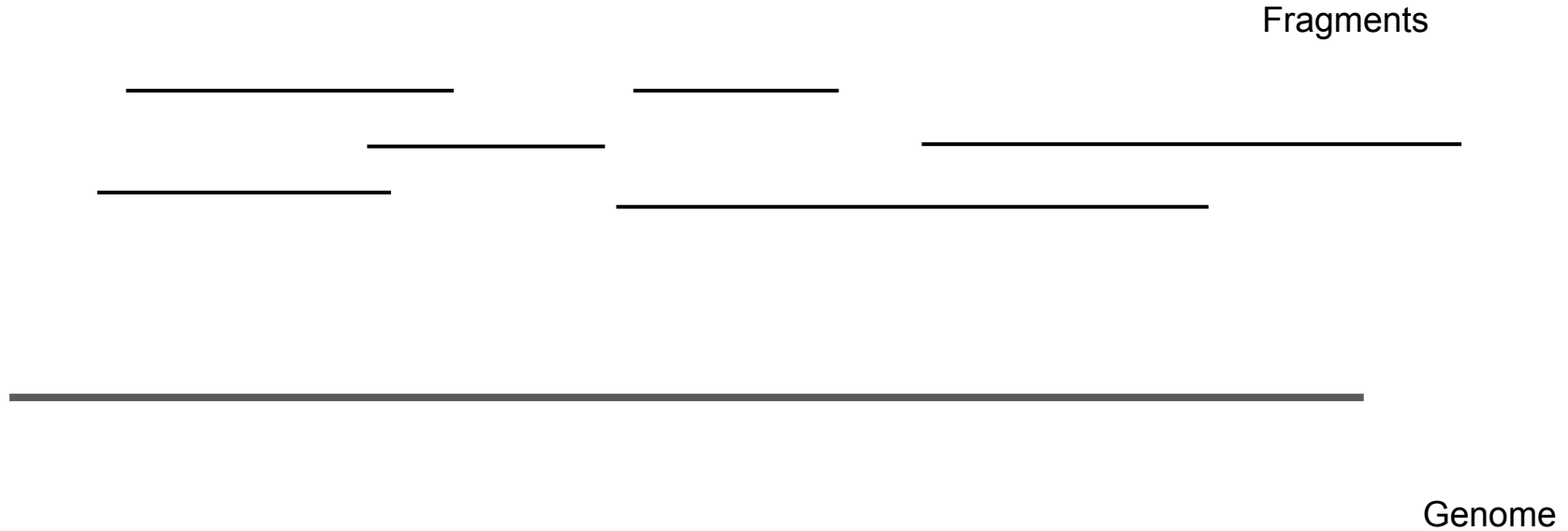


# Assembly of 10x Genomics GemCode reads

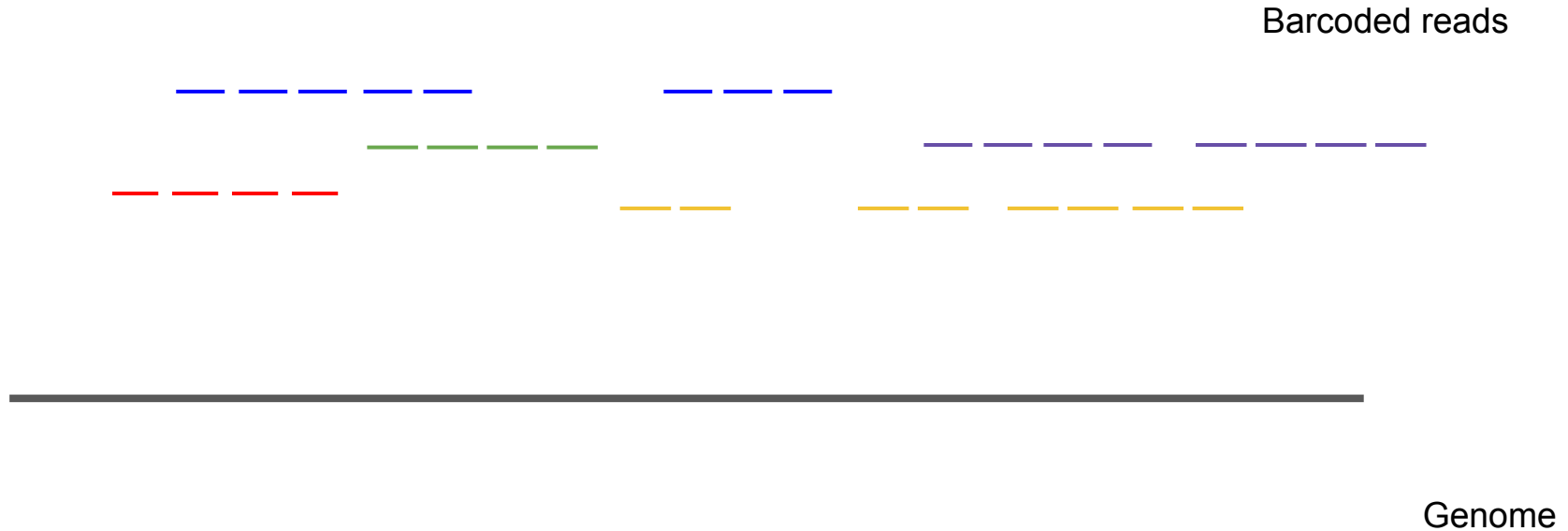
Anton Bankevich, Ivan Tolstogonov

# GemCode barcoded sequencing



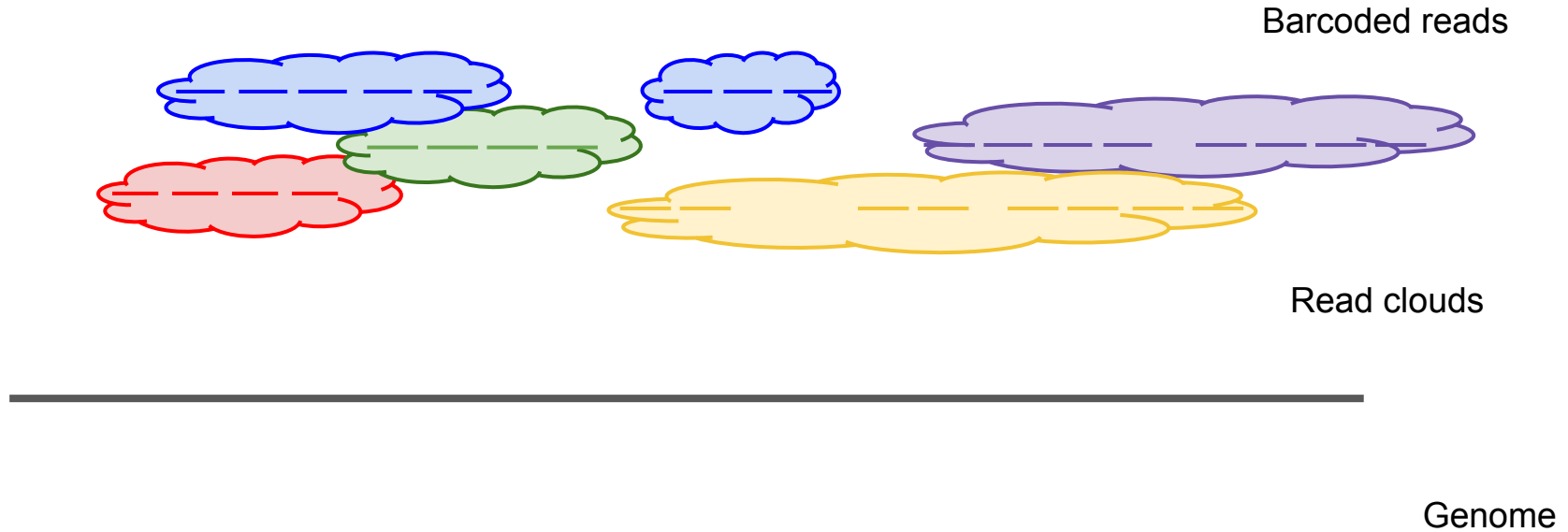
- DNA is sheared into long fragments (up to 50 Kb)

