The tumultuous fate of sequence bioinformatics ideas

Rayan Chikhi
Institut Pasteur
SeqBio team

SeqBim 2022
Slides: https://t.ly/ZC-Y

Hello

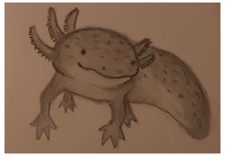
- PI, Bioinformatics algorithms lab @ Institut Pasteur
- CV: PhD @ ENS Rennes/Genscale, Postdoc @ PSU, CNRS @ Lille/Bonsai

@RayanChikhi on Twitter

http://rayan.chikhi.name

Sequence Bioinformatics

@ Institut Pasteur



Genomes & metagenomes assembly

Algorithms and data structures on k-mers

Sequence search in very large datasets

Pangenomics

cnrs PR[AI]RIE PaRis Artificial Intelligence Research InstitutE INSTITUT PASTEUR

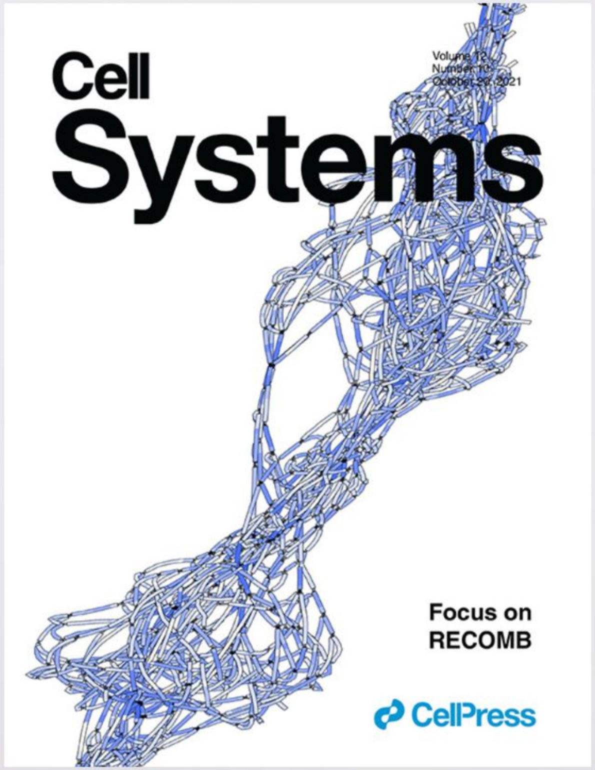
Talk plan

1) mdBG behind-the-scenes
2) WFA history

Who’s read:

1. the mdBG paper? (Ekim et al, RECOMB 2021, Cell Systems 2013)
2. the minia paper? (Rizk, Chikhi, WABI 2012, Almob 2013)
3. the WFA paper? (Marco-Sola et al, Bioinformatics 2021)

highly scalable dBGs: Minimizer-space de Bruijn graphs



Barış Ekim

Bonnie Berger

Preliminaries *k*-mers, de Bruijn graph (dBG)

Reference genome ACTGAGTACCATGGAC

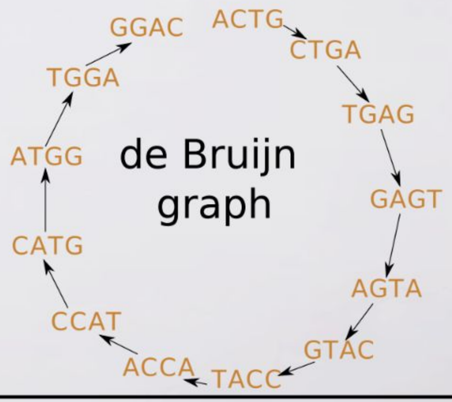
Reads

```
ACTGAGTAC
 CTGAGTACCAT
   GAGTACCATGGAC
```

Base-space

k-mers

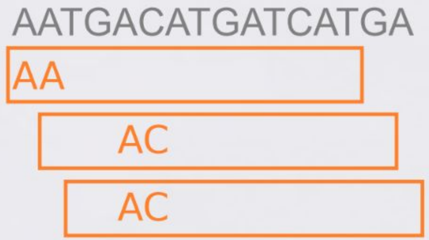
```
ACTG      TACC     GGAC
 CTGA      ACCA
  TGAG      CCAT
   GAGT      CATG
    AGTA      ATGG
     GTAC      TGGA
```



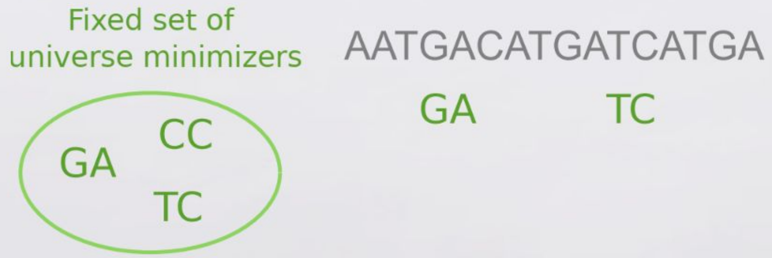
Preliminaries: Minimizers

Two kinds:

- **window**. Local: “smallest” ℓ-mer in a window



- **universe**. Global: set of ℓ-mers with low hash values



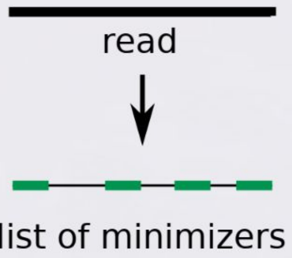
From now on: **universe**. (Also called Scaled MinHash)

