] がそ入(アダーへ))-1----1--1--いい-いいい Set Control and the set of a second a second second second second

How an assembler works

[SPAdes, Velvet, ABySS, SOAPdenovo, SGA, Megahit, Minia, .., FALCON, Canu]

- 1) Maybe correct the reads. (SPAdes, SGA, FALCON, Canu)
- 2) Construct a graph from the reads.

Assembly graph with variants & errors



3) Likely sequencing errors are removed. (not in FALCON and Canu)



3) Known biological events are removed. (not in FALCON and Canu)

4) Finally, **simple paths** (i.e. contigs) are returned.



Steps of Illumina-specific assemblers



- computational bottlenecks at early stages
- qualitative choices at later stages

Graph formats

- FASTG
- GFA
- GFA2

Η	VN:	Z:1.()		
S	11	ACC	ГТ		
S	12	TCA	AGG		
S	13	CTT	GATT		
L	11	+	12	-	4M
L	12	-	13	+	5M
L	11	+	13	+	ЗM
Ρ	14	11+	,12-	,13+	4M,5M



Outline

3 technical focuses:

- graph compaction
- graph cleaning
- multi-k assembly

Compacted de Bruijn graph



Each non-branching path becomes a single node (*unitig*).

- no loss of information
- less space
- actually an assembly (very conservative, preserves all variants)

Compaction: BCALM2 (ISMB'16)



minimizer of s: smallest ℓ-mer in s [Roberts et al, 2004]

e.g. $(\ell = 2, \text{lexicographical order})$

TGACGGG GACGGGT ACGGGTC CGGGTCA GGGTCAG GGTCAGA

Frequency ordering → better repartition. [RECOMB'14]



Compaction of partitions

k-mers are partitioned w.r.t minimizer. In this case, compacting all partitions returns exactly all the unitigs.

Can't just compact partitions



Left: 1 unitig = multiple partitions. (*Merge them?*) Right: 1 partition = multiple unitigs. (*Split them?*)

Strategy

- Put certain *k*-mers into two partitions.
- x is a **doubled kmer** when minimizer $(x[1..k-1]) \neq$ minimizer(x[2...k]).



Partitions with doubled k-mers



ATGACC



Compacted partitions: GTGACGA ACGAC **ACGAA** ACGAAG

BCALM 2's partial compaction module

Doubled kmers



Lemma 1: doubled *k*-mers appear as prefixes or suffixes of compacted strings.

Lemma 2: Gluing together strings with matching doubled *k*-mers yield unitigs.

Big picture



BCALM2 performance

BCALM 2 Spruce (1.1 TB reads, k = 61)

Unitigs 56.0 Gbp

Time 8 h 52 m



Memory 31 GB

Previousa assembly of a 20 Gbp tree: 4 months, 0.8 TB RAM in [Zimin 2014]

Human NA18507	Bcalm 2	Bcalm 1	ABySS-P 1.9
Time	2 h	13 h	6.5 h
Memory	2.8 GB	43 MB	89 GB

32 Gbp axolotl genome

full assembly memory	140 GB
full assembly time	\approx 8 days
Contigs	21 Gbp
#	47 M
N50	1.3 kb





Graph simplifications (SPAdes-inspired)



Tip removal: len_{tip} ≤ 3.5k or len_{tip} ≤ 10k 2cov_{tip} ≤ cov_{neighbors}



Bulge removal:

len_{bulge} ≤ max(3k, 100)
cov_{bulge} ≤ 1.1cov_{altpath}
len_{altpath} = len_{bulge} ± delta
delta = max(0.1len_{bulge}, 3)



Erroneous connection removal: $len_{EC} \le 10k$ $4cov_{EC} \le cov_{neighbors}$

Metagenome assembly

- 1. closely related strains
- 2. uneven depths, & low depths
- 3. inter-species repeats
- 4. large size of datasets
- 5. lack of long reads

(adapted from A. Korobeynikov)

Multi-k



Enables to include low-coverage (short) k-mers as well as repeat-resolving (longer) k-mers.

Visualization of multi-k graphs

Salmonella genome, cleaned assembly (SPAdes), MiSeq reads.



Visualization of multi-k graphs

Salmonella genome, cleaned assembly (SPAdes), MiSeq reads.



Visualization of multi-k graphs

Salmonella genome, cleaned assembly (SPAdes), MiSeq reads.

k = 99

the SPAdes assembler

SPAdes: multi-k de Bruijn graph assembler, no graph coverage cut-off, graph simplifications.

exSPAnder: universal module for repeat resolution and scaffolding

metaSPAdes¹:

- performance enhancements
- coverage-aware utilization of paired reads in repeat resolution
- some tuning in graph simplifications

¹from A. Korobeynikov slides

metaSPAdes

figure from A. Korobeynikov

the MEGAHIT assembler

MEGAHIT: HKU metagenomic assembler. Steps:

- 1. memory-intensive k-mer counting
- 2. succinct de Bruijn graph (SDBG) construction
- 3. mercy k-mers
- 4. graph simplifications (now using IDBA's)
- 5. metagenome-tuned scaffolding
- 6. multi-k iterations

CAMI metagenomics assembly

Assembly "debugging"

Investigating long-read genome assemblies using string graph analysis

Canu assembled contigs projected onto minimap's overlap graph

P. Marijon

Conclusion

Research directions in assembly:

- 1. Improving quality of large genome/metagenome Illumina assemblies
- 2. accurate consensus (3rd gen)
- 3. haplotype-resolution (3rd gen)
- 4. performance (3rd gen)

What I didn't talk about:

- BBHash: minimal perfect hashing (for k-mers or anything)
- GATB library: C++ library for creating NGS tools
- DE-kupl: differential expression on k-mers