# GemCode barcoded sequencing



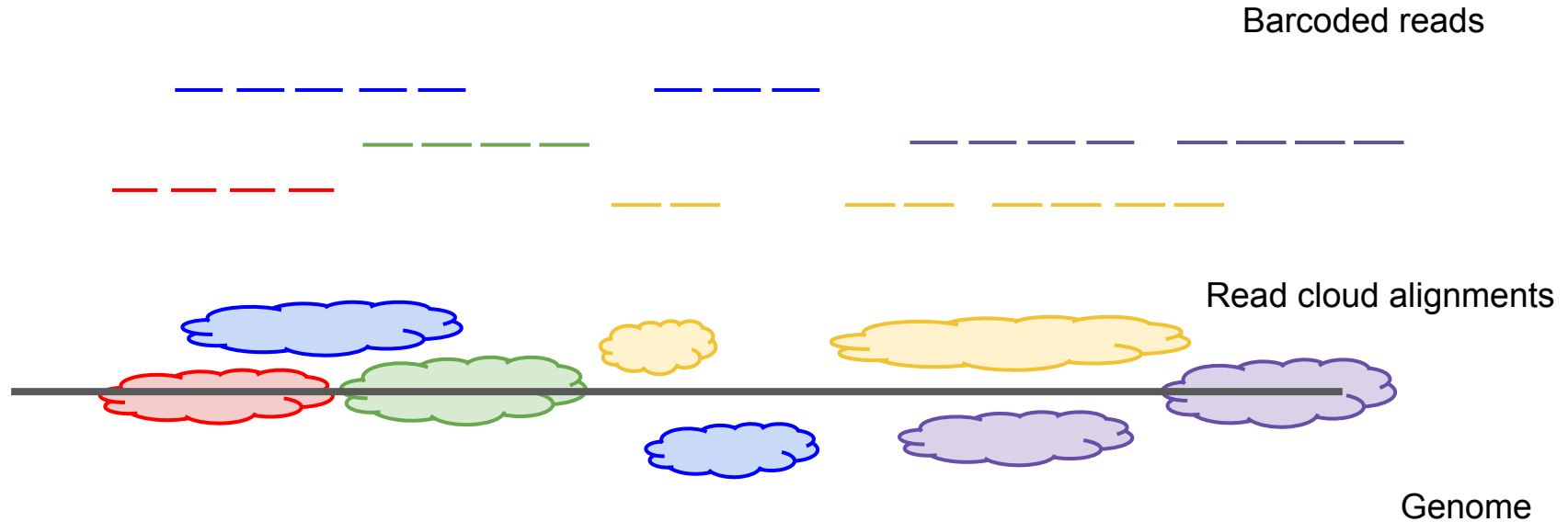
- DNA is sheared into long fragments (up to 50 Kb)
- Fragments are barcoded and sequenced

# Read clouds



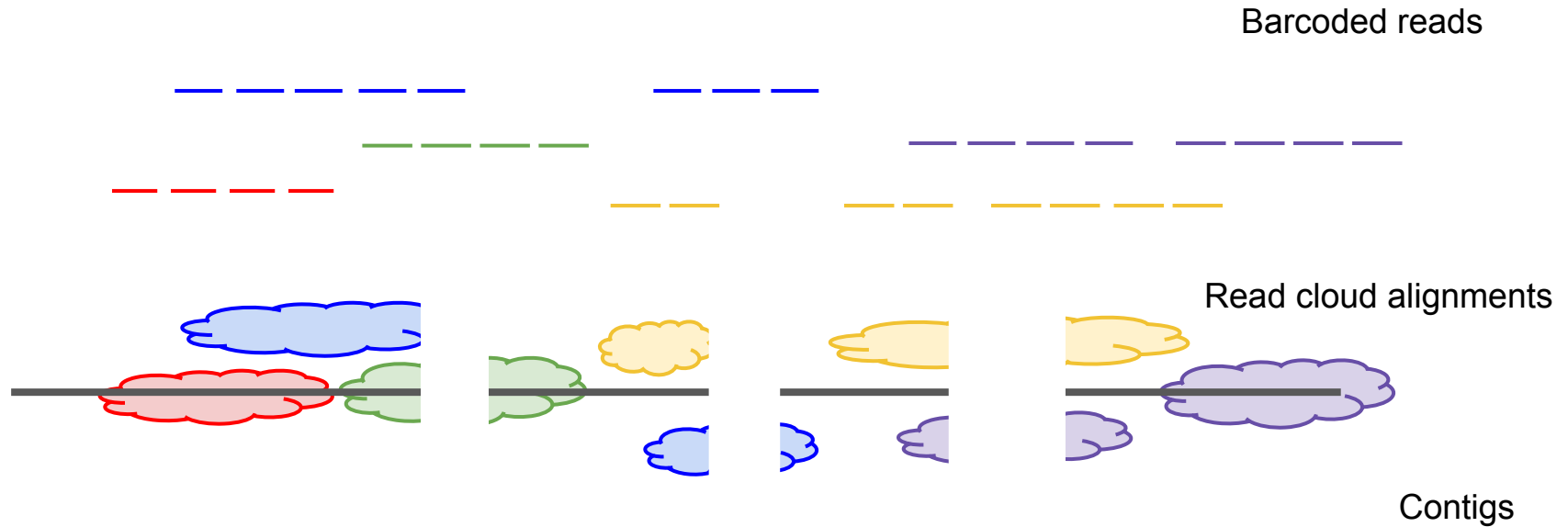
- DNA is sheared into long fragments (up to 50 Kb)
- Fragments are barcoded and sequenced to form read clouds