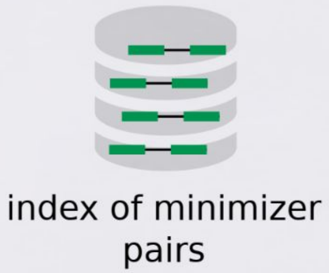
This work: stems from three ideas

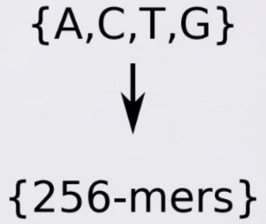
Shasta
(Shafin et al, 2020)

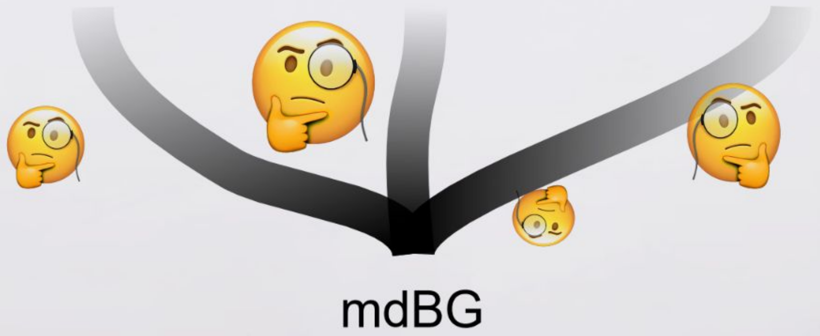
Peregrine
(Chin et al, 2019)

wtdbg2
(Ruan et al, 2020)









Our approach: Minimizers as *tokens* of the alphabet

Classical alphabet: $\Sigma_{DNA} = \{A, C, T, G\}$

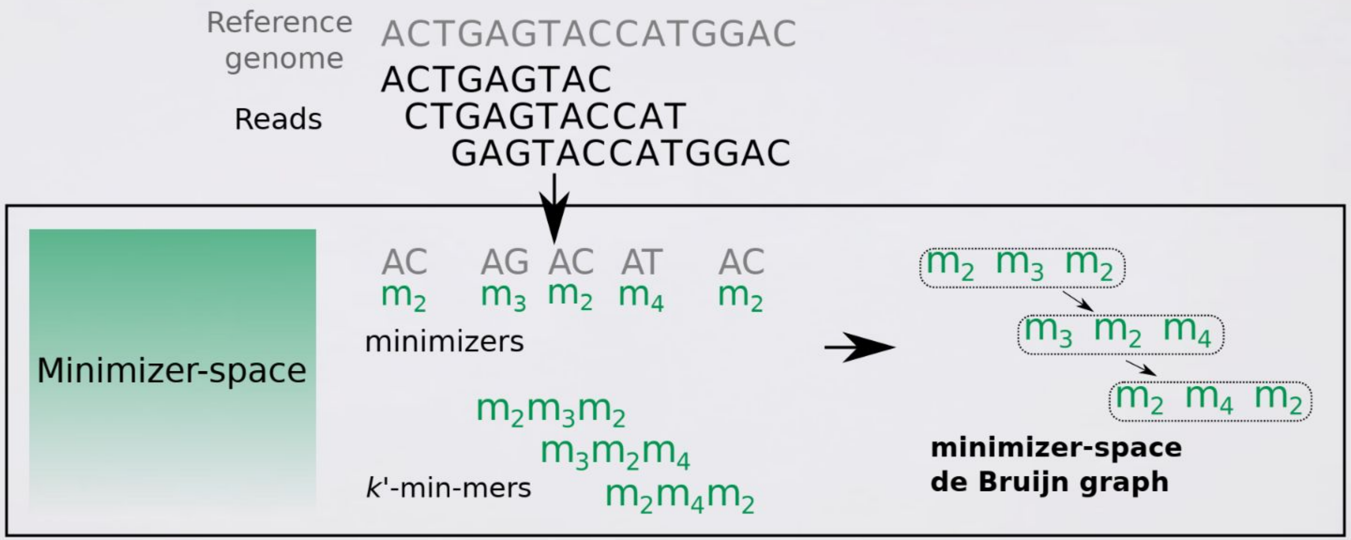
A k-mer with $k = 3$: AGT

Minimizer alphabet: $\Sigma^{\ell} = \{\text{all minimizers of length } \ell\} = \{m_1, m_2, m_3, \ldots\}$

where e.g. $\ell = 2$, $m_1 = AA$, $m_2 = AC$, $m_3 = AG$, $m_2 = AT$

A k-mer over Σ^{ℓ} (a k**-min-mer**): $m_1 m_3 m_2$

Minimizer-space de Bruijn graph



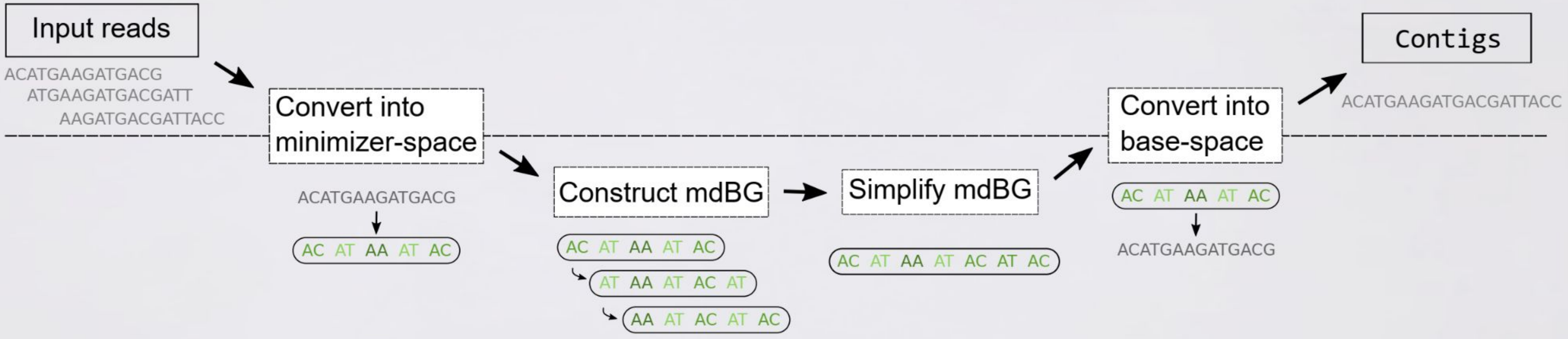
A **minimizer-space de Bruijn graph** is a **de Bruijn graph** over the **minimizer alphabet**.

