HG-CoLoR: enHanced de bruijn Graph for the error COrrection of LOng Reads

Pierre Morisse, Thierry Lecroq and Arnaud Lefebvre
`pierre.morisse2@univ-rouen.fr`

Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes

November 7, 2017

Plan

Third Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

Third Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

Third Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

Third Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

Third Generation Sequencing

- Recently, Third Generation Sequencing technologies started to develop
- Two main technologies: Pacific Biosciences and Oxford Nanopore
- Allow the sequencing of longer reads (several thousand of bases)
- Very useful to resolve assembly problems for large and complex genomes
- Much higher error rate, around 15% for Pacific Biosciences and up to 30% for Oxford Nanopore

Problem

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

Problem

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

Problem

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

Problem

- Due to their high error rate, error correction of long reads is mandatory
- Various methods already exist for the correction of short reads, but are not applicable to long reads
- Forces the development of new error correction methods
- Two main categories: self-correction and hybrid correction

Motivations

- Most hybrid methods focus on reducing the error rate...
- ...But yield bad assembly results
- $\Rightarrow$ Focus more on assembly results

Motivations

- Most hybrid methods focus on reducing the error rate...
- ...But yield bad assembly results
- $\Rightarrow$ Focus more on assembly results

Motivations

- Most hybrid methods focus on reducing the error rate...
- ...But yield bad assembly results
- $\Rightarrow$ Focus more on assembly results

Inspiration

- NaS [Madoui et al., 2015]
- Yields highly contiguous assembly results
- Does not locally correct erroneous regions
- Uses long reads as templates to generate corrected long reads from assemblies of short reads
- Requires the mapping of the short reads both on the long reads and against each other

Inspiration

- NaS [Madoui et al., 2015]
- Yields highly contiguous assembly results
- Does not locally correct erroneous regions
- Uses long reads as templates to generate corrected long reads from assemblies of short reads
- Requires the mapping of the short reads both on the long reads and against each other

Inspiration

- NaS [Madoui et al., 2015]
- Yields highly contiguous assembly results
- Does not locally correct erroneous regions
- Uses long reads as templates to generate corrected long reads from assemblies of short reads
- Requires the mapping of the short reads both on the long reads and against each other

Inspiration

- NaS [Madoui et al., 2015]
- Yields highly contiguous assembly results
- Does not locally correct erroneous regions
- Uses long reads as templates to generate corrected long reads from assemblies of short reads
- Requires the mapping of the short reads both on the long reads and against each other

Inspiration

- NaS [Madoui et al., 2015]
- Yields highly contiguous assembly results
- Does not locally correct erroneous regions
- Uses long reads as templates to generate corrected long reads from assemblies of short reads
- Requires the mapping of the short reads both on the long reads and against each other

NaS overview

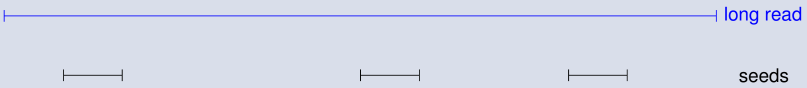
NaS corrects a long read as follows:

NaS overview

NaS corrects a long read as follows:

First step

Align the short reads to the long reads

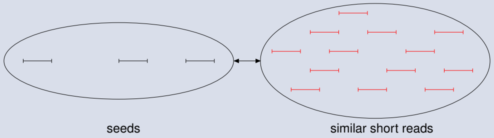


NaS overview

NaS corrects a long read as follows:

Second step

For each long read, recruit short reads that are similar to the seeds

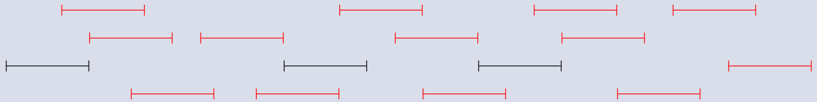


NaS overview

NaS corrects a long read as follows:

Third step

Assemble the obtained subset of short reads



NaS overview

NaS corrects a long read as follows:

Fourth step

Use the obtain contig as the correction of the initial long read

Main idea

- Use long reads as templates
- Get rid of the time consuming step of aligning the short reads against each other
- Focus on a seed and extend approach
- Rely on an enhanced de Bruijn graph, built from the short reads

Main idea

- Use long reads as templates
- Get rid of the time consuming step of aligning the short reads against each other
- Focus on a seed and extend approach
- Rely on an enhanced de Bruijn graph, built from the short reads

Main idea

- Use long reads as templates
- Get rid of the time consuming step of aligning the short reads against each other
- Focus on a seed and extend approach
- Rely on an enhanced de Bruijn graph, built from the short reads

Main idea

- Use long reads as templates
- Get rid of the time consuming step of aligning the short reads against each other
- Focus on a seed and extend approach
- Rely on an enhanced de Bruijn graph, built from the short reads

Enhanced de Bruijn graph

Problem

- de Bruijn graphs are widely used for correction and assembly...
- ...But face difficulties with locally insufficient coverage

Usual solutions

- Usually, multiple de Bruijn graphs of different orders are built
- Requires a different graph for each order
- Consumes large amounts of time and memory

Enhanced de Bruijn graph

Problem

- de Bruijn graphs are widely used for correction and assembly...
- ...But face difficulties with locally insufficient coverage

Usual solutions

- Usually, multiple de Bruijn graphs of different orders are built
- Requires a different graph for each order
- Consumes large amounts of time and memory

Enhanced de Bruijn graph

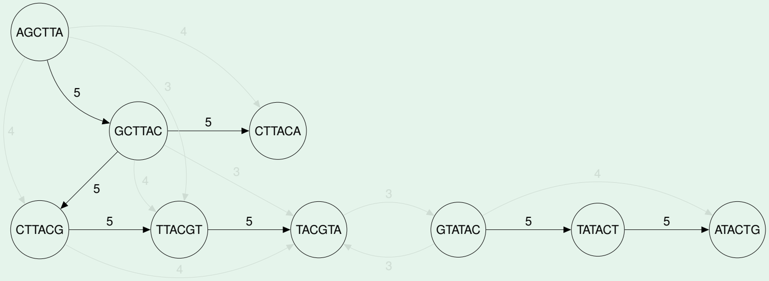
Idea

Enhance the de Bruijn graph with the capability of computing overlaps of variable lengths between the *k*-mers, in an overlap graph fashion, in order to avoid building multiple de Bruijn graphs of different orders.