# Read cloud alignment



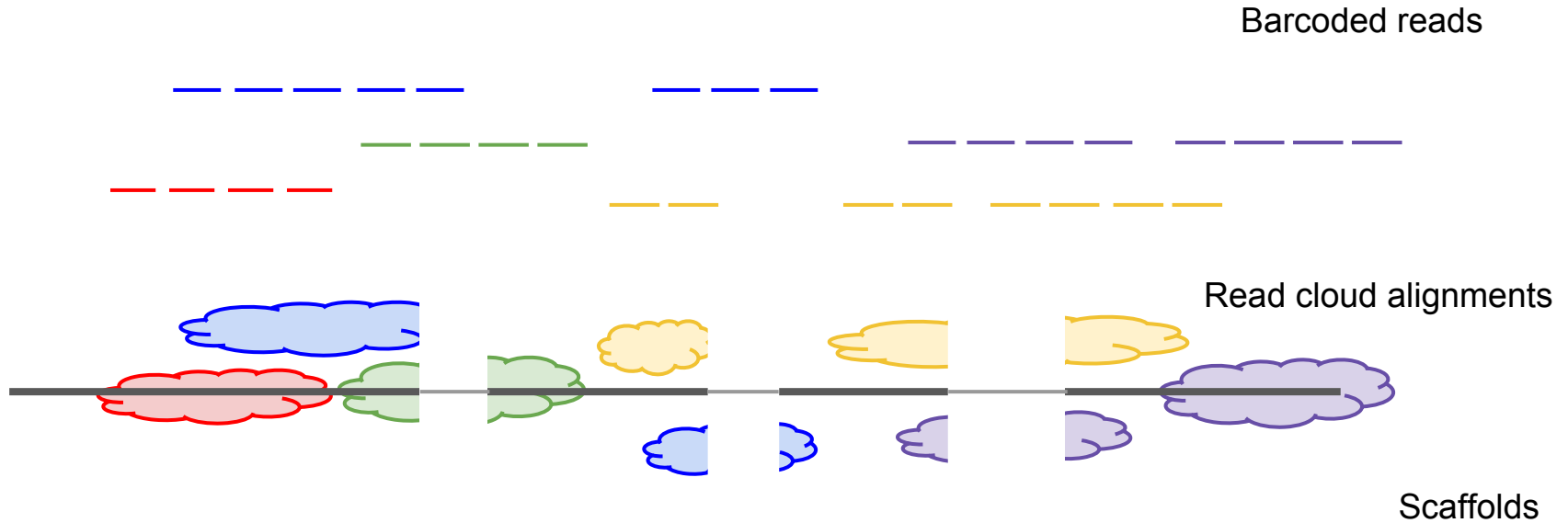
- DNA is sheared into long fragments (up to 50 Kb)
- Fragments are barcoded and sequenced to form read clouds
- Read clouds can be recovered by alignment