Nodes = k-min-mers,

Edges = exact overlaps between k-1 minimizers

Applied to whole-genome *de novo* assembly

From accurate HiFi (< 1% error-rate) reads



Whole human PacBio HiFi (HG002) 50x coverage:

Tool name	Peregrine	hifiasm	rust-mdbg
Wall-clock time	14h8m	58h41m	**10m23s**
Memory usage	188 GB	195 GB	**10 GB**
# contigs	8109	431	805
NG50 (Mbp)	18.2	88.0	16.1
Genome fraction	97.0%	94.2%	95.5%

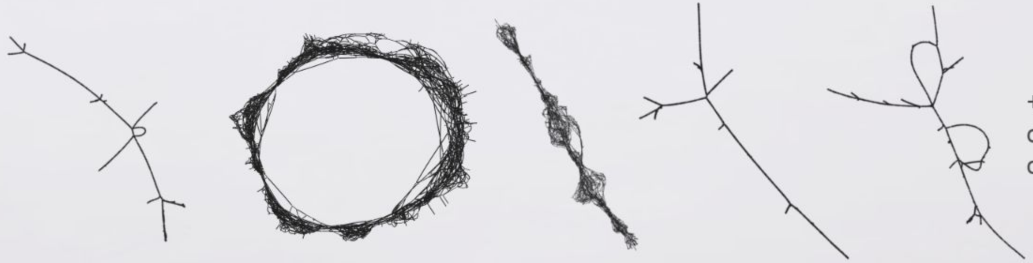
Results: Pangenome graph of 661,405 bacterial genomes

Data from Blackwell et al, 2021:

```
2.9T 661k_assemblies.fa
1.6T 661k_assemblies.fa.lz4
```

```
rust-mdbg -k 10 -l 12 --density 0.001 --minabund 1 661k_assemblies.fa.lz4
```

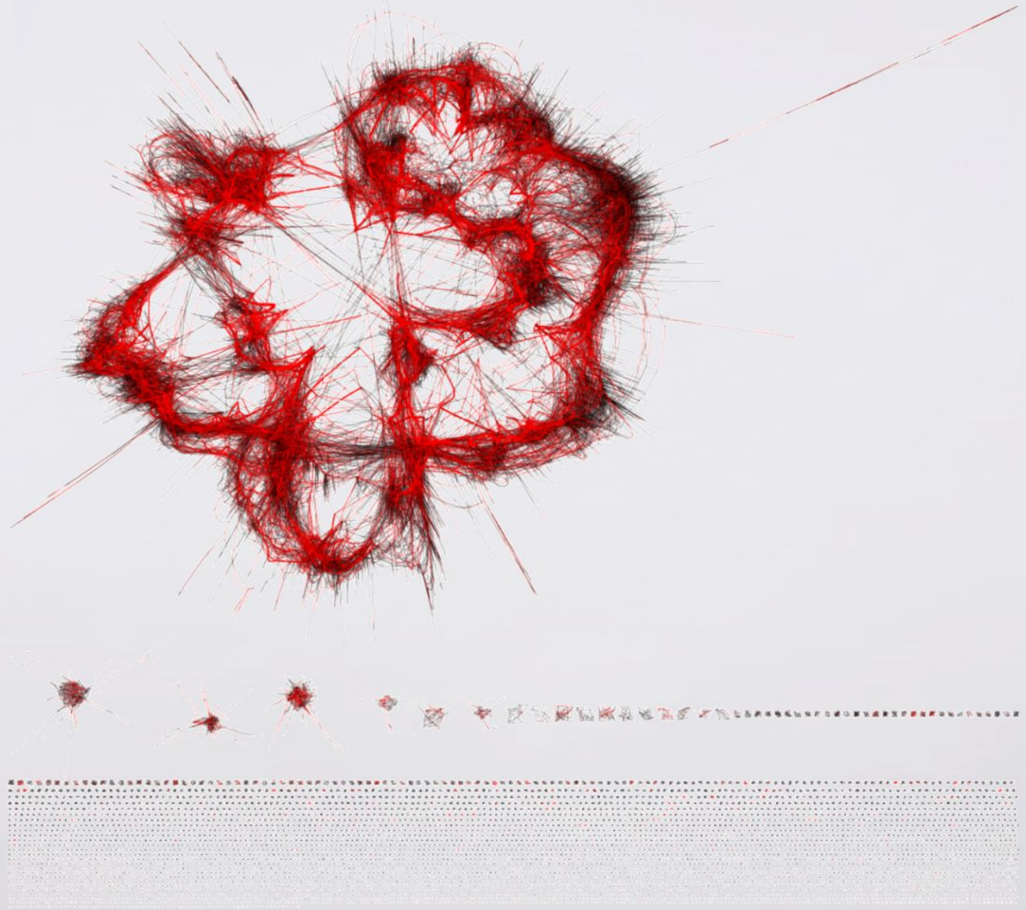
Largest 5 connected components:



+ 725,820 connected components

Taxons in component	18	22	4	22	10
Dominant species	*Mycobacterium tuberculosis*	*Salmonella enterica*	*Burkholderia gladioli*	*Pseudomonas protegens*	*Cupriavidus alkaliphilus*

Results: Pangenome graph of 160,000 E. coli genomes



mdBG behind-the-scenes history

Summer 2019

Idea,
1st prototype

Doesn't work for **Nanopore**, project dropped

But worked fine with no error

......