Enhanced de Bruijn graph

Example

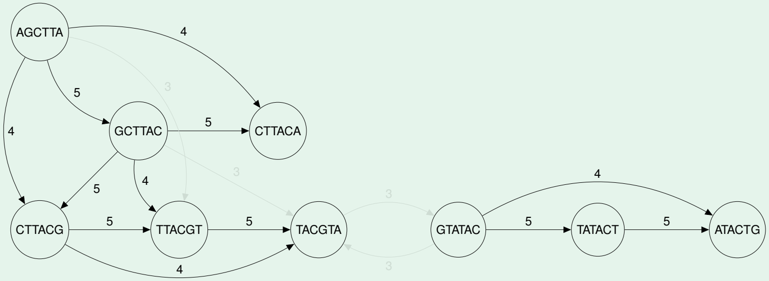
With the set of reads $S = \{\text{AGCTTACA, CTTACGTA, GTATACTG}\}$, we obtain the following enhanced de Bruijn graph of order 6:



Enhanced de Bruijn graph

Example

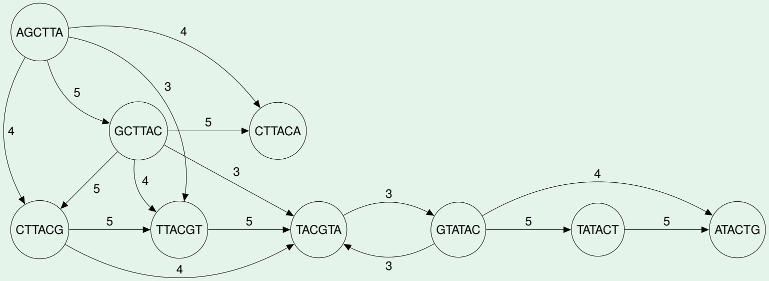
With the set of reads $S = \{$AGCTTACA, CTTACGTA, GTATACTG$\}$, we obtain the following enhanced de Bruijn graph of order 6:



Enhanced de Bruijn graph

Example

With the set of reads $S = \{$AGCTTACA, CTTACGTA, GTATACTG$\}$, we obtain the following enhanced de Bruijn graph of order 6:



Traversal

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of PgSA [Kowalski et al., 2015]:
  - The *k*-mers from the reads are stored in the index
  - The index is queried in order to retrieve the edges
- Makes backwards traversal easy

Traversal

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of PgSA [Kowalski et al., 2015]:
  - The *k*-mers from the reads are stored in the index
  - The index is queried in order to retrieve the edges
- Makes backwards traversal easy

Traversal

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of PgSA [Kowalski et al., 2015]:
  - The *k*-mers from the reads are stored in the index
  - The index is queried in order to retrieve the edges
- Makes backwards traversal easy

Traversal

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of PgSA [Kowalski et al., 2015]:
  - The *k*-mers from the reads are stored in the index
  - The index is queried in order to retrieve the edges
- Makes backwards traversal easy

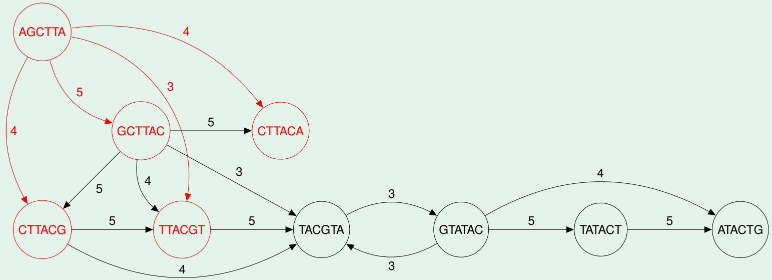
Traversal

- The enhanced de Bruijn graph does not need to be explicitly built
- It can be traversed with the help of PgSA [Kowalski et al., 2015]:
  - The *k*-mers from the reads are stored in the index
  - The index is queried in order to retrieve the edges
- Makes backwards traversal easy

Traversal

Example

Traversing the previous enhanced de Bruijn graph:



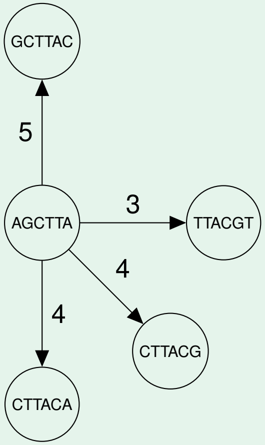
Traversal

Example

k-mers set

1: AGCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index



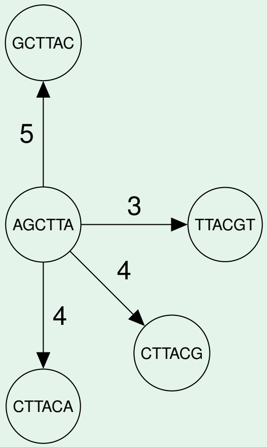
Traversal

Example

k-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index



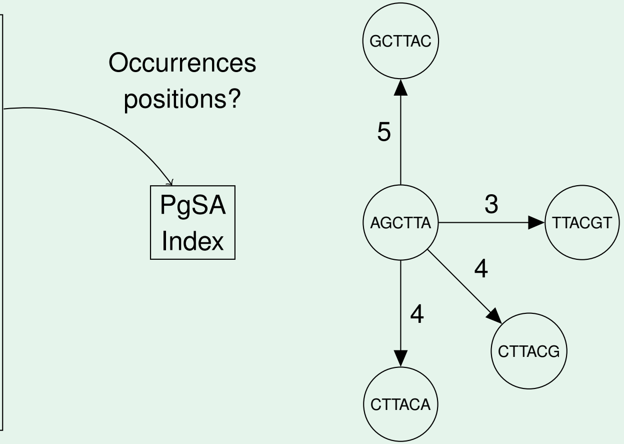
$$\begin{tikzcd}
 & \text{GCTTAC} & \\
 & \text{AGCTTA} \arrow[u, "5"] \arrow[r, "3"] \arrow[d, "4"] \arrow[dr, "4"] & \text{TTACGT} \\
 & \text{CTTACA} & \text{CTTACG}
\end{tikzcd}$$

Traversal

Example

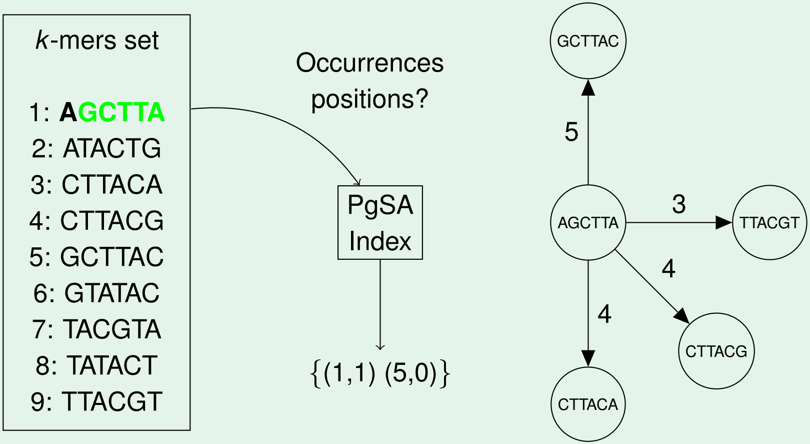
k-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Traversal