# Read cloud scaffolding



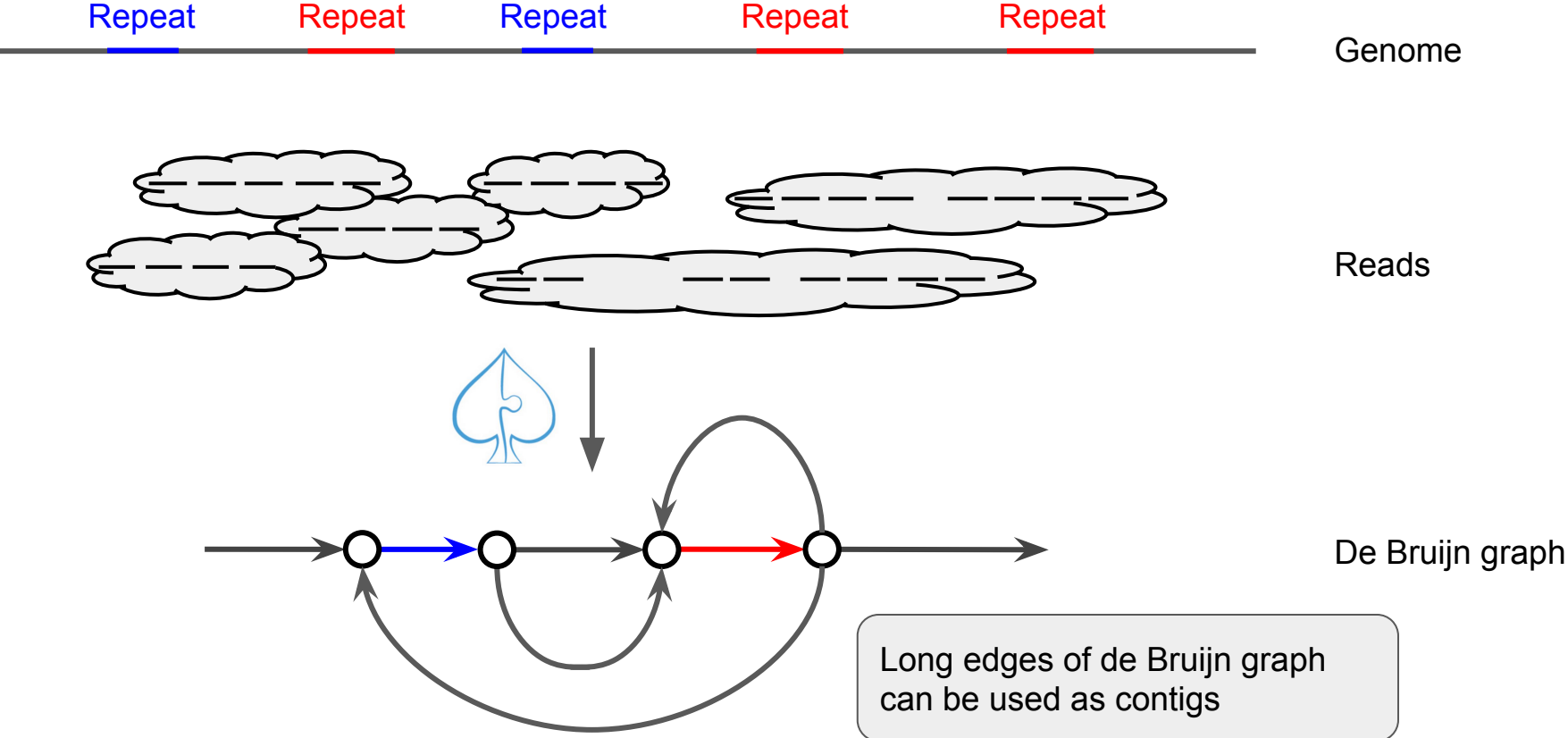
- Read clouds can be aligned to contigs

# Read cloud scaffolding



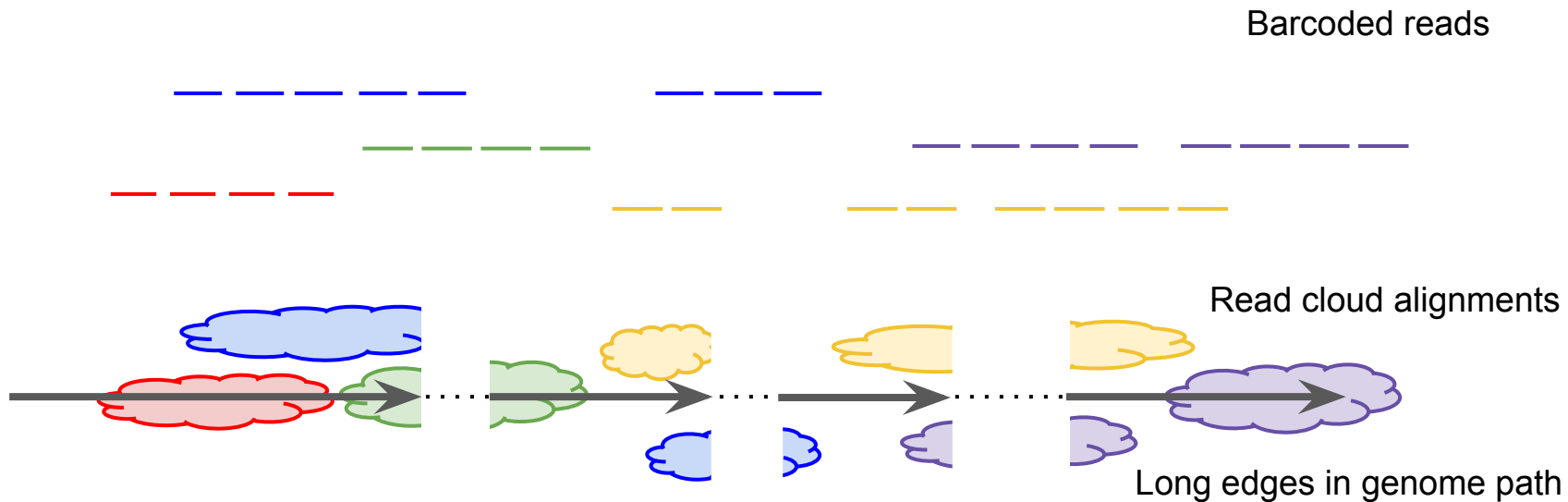
- Read clouds can be aligned to contigs
- Alignment of read clouds can be used for scaffolding

# De Bruijn graph



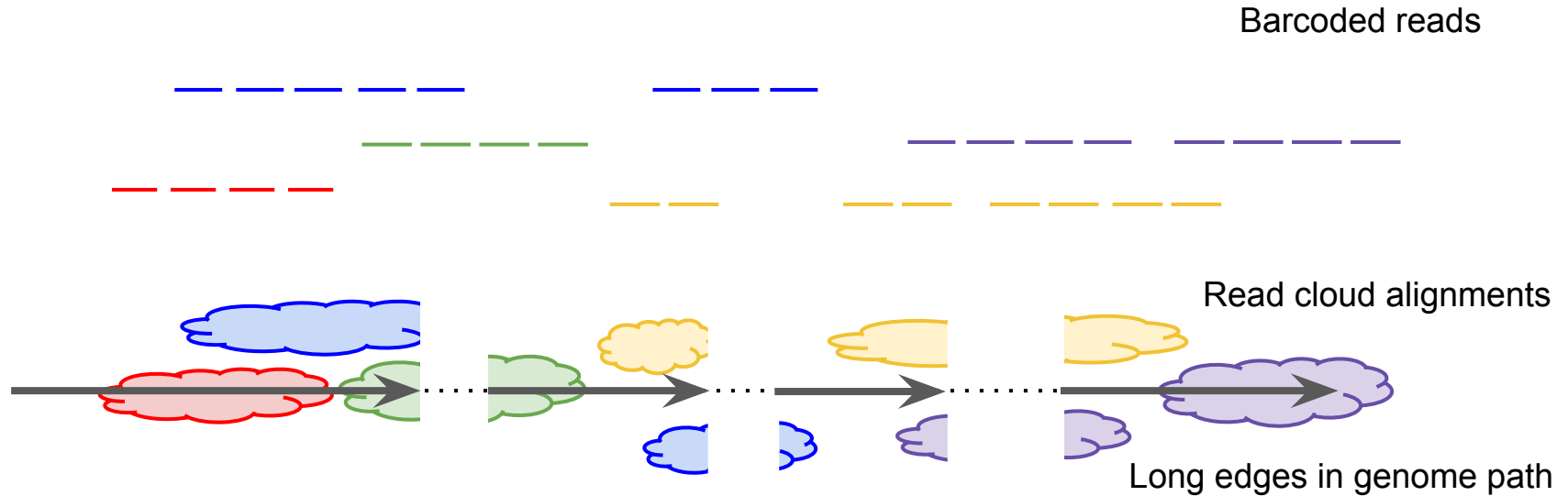


# Clouds over de Bruijn graph



- Clouds can be aligned to the long edges of the graph

# Read cloud scaffolding of de Bruijn graph

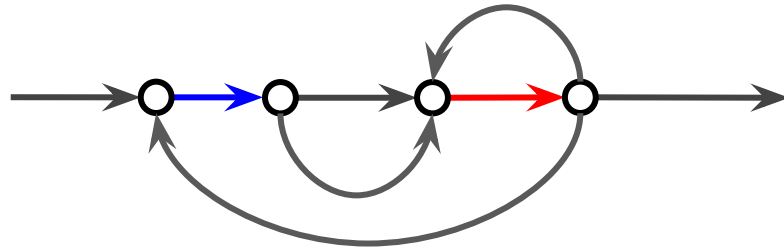


- Clouds can be aligned to the long edges of the graph
- For every edge we will try to find the next long edge in genome path

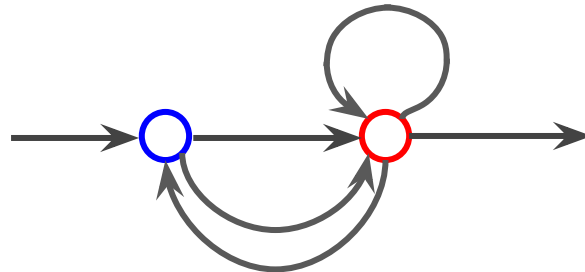
# Contracted de Bruijn graph

- **Vertices:** connected components of short edges of de Bruijn graph
- **Edges:** long edges of de Bruijn graph

**Motivation:** finding genome path through complex components of short edges is very difficult. Thus we limit our attention to long edges and connections between them