Late Spring 2020

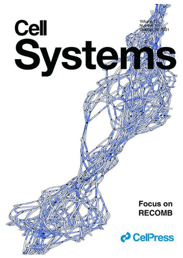
Baris Ekim wants a project, gave mdBG

Focus on error-correcting for Nanopore

Summer 2020

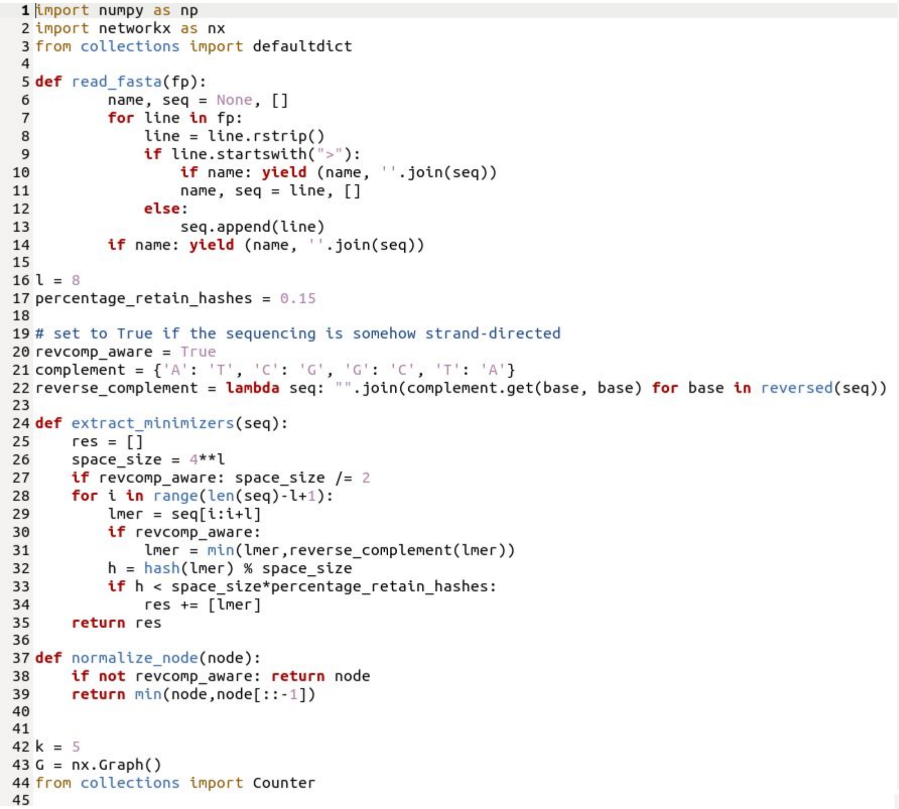
Pivot to uncorrected **HiFi**

Fall 2020

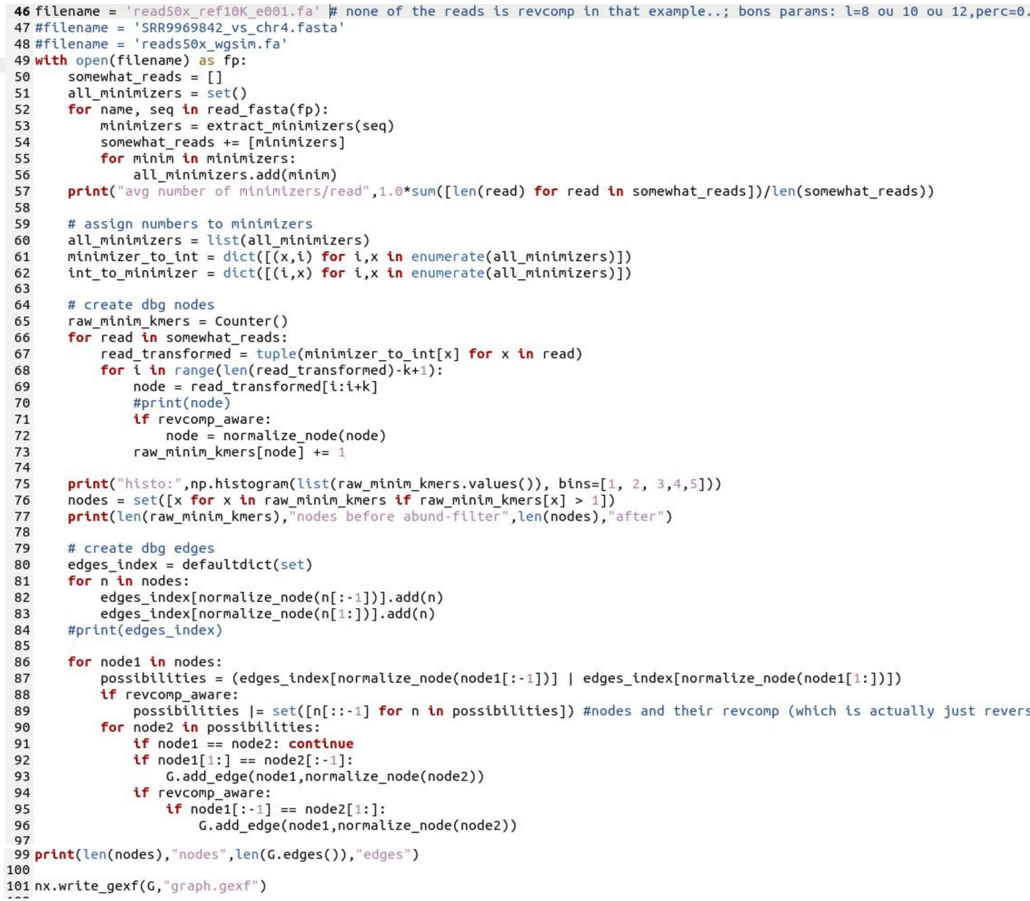


At the same time:
MBG, T2T,
LJA (competitor), ..