Example

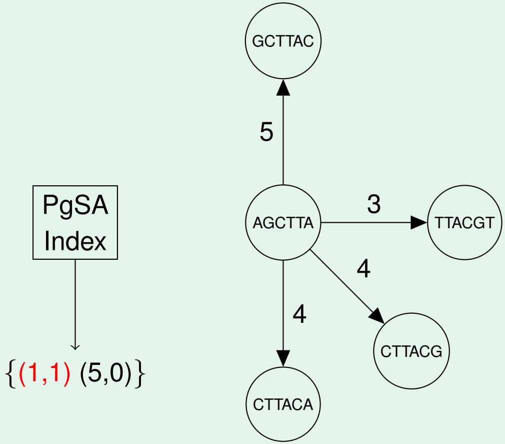


Traversal

Example

k-mers set

1: **A**GCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

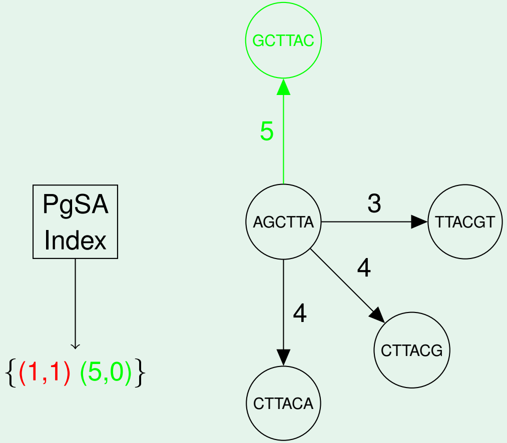


Traversal

Example

k-mers set

1: **A**GCTTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

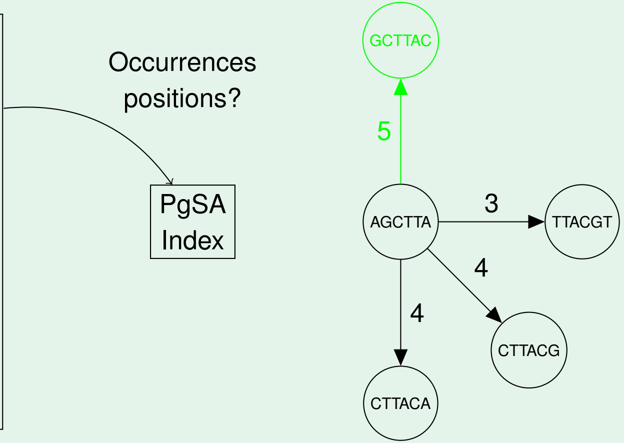


Traversal

Example

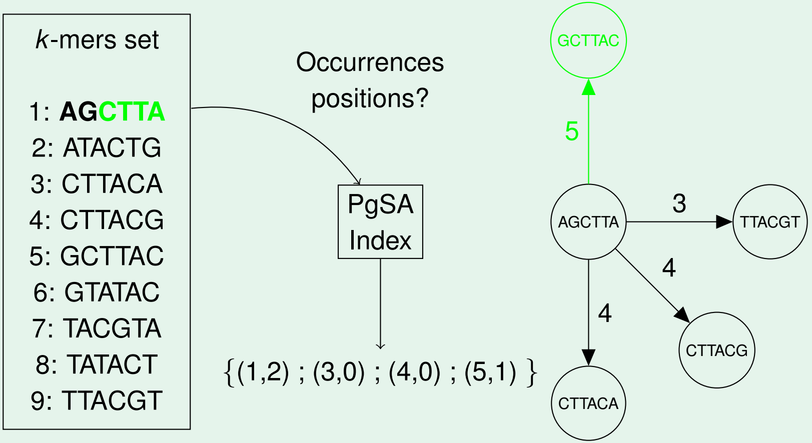
k-mers set

1: **AG**<b>CTTA</b>
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Traversal

Example

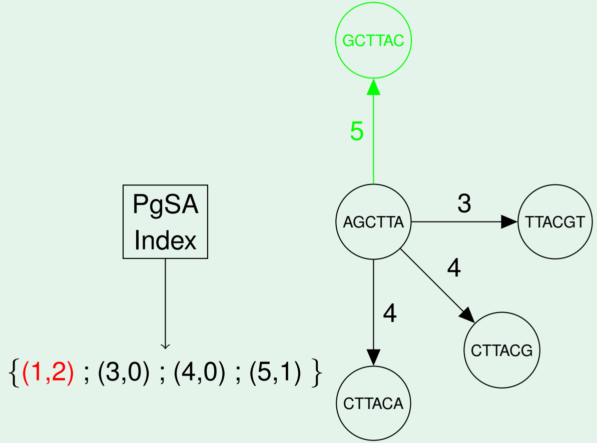


Traversal

Example

k-mers set

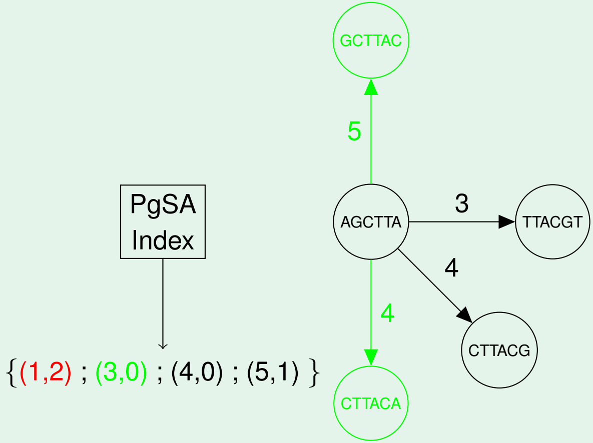
1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Traversal