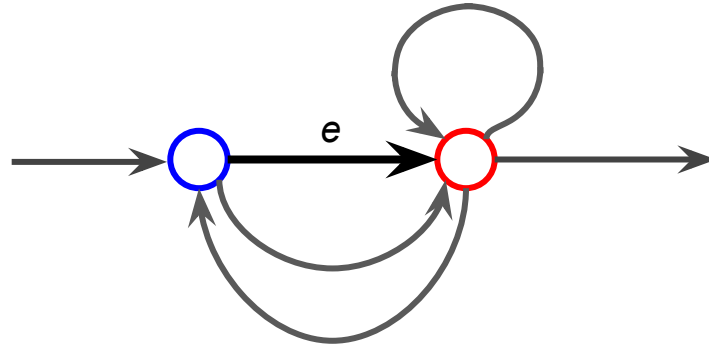


De Bruijn graph



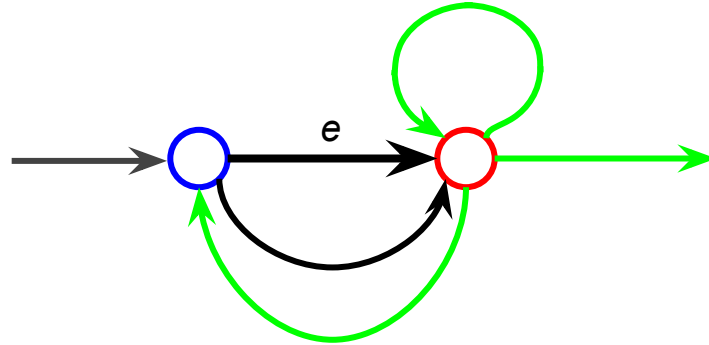
Contracted graph

# Scaffolding: next edge candidate selection



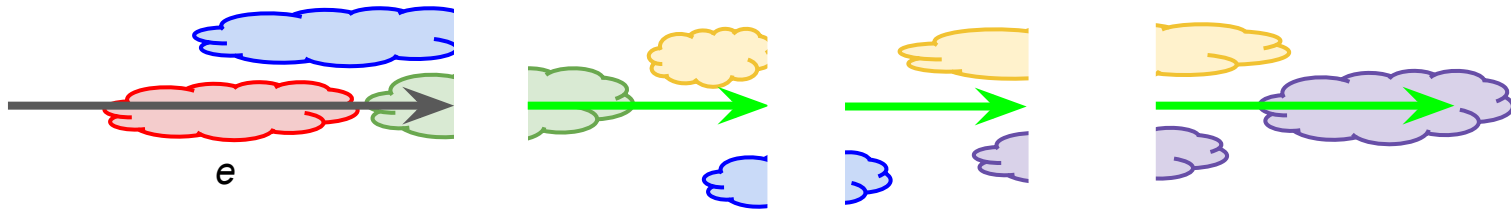
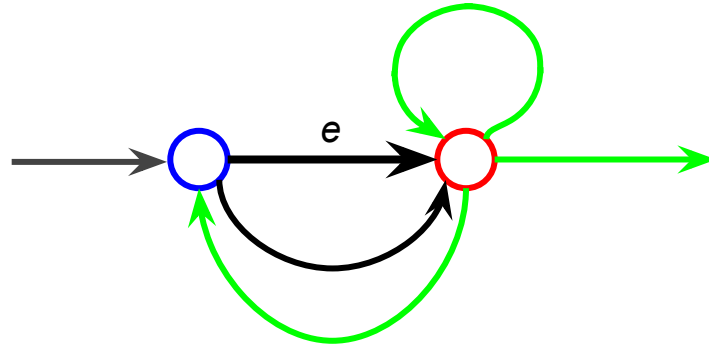
- We want to find the next long edge in the genome path after edge  $e$

# Scaffolding: next edge candidate selection



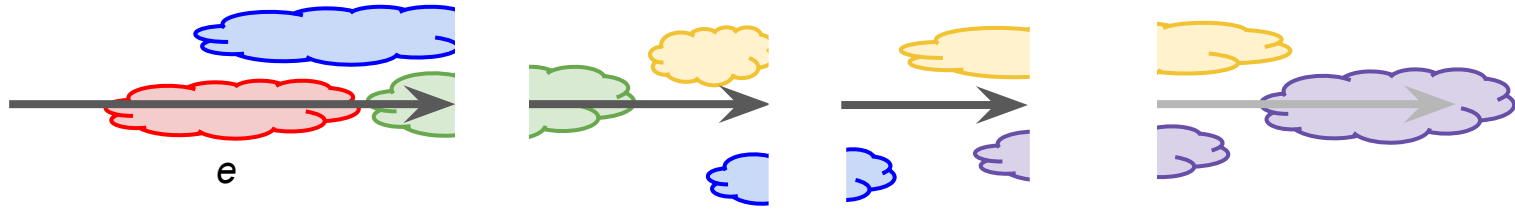
- We want to find the next long edge in the genome path after edge  $e$
- We limit a set of candidates to edges adjacent to  $e$  in contracted graph

# Scaffolding: next edge candidate selection



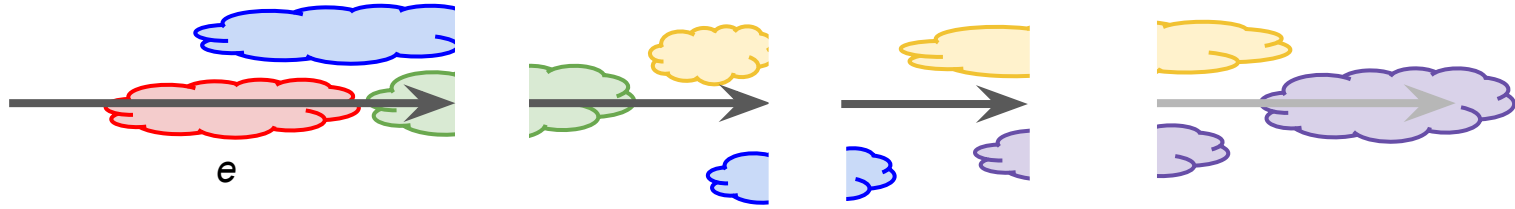
- We want to find the next long edge in the genome path after edge  $e$
- We limit a set of candidates to edges adjacent to  $e$  in contracted graph

# Shared barcodes pruning



- We discard candidates that share no or very few barcodes with  $e$

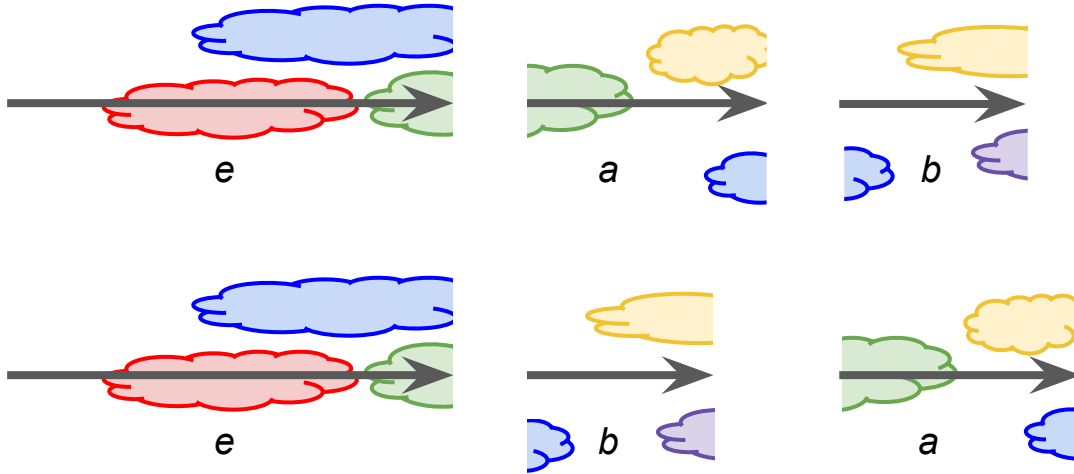
# Shared barcodes pruning



- We discard candidates that share no or very few barcodes with  $e$
- Remaining candidates are typically close to  $e$  in genome path
- Next step is to determine their order