mDBG circa 2019



```python
import numpy as np
import networkx as nx
from collections import defaultdict

def read_fasta(fp):
        name, seq = None, []
        for line in fp:
            line = line.rstrip()
            if line.startswith(">"):
                if name: yield (name, ''.join(seq))
                name, seq = line, []
            else:
                seq.append(line)
        if name: yield (name, ''.join(seq))

l = 8
percentage_retain_hashes = 0.15

# set to True if the sequencing is somehow strand-directed
revcomp_aware = True
complement = {'A': 'T', 'C': 'G', 'G': 'C', 'T': 'A'}
reverse_complement = lambda seq: "".join(complement.get(base, base) for base in reversed(seq))

def extract_minimizers(seq):
    res = []
    space_size = 4**l
    if revcomp_aware: space_size /= 2
    for i in range(len(seq)-l+1):
        lmer = seq[i:i+l]
        if revcomp_aware:
            lmer = min(lmer,reverse_complement(lmer))
        h = hash(lmer) % space_size
        if h < space_size*percentage_retain_hashes:
            res += [lmer]
    return res

def normalize_node(node):
    if not revcomp_aware: return node
    return min(node,node[::-1])


k = 5
G = nx.Graph()
from collections import Counter

```



```python
filename = 'read50x_ref10K_e001.fa' # none of the reads is revcomp in that example..; bons params: l=8 ou 10 ou 12,perc=0.
#filename = 'SRR9969842_vs_chr4.fasta'
#filename = 'reads50x_wgsim.fa'
with open(filename) as fp:
    somewhat_reads = []
    all_minimizers = set()
    for name, seq in read_fasta(fp):
        minimizers = extract_minimizers(seq)
        somewhat_reads += [minimizers]
        for minim in minimizers:
            all_minimizers.add(minim)
    print("avg number of minimizers/read",1.0*sum([len(read) for read in somewhat_reads])/len(somewhat_reads))

    # assign numbers to minimizers
    all_minimizers = list(all_minimizers)
    minimizer_to_int = dict([(x,i) for i,x in enumerate(all_minimizers)])
    int_to_minimizer = dict([(i,x) for i,x in enumerate(all_minimizers)])

    # create dbg nodes
    raw_minim_kmers = Counter()
    for read in somewhat_reads:
        read_transformed = tuple(minimizer_to_int[x] for x in read)
        for i in range(len(read_transformed)-k+1):
            node = read_transformed[i:i+k]
            #print(node)
            if revcomp_aware:
                node = normalize_node(node)
            raw_minim_kmers[node] += 1

    print("histo:",np.histogram(list(raw_minim_kmers.values()), bins=[1, 2, 3,4,5]))
    nodes = set([x for x in raw_minim_kmers if raw_minim_kmers[x] > 1])
    print(len(raw_minim_kmers),"nodes before abund-filter",len(nodes),"after")

    # create dbg edges
    edges_index = defaultdict(set)
    for n in nodes:
        edges_index[normalize_node(n[:-1])].add(n)
        edges_index[normalize_node(n[1:])].add(n)
    #print(edges_index)

    for node1 in nodes:
        possibilities = (edges_index[normalize_node(node1[:-1])] | edges_index[normalize_node(node1[1:])])
        if revcomp_aware:
            possibilities |= set([n[::-1] for n in possibilities]) #nodes and their revcomp (which is actually just revers
        for node2 in possibilities:
            if node1 == node2: continue
            if node1[1:] == node2[:-1]:
                G.add_edge(node1,normalize_node(node2))
            if revcomp_aware:
                if node1[:-1] == node2[1:]:
                    G.add_edge(node1,normalize_node(node2))

print(len(nodes),"nodes",len(G.edges()),"edges")

nx.write_gexf(G,"graph.gexf")
```

mdBG takeaways

Projects can be stuck

Ideas can pivot

Big leaps (should) happen quick

Post-hoc motivation

mdBG/Minia:

- Pre-hoc: push dBG space optimization
- Post-hoc: more efficient assembly by 10x

Additional motivation:

- Make a theoretical advance in one of the 2 basic bioinfo problems
- Personal interests

mdBG future

(Not really human assembly)

- Pangenomics
- Metagenomics assembly (Benoit et al)
- Alignment (Mapquik: Ekim et al)

The tumultuous fate of ideas

On the minimizer front:

- **1995** Locally Consistent Parsing (Sahinalp, Vishkin TR)
- **2004** Minimizers (Roberts)
- **2012** SparseAssembler (Ye), SCALCE (Hach)
- **2013** Minimum Substring Partitioning (Y Li)
- **2014** BCALM1 (Limasset)
- **2015** KMC2 (Kokot)

from concept to
1st application on
seq data: 17 years

The tumultuous fate of ideas

On the dBG front:

- **1894** "Question 48" Camille Flye Sainte-Marie
- **1989** “1-Tuple DNA Sequencing: Computer Analysis” Pevzner
- **1995** “A new algorithm for DNA sequence assembly” Idury and Waterman
- **2001** Euler assembler
- Velvet **2008**, Soapdenovo **2009,** SPAdes **2012**

On the dBG data structures front:

- **2009** Soapdenovo
- **2011** Conway-Bromage, khmer
- **2012** Minia
- .. (many more in “*A tale of optimizing the space taken by de Bruijn graphs*”, CiE 2021)