Example

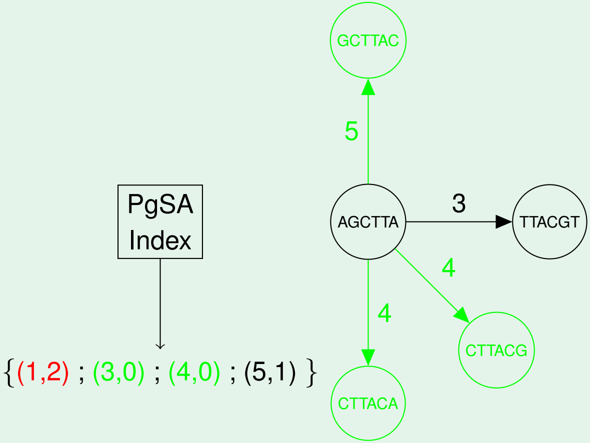
k-mers set
1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Traversal

Example

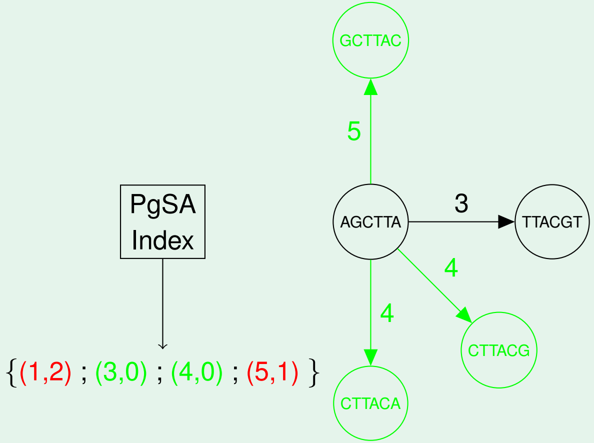
k-mers set
1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Traversal