# Edge order pruning

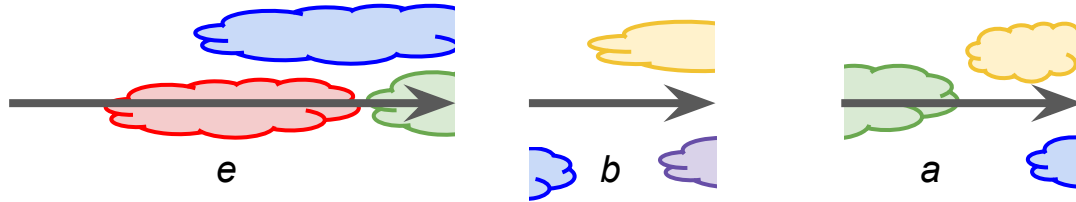


Correct ordering

Incorrect ordering: central edge does not contain green barcode

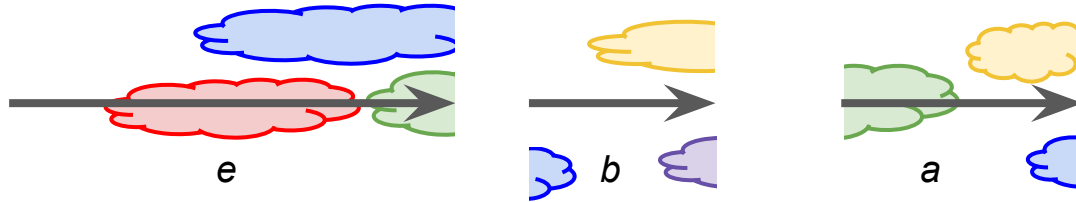
- Next edge should be located between **e** and every other candidate
- If edge **a** is located between **e** and **b** then **a** should contain every barcode shared by **e** and **b**

# Edge order pruning



- It is possible that green barcode may contain long gaps
- Or two green fragments are located near each other in the genome
- We need statistical model to deal with these cases

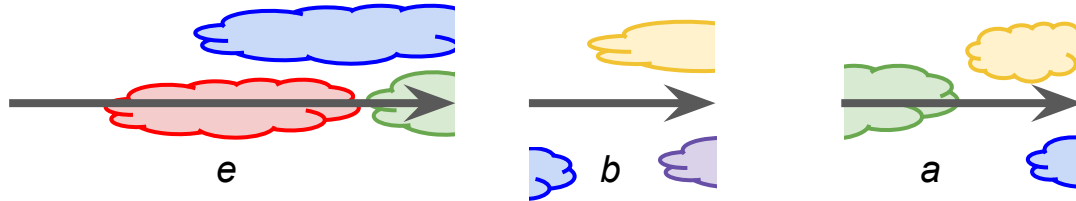
# Edge order pruning



We define:

- $Contradiction(\mathbf{e}, \mathbf{a}, \mathbf{b})$  : number of barcodes shared by  $\mathbf{e}$  and  $\mathbf{a}$  and not present on  $\mathbf{b}$
- $Intersection(\mathbf{e}, \mathbf{b})$  : number of barcodes shared by  $\mathbf{e}$  and  $\mathbf{b}$

# Edge order pruning

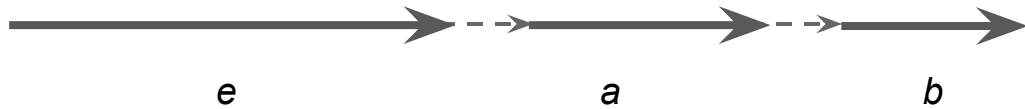


We define:

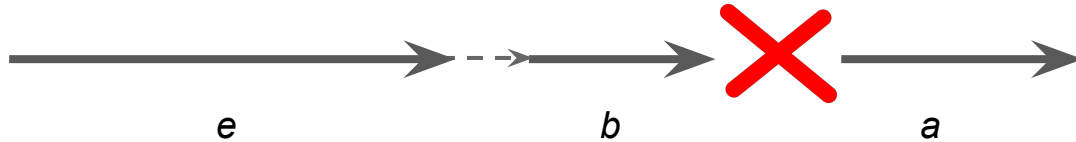
- $Contradiction(\mathbf{e}, \mathbf{a}, \mathbf{b})$  : number of barcodes shared by  $\mathbf{e}$  and  $\mathbf{a}$  and not present on  $\mathbf{b}$
- $Intersection(\mathbf{e}, \mathbf{b})$  : number of barcodes shared by  $\mathbf{e}$  and  $\mathbf{b}$

$P(Contradiction, Intersection \mid length(\mathbf{b}), cov(\mathbf{e}), cov(\mathbf{a}))$  is estimated using information from long edges

# Edge order pruning



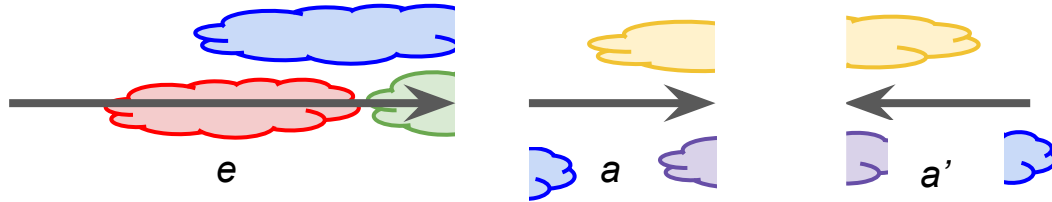
Correct ordering



Incorrect ordering: *a* is not reachable from *b*

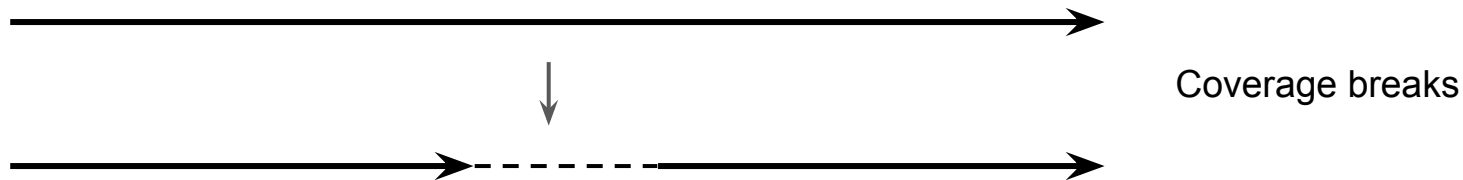
- Next edge could also be detected using graph topology
- If edge *a* is located between *e* and *b* then *a* and *b* should be adjacent in the contracted graph