The tumultuous fate of ideas

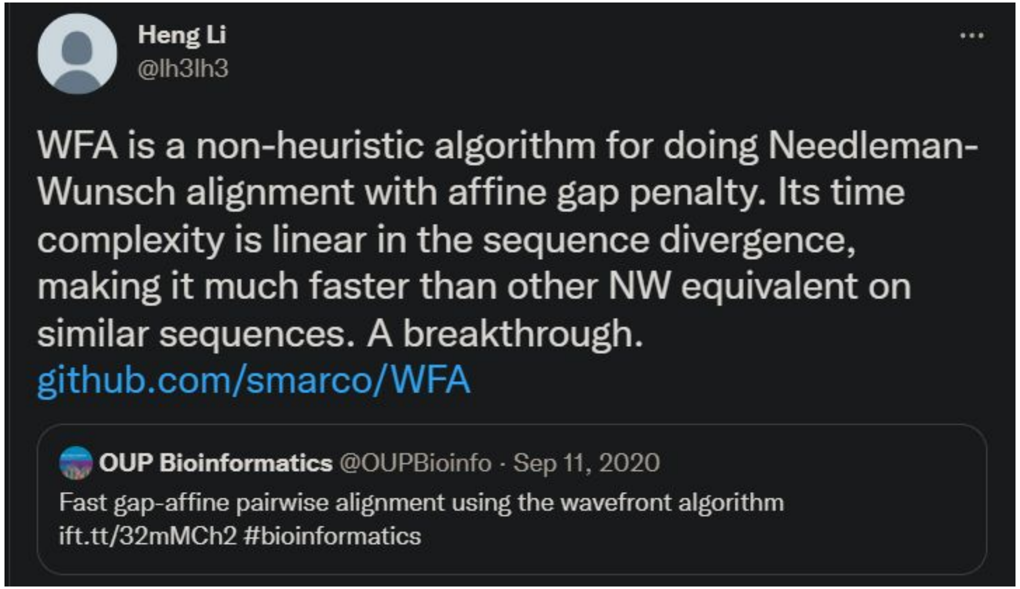
On the mdBG front:

- Failed prototype **2019**
- Ekim *et al* **2021**
- mapquik **2022** (in prep)
- metaMDBG **2023** (in prep)

;)

Questions? (before moving to part 2)

WFA



Heng Li
@lh3lh3

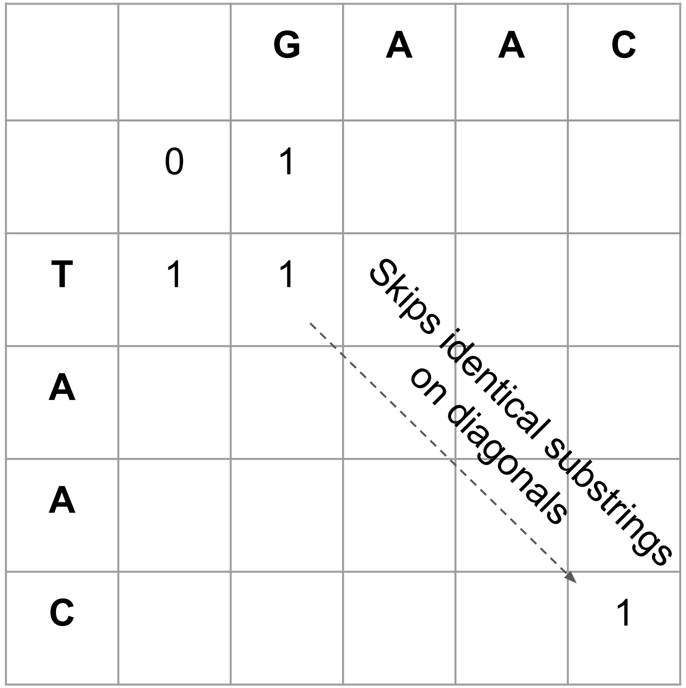
WFA is a non-heuristic algorithm for doing Needleman-Wunsch alignment with affine gap penalty. Its time complexity is linear in the sequence divergence, making it much faster than other NW equivalent on similar sequences. A breakthrough.
github.com/smarco/WFA

OUP Bioinformatics @OUPBioinfo · Sep 11, 2020
Fast gap-affine pairwise alignment using the wavefront algorithm
ift.tt/32mMCh2 #bioinformatics

WFA primer

		G	A	A	C
	0	-1	-2	-3	-4
T	-1	**-1**	-2	-3	-4
A	-2	-2	**0**	-1	-2
A	-3	-3	-1	**1**	0
C	-4	-4	-2	0	**2**

Needleman-Wunsch



		G	A	A	C
	0	1			
T	1	1			
A					
A					
C					1

WFA

WFA facts

- $O(ns)$ time
  - n = sequences length
  - s = score of best alignment
- Designed only for affine gap penalties
- $O(s^2)$ space (BiWFA: $O(s)$ space)

The tumultuous fate of ideas

On the O(nd) front:

- d = edit distance
- **1985** Ukkonen (ED)
- **1986** Myers (ED)
- **1989** Landau-Vishkin (ED)
- **2017** Xin et al (affine gap)
- **2021** Marco-Sola et al (affine gap)
- **2022** Koerkamp, Ivanov (ED, O($n^{1.08}$))