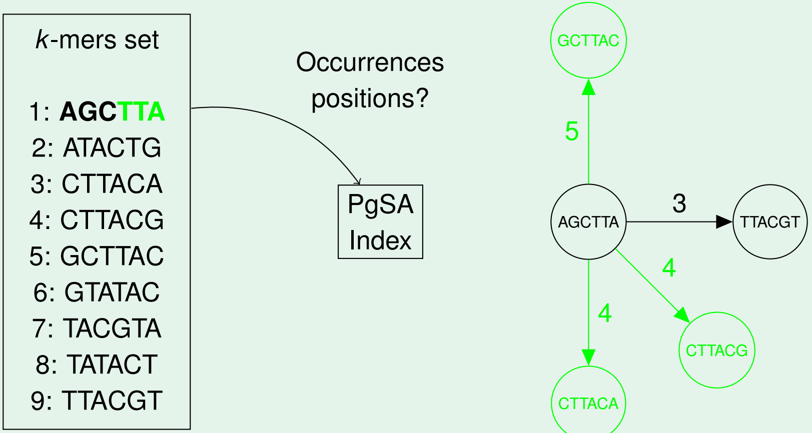
Example

k-mers set
1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



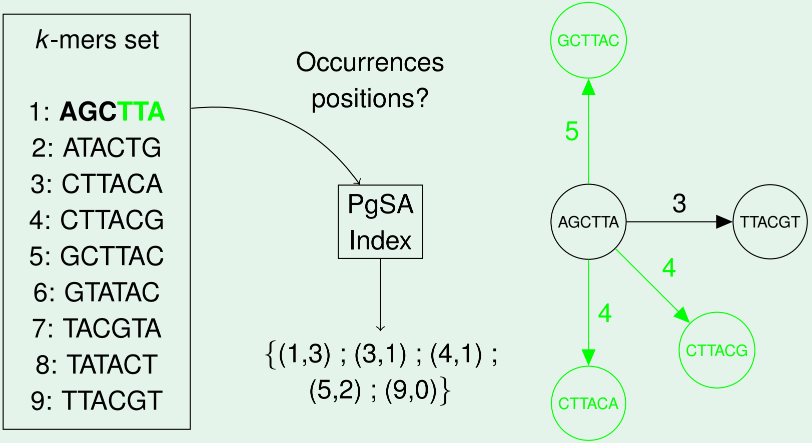
Traversal

Example



Traversal

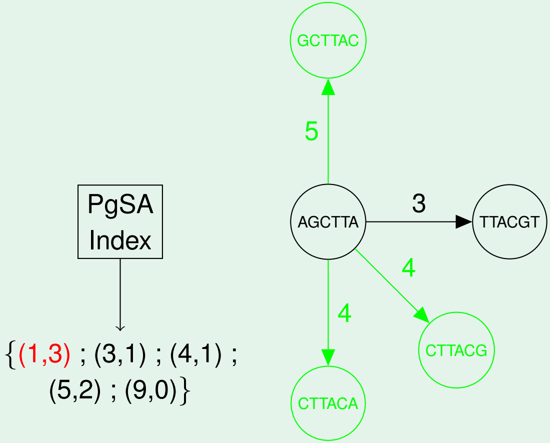
Example



Traversal

Example

k-mers set
1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

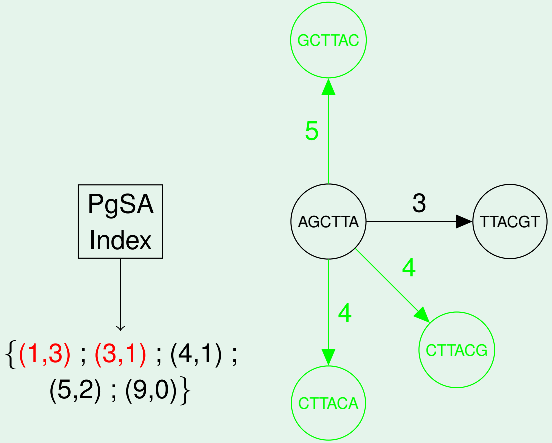


Traversal

Example

k-mers set

1: **AGC**TTA
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

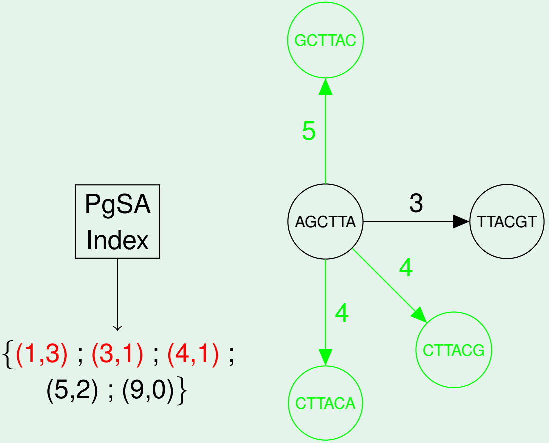


Traversal

Example

k-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Traversal

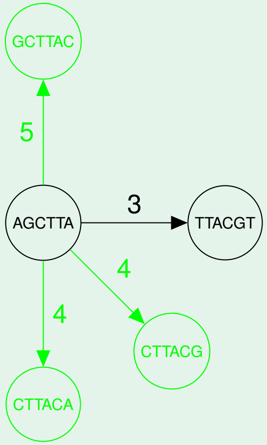
Example

k-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT

PgSA Index

{(1,3) ; (3,1) ; (4,1) ; (5,2) ; (9,0)}

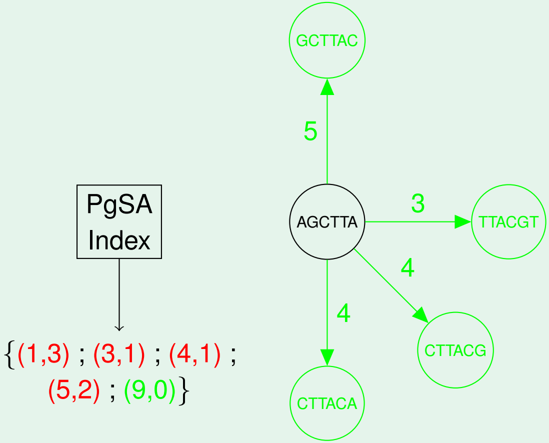


Traversal

Example

k-mers set

1: **AGCTTA**
2: ATACTG
3: CTTACA
4: CTTACG
5: GCTTAC
6: GTATAC
7: TACGTA
8: TATACT
9: TTACGT



Workflow

5 steps:

1. Correct the short reads (with QuorUM [Marçais et al., 2015])
2. Filter out corrected short reads containing weak *k*-mers, and index solid *k*-mers with PgSA
3. Align the remaining short reads to the long reads, to find seeds (with BLASR [Chaisson and Tesler, 2012])
4. Merge the overlapping seeds, and link them together, by traversing the graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

Workflow

5 steps:

1. Correct the short reads (with QuorUM [Marçais et al., 2015])
2. Filter out corrected short reads containing weak *k*-mers, and index solid *k*-mers with PgSA
3. Align the remaining short reads to the long reads, to find seeds (with BLASR [Chaisson and Tesler, 2012])
4. Merge the overlapping seeds, and link them together, by traversing the graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

Workflow

5 steps:

1. Correct the short reads (with QuorUM [Marçais et al., 2015])
2. Filter out corrected short reads containing weak *k*-mers, and index solid *k*-mers with PgSA
3. Align the remaining short reads to the long reads, to find seeds (with BLASR [Chaisson and Tesler, 2012])
4. Merge the overlapping seeds, and link them together, by traversing the graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

Workflow

5 steps:

1. Correct the short reads (with QuorUM [Marçais et al., 2015])
2. Filter out corrected short reads containing weak *k*-mers, and index solid *k*-mers with PgSA
3. Align the remaining short reads to the long reads, to find seeds (with BLASR [Chaisson and Tesler, 2012])
4. Merge the overlapping seeds, and link them together, by traversing the graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

Workflow

5 steps:

1. Correct the short reads (with QuorUM [Marçais et al., 2015])
2. Filter out corrected short reads containing weak *k*-mers, and index solid *k*-mers with PgSA
3. Align the remaining short reads to the long reads, to find seeds (with BLASR [Chaisson and Tesler, 2012])
4. Merge the overlapping seeds, and link them together, by traversing the graph
5. Extend the obtained corrected long read, on the left (resp. right) of the leftmost (resp. rightmost) seed

Step 4: Seeds merging and linking

- Seeds with overlapping mapping positions are merged
  - Perfect overlap: merge
  - Otherwise: keep the best seed
- Seeds are used as anchor points on the graph
- The graph is traversed to link the seeds together and assemble the k-mers

Step 4: Seeds merging and linking

- Seeds with overlapping mapping positions are merged
  - Perfect overlap: merge
  - Otherwise: keep the best seed
- Seeds are used as anchor points on the graph
- The graph is traversed to link the seeds together and assemble the *k*-mers

Step 4: Seeds merging and linking

- Seeds with overlapping mapping positions are merged
  - Perfect overlap: merge
  - Otherwise: keep the best seed
- Seeds are used as anchor points on the graph
- The graph is traversed to link the seeds together and assemble the *k*-mers

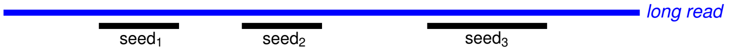
Step 4: Seeds merging and linking

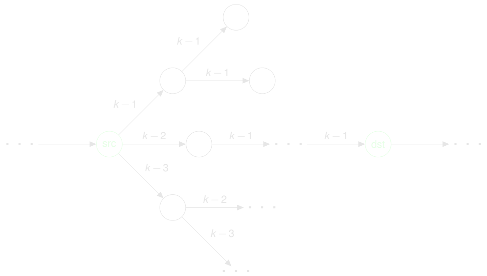
- Seeds with overlapping mapping positions are merged
  - Perfect overlap: merge
  - Otherwise: keep the best seed
- Seeds are used as anchor points on the graph
- The graph is traversed to link the seeds together and assemble the *k*-mers

Step 4: Seeds merging and linking

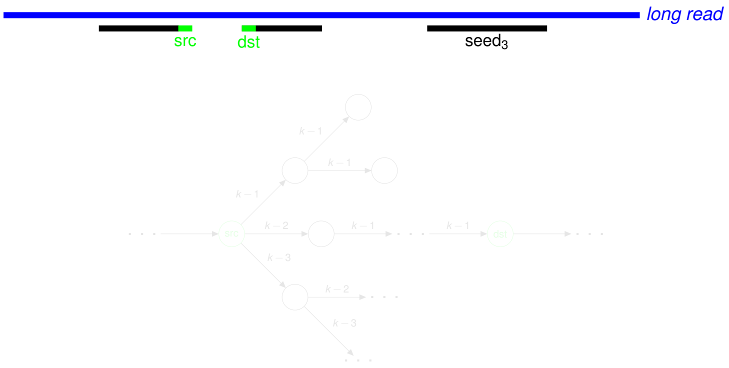
- Seeds with overlapping mapping positions are merged
  - Perfect overlap: merge
  - Otherwise: keep the best seed
- Seeds are used as anchor points on the graph
- The graph is traversed to link the seeds together and assemble the *k*-mers