# Problems: reverse complement edges



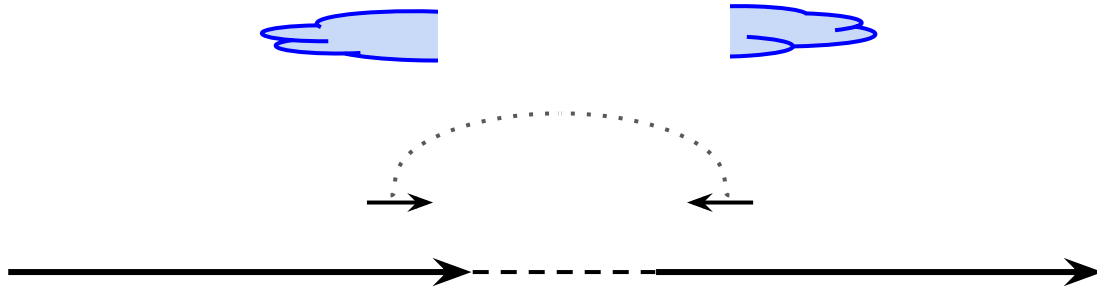
- Reverse complement edges have the same sets of barcodes
- We need specific criterias to find out the right order

# Problems: candidate search



- DBG is not perfect due to coverage breaks and simplification errors
- Often we can not reach any candidates using DBG edges

# Problems: coverage breaks



- DBG is not perfect due to coverage breaks and simplification errors
- Often we can not reach any candidates using DBG edges
- We can use paired end and read cloud information to traverse coverage breaks



# Validation: metagenomics

Simple metagenomic dataset which comprised of 10 isolates

Complete reference genomes unavailable, instead:

- draft assembly for every isolate
- 23 validated ribosomal operons

One of the draft assemblies was discarded since it was not covered by metagenomic assemblies. 9 remaining draft assemblies were provided as “references” to metaQUAST

We compared our results with Athena<sup>1</sup>, read cloud assembler aimed at metagenomic data

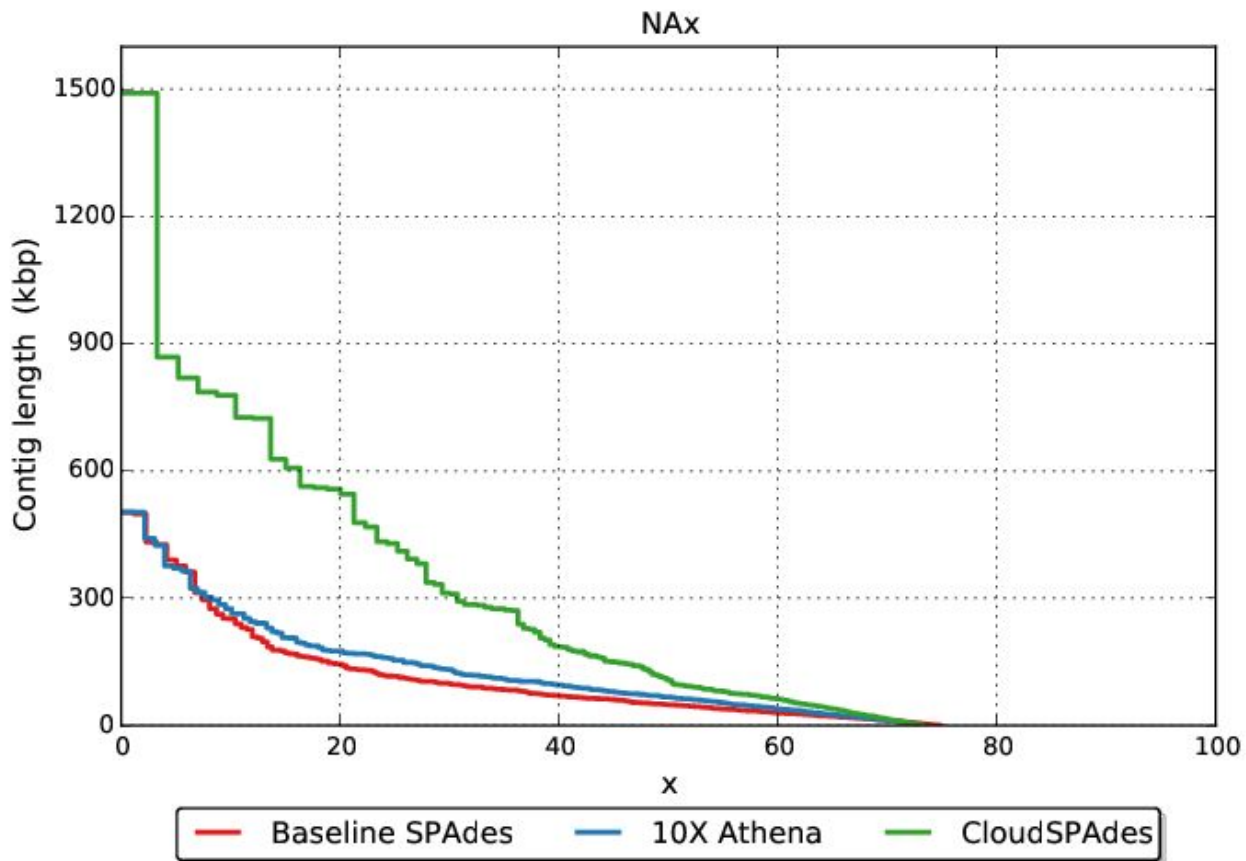
Bishara *et al* (2017). De novo assembly of microbial genomes from human gut metagenomes using barcoded short read sequences. *bioRxiv*.