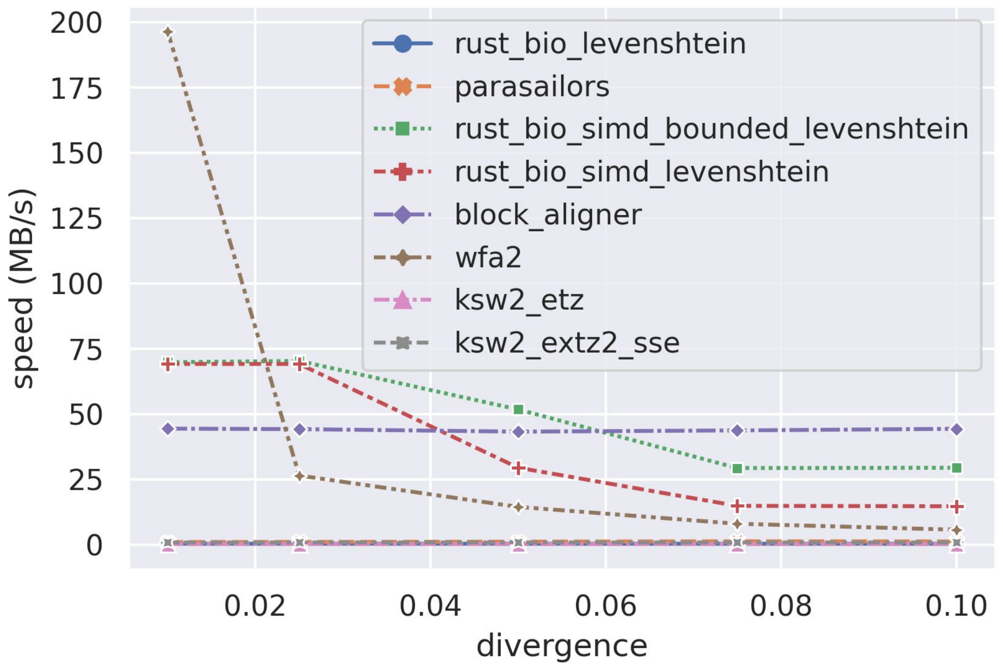
"The search for a similar algorithm for linear or affine gap penalties took three decades."
(- Heng Li)

On the O(s) space front:

- **1975** Hirschberg
- **1986** Myers
- **2022** BiWFA

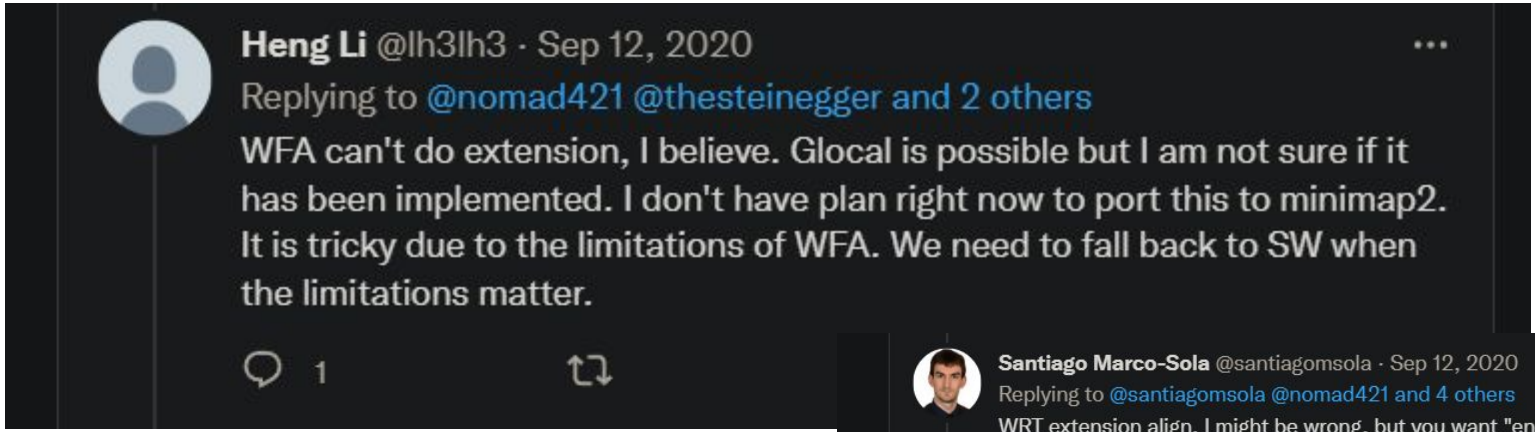
Source: https://github.com/lh3/miniwfa

Alignment speed SOTA



https://github.com/rchikhi/rust-alignbench/

WFA in minimap2 ?

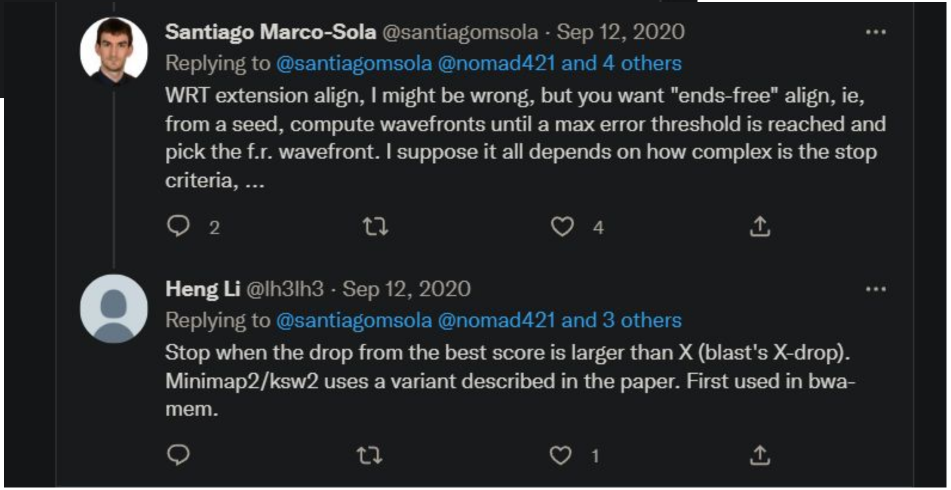


Heng Li @lh3lh3 · Sep 12, 2020

Replying to @nomad421 @thesteinegger and 2 others

WFA can't do extension, I believe. Glocal is possible but I am not sure if it has been implemented. I don't have plan right now to port this to minimap2. It is tricky due to the limitations of WFA. We need to fall back to SW when the limitations matter.