Step 4: Seeds linking

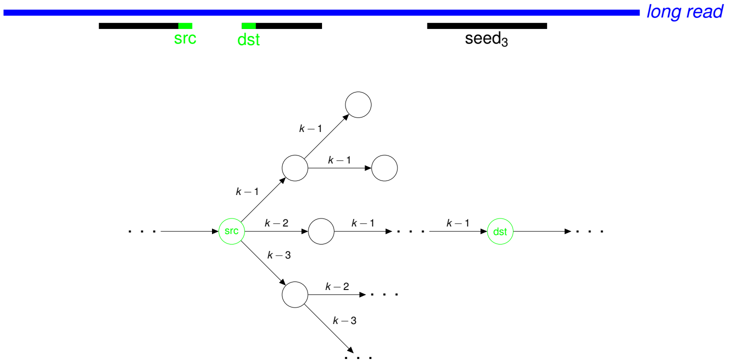




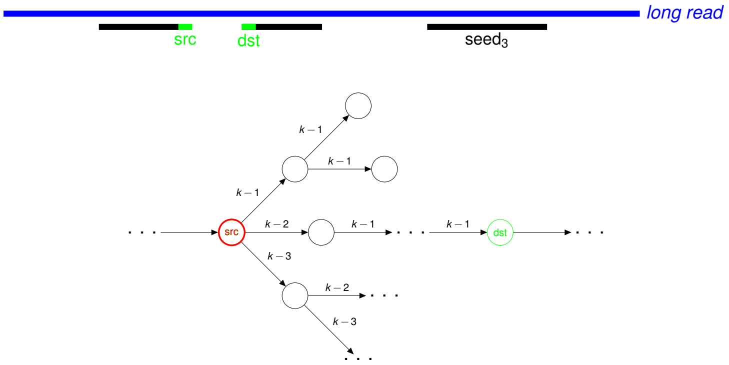
Step 4: Seeds linking



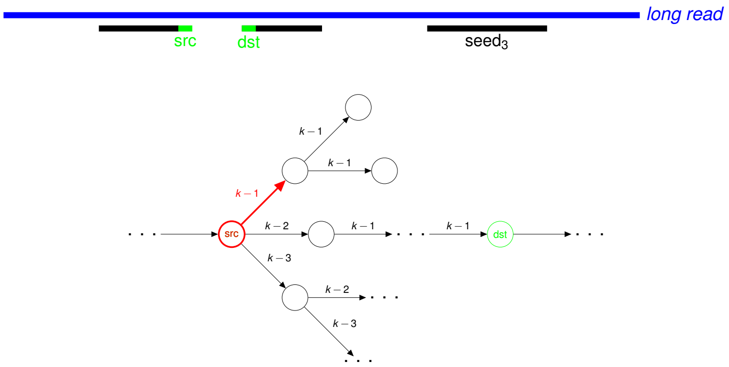
Step 4: Seeds linking



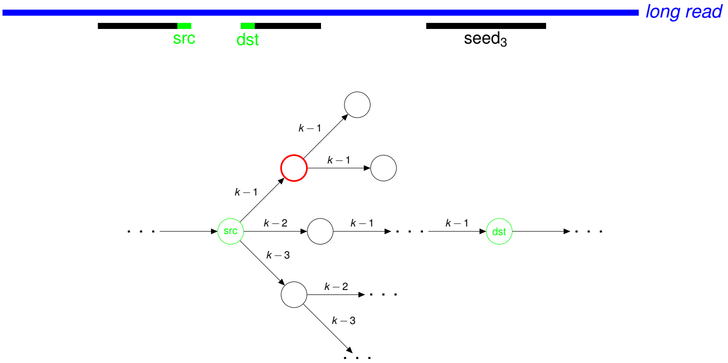
Step 4: Seeds linking



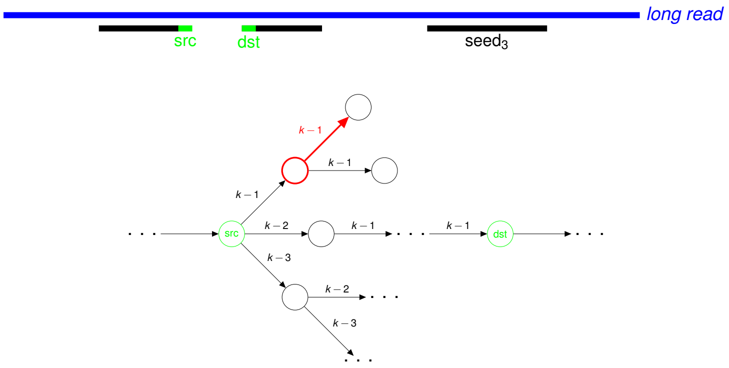
Step 4: Seeds linking



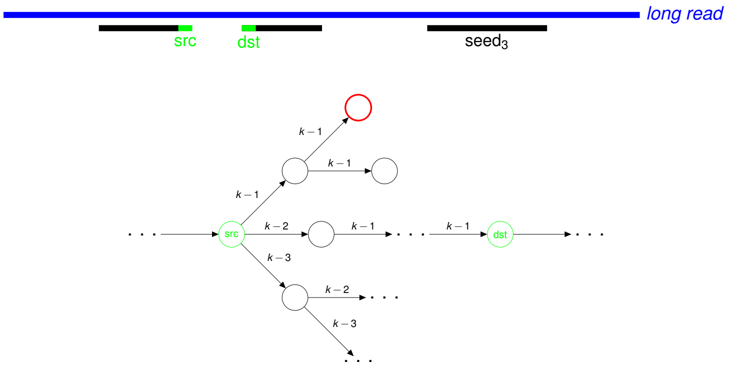
Step 4: Seeds linking



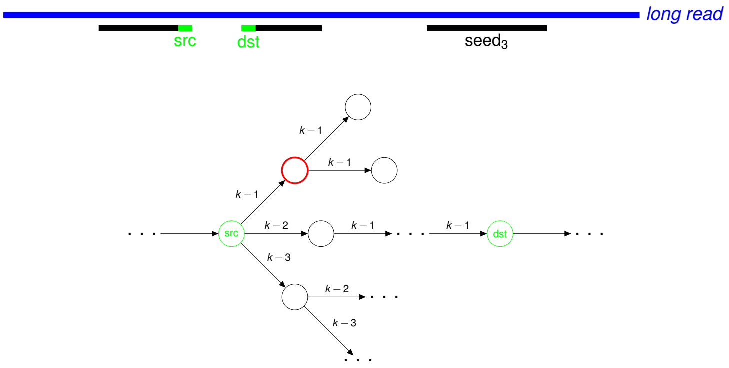
Step 4: Seeds linking



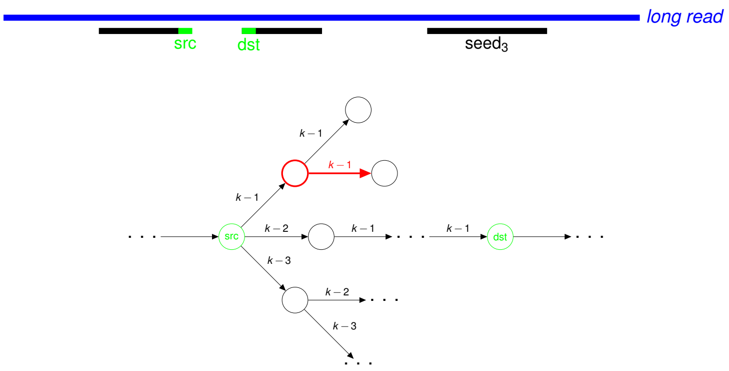
Step 4: Seeds linking



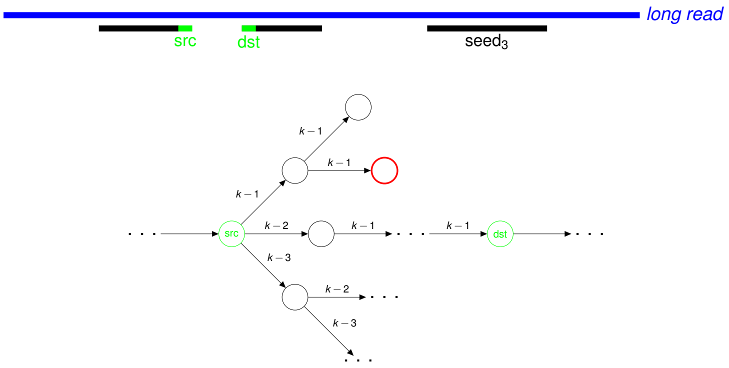
Step 4: Seeds linking



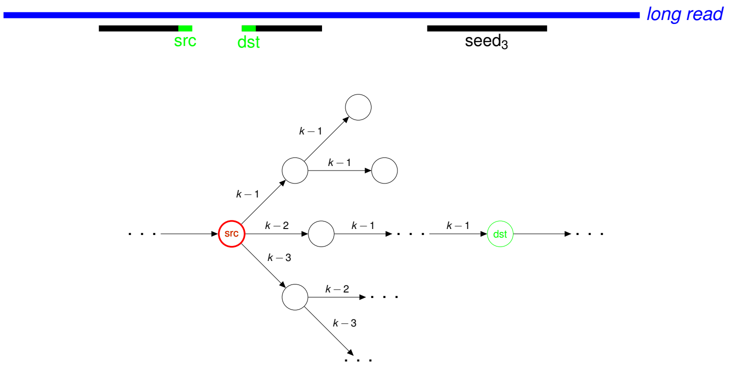
Step 4: Seeds linking



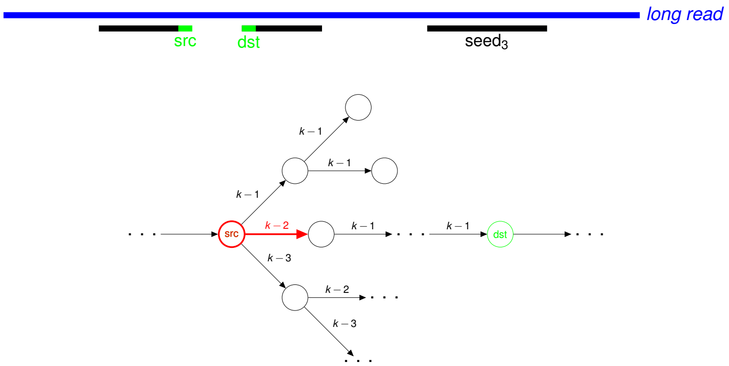
Step 4: Seeds linking



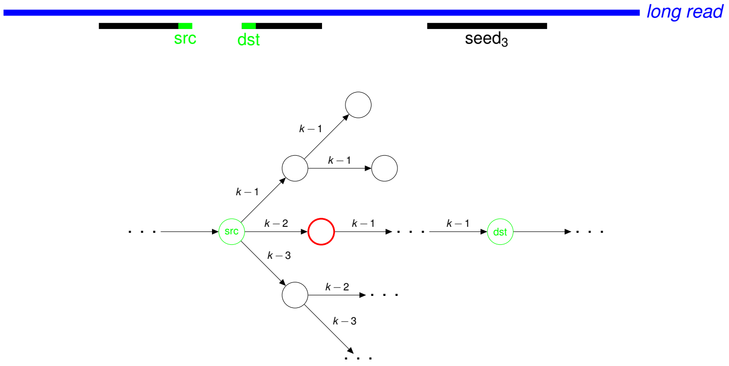
Step 4: Seeds linking



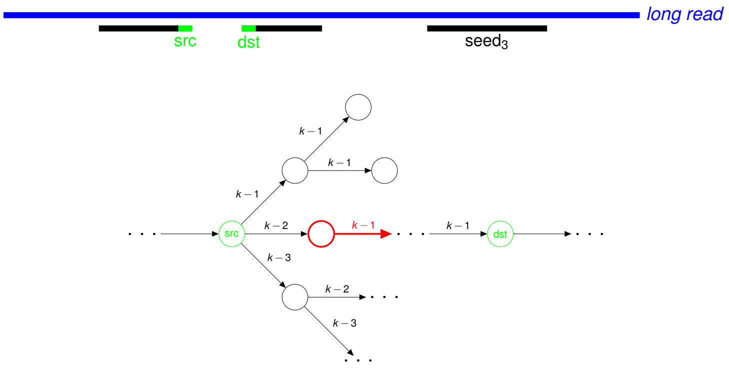
Step 4: Seeds linking



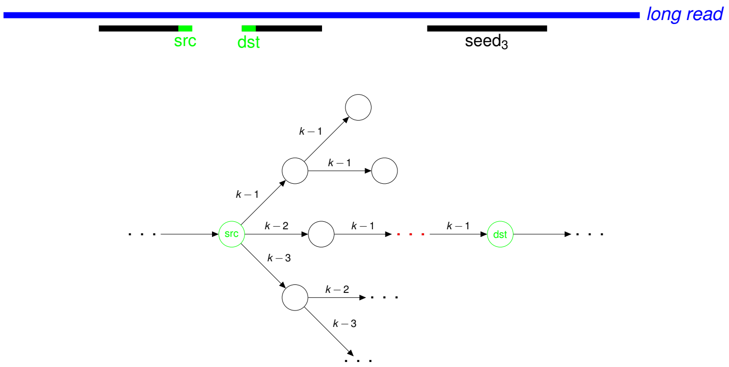
Step 4: Seeds linking



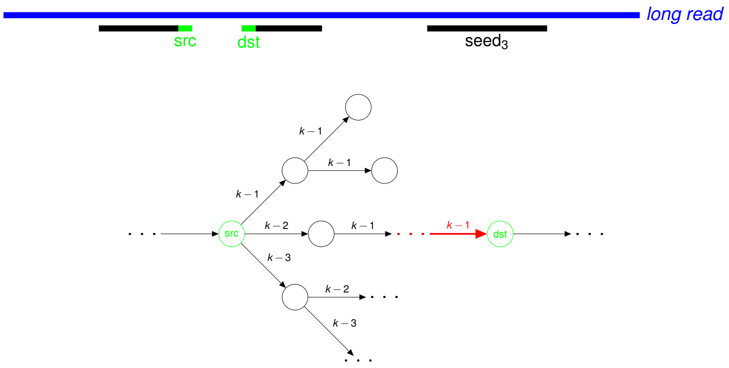