# Reference assembly validation

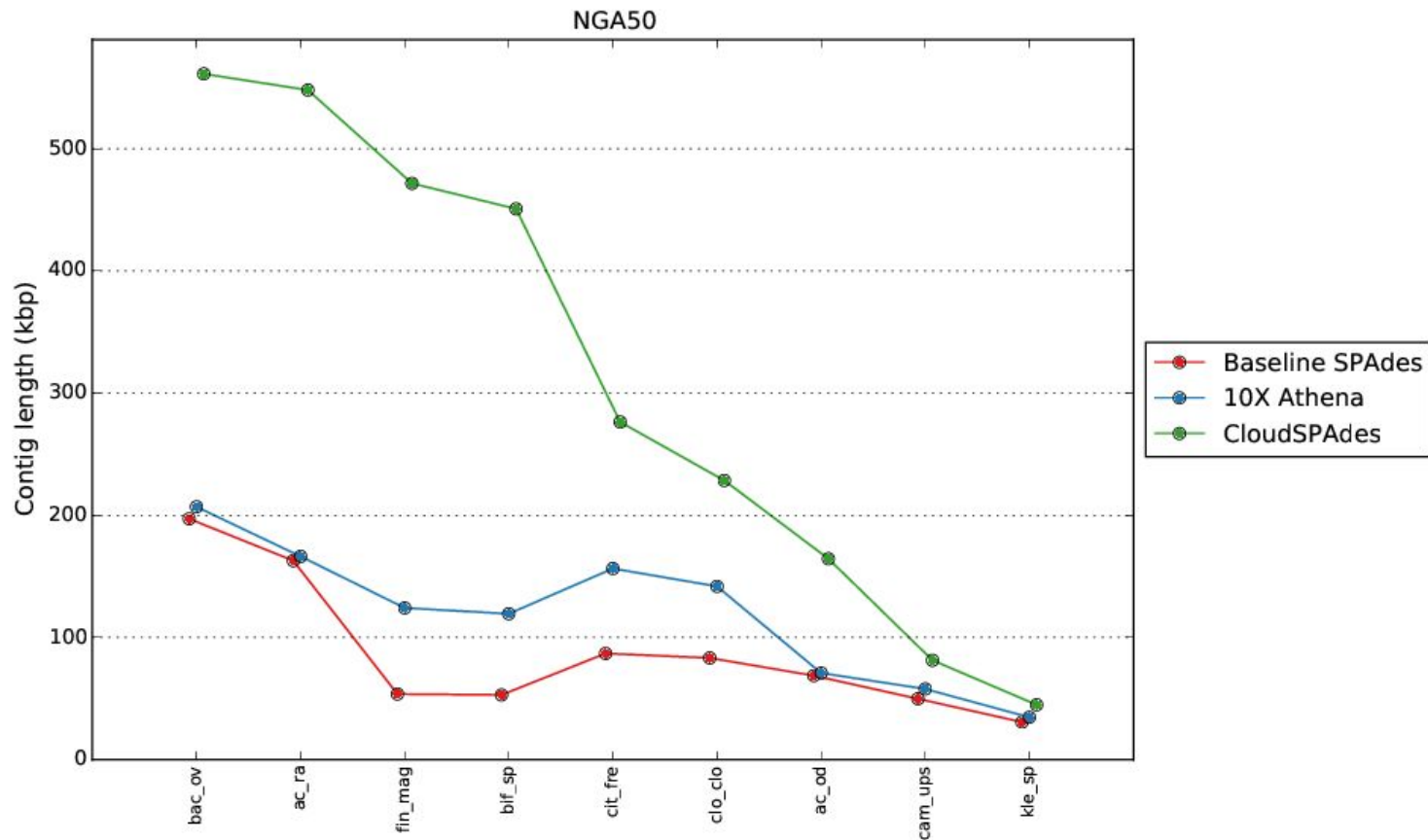
Assembly	Baseline SPAdes	10X Athena	CloudSPAdes
# contigs	4242	3700	3682
Largest contig	618498	506140	<b>1639058</b>
Total length	44596618	<b>47352283</b>	45423756
Reference length	34324828	34324828	34324828
N50	83644	114821	<b>271124</b>
# misassemblies	<b>16</b>	22	39
# N's per 100 kbp	27.79	<b>0.00</b>	997.58
Largest alignment	503176	503352	<b>1490986</b>
NA50	49220	67118	<b>108061</b>

cloudSPAdes: SPAdes assembly with read cloud repeat resolution module

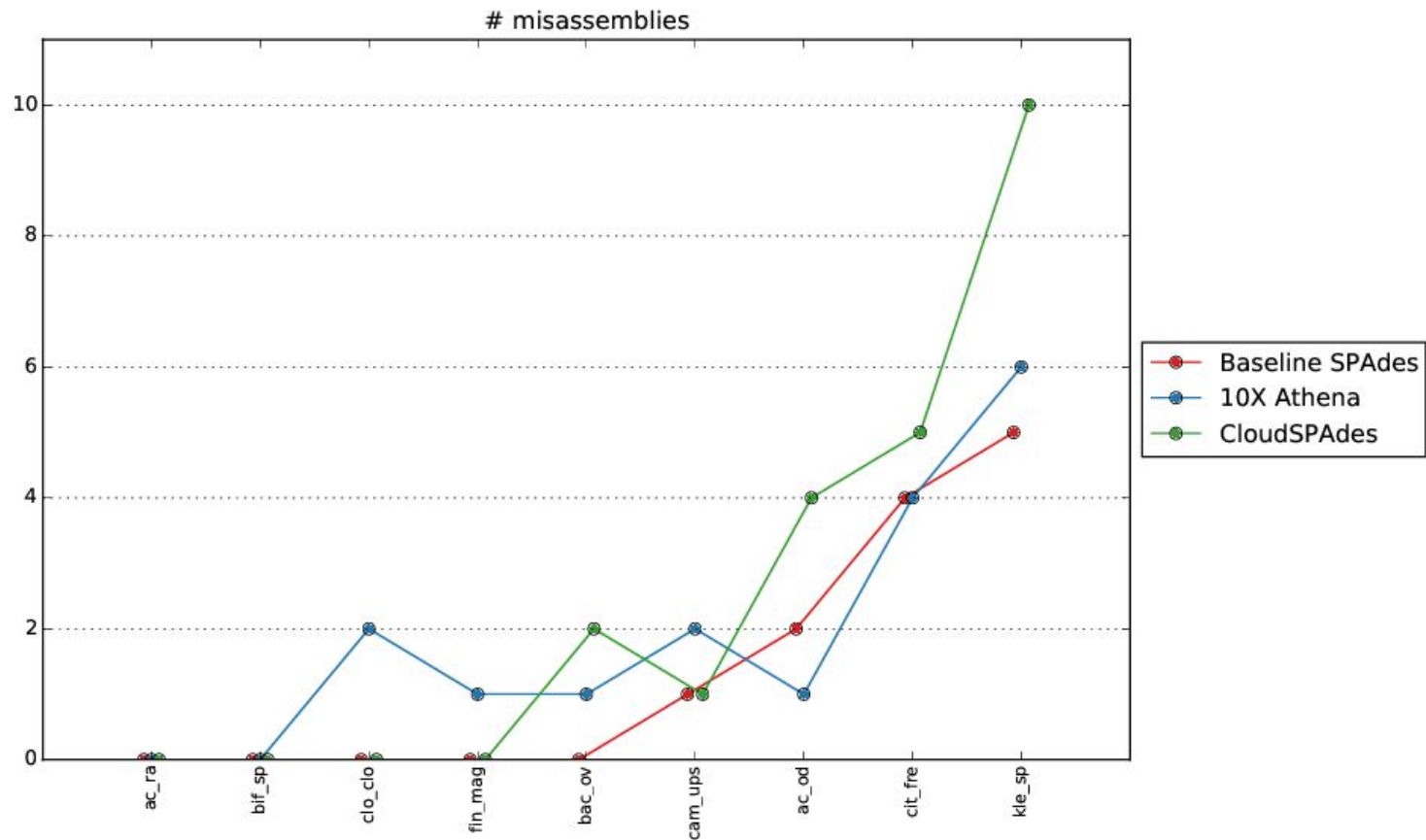
# NAx plot



# NGA50 per reference



# Misassemblies



# Summary

Advantages of cloudSPAdes assembly:

- Longer contigs

Disadvantages:

- More misassemblies
- Large fraction of N's

cloudSPAdes reconstructed 20 out of 23 operons

Thank you  
Questions?