1



Santiago Marco-Sola @santiagomsola · Sep 12, 2020

Replying to @santiagomsola @nomad421 and 4 others

WRT extension align, I might be wrong, but you want "ends-free" align, ie, from a seed, compute wavefronts until a max error threshold is reached and pick the f.r. wavefront. I suppose it all depends on how complex is the stop criteria, ...

2 4

Heng Li @lh3lh3 · Sep 12, 2020

Replying to @santiagomsola @nomad421 and 3 others

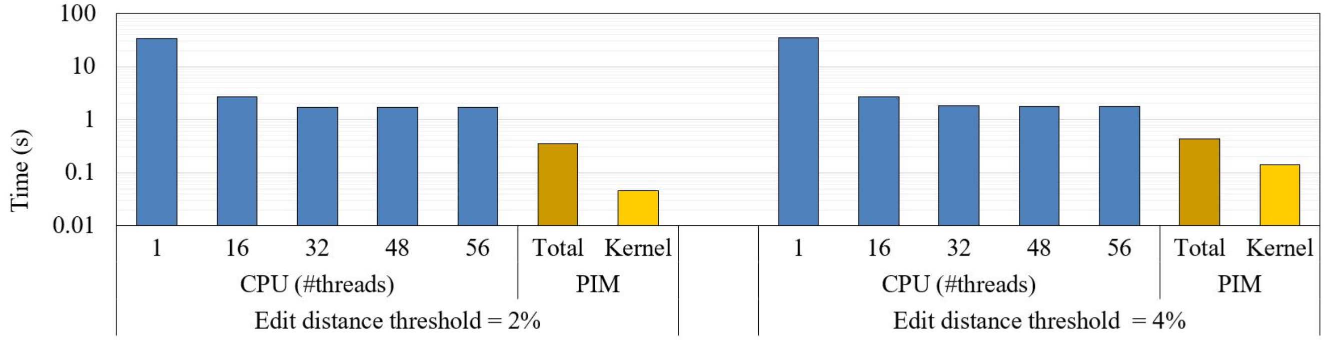
Stop when the drop from the best score is larger than X (blast's X-drop). Minimap2/ksw2 uses a variant described in the paper. First used in bwa-mem.

1

WFA on specialized hardware (PIM)

Evaluation Model & Results

- Read Length: 100bp, Edit distance thresholds: 2% and 4%, Number of read pairs: 5Million
- CPU: Intel® Xeon® Gold 5120 Processor: 56 CPU threads, and 64 GB Memory
- PIM: 2,560 UPMEM DPUs at 425MHz and a total of 150GB MRAM



4x

High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory

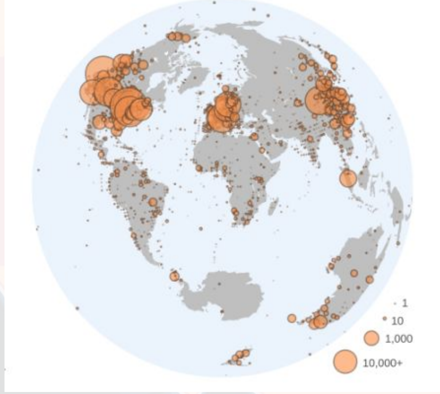
Safaa Diab[1], Amir Nassereldine[1], Mohammed Alser[2], Juan Gómez Luna[2], Onur Mutlu[2], Izzat El Hajj[1]

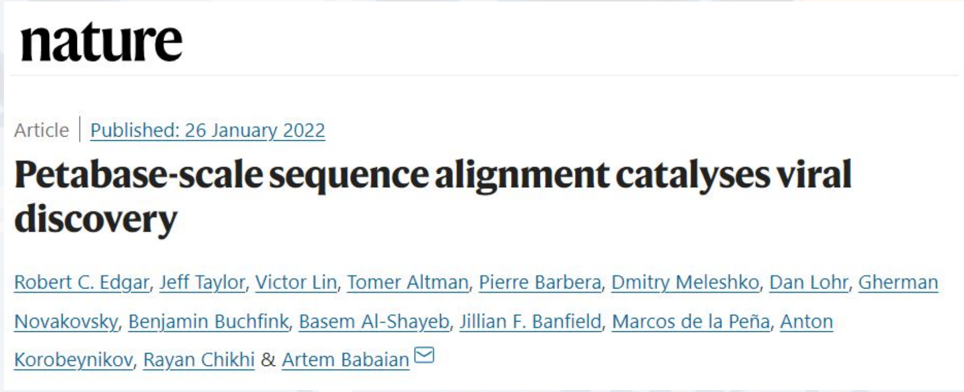
Serratus: Petabase-scale sequence search

- 5M RNA-seqs aligned (10 PB)
  50k assemblies, 28,000 vCPUs on AWS,
  w/ highly optimized align
  & fast download from S3

- Discovery of a new
  coronavirus species

- 10x expansion of RNA
  viruses species




nature

Article | Published: 26 January 2022

Petabase-scale sequence alignment catalyses viral discovery

Robert C. Edgar, Jeff Taylor, Victor Lin, Tomer Altman, Pierre Barbera, Dmitry Meleshko, Dan Lohr, Gherman Novakovsky, Benjamin Buchfink, Basem Al-Shayeb, Jillian F. Banfield, Marcos de la Peña, Anton Korobeynikov, Rayan Chikhi & Artem Babaian

Thanks lots to SeqBim organizers!

Thank you for your attention!