Step 4: Seeds linking



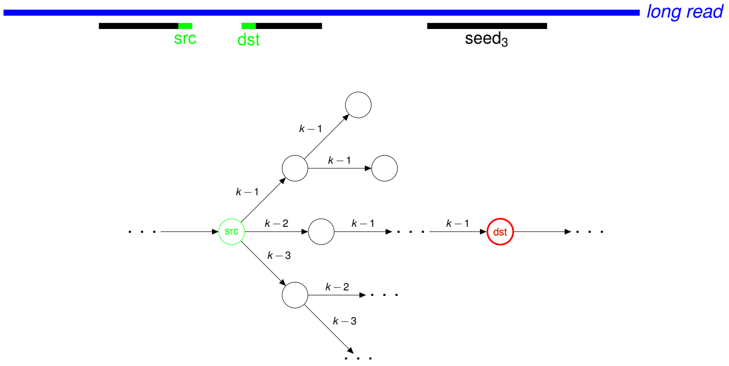
Step 4: Seeds linking



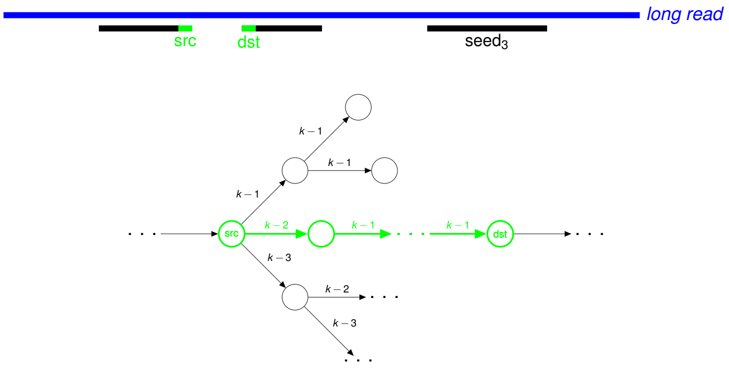
Step 4: Seeds linking



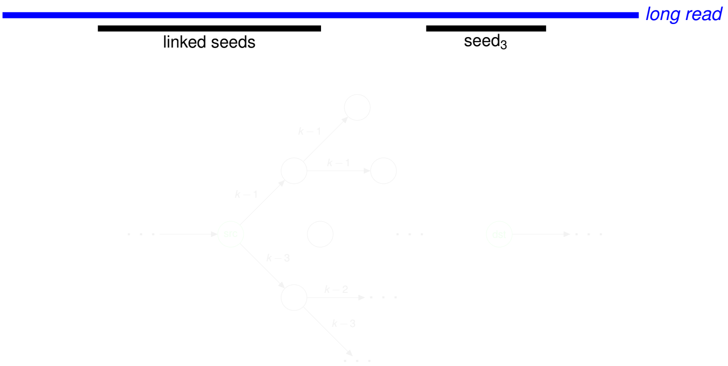
Step 4: Seeds linking



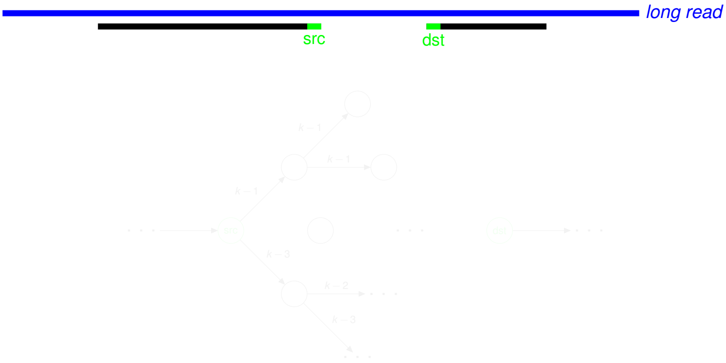
Step 4: Seeds linking



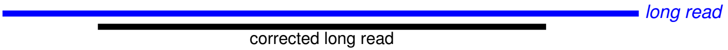
Step 4: Seeds linking



Step 4: Seeds linking



Step 4: Seeds linking



Step 5: Tips extension

- Seeds don’t always map right at the beginning or until the end of the long read
- Once all the seeds have been linked, HG-CoLoR keeps on traversing the graph
- The traversal stops when the borders of the long read or a branching path are reached

Step 5: Tips extension

- Seeds don't always map right at the beginning or until the end of the long read
- Once all the seeds have been linked, HG-CoLoR keeps on traversing the graph
- The traversal stops when the borders of the long read or a branching path are reached

Step 5: Tips extension

- Seeds don't always map right at the beginning or until the end of the long read
- Once all the seeds have been linked, HG-CoLoR keeps on traversing the graph
- The traversal stops when the borders of the long read or a branching path are reached

Remark

- Some seeds might be impossible to link together
- $\Rightarrow$ Production of a corrected long read fragmented in multiple parts

Remark

- Some seeds might be impossible to link together
- $\Rightarrow$ Production of a corrected long read fragmented in multiple parts

Datasets

HG-CoLoR was compared to NaS, and two other state-of-the-art long read hybrid correction methods: CoLoRMap [Haghshenas et al., 2016] and Jabba [Miclotte et al., 2016]

The different tools were compared on the following datasets:

Dataset	Reference genome			Oxford Nanopore data			Illumina data		
	Strain	Reference sequence	Genome size	# Reads	Average length	Coverage	# Reads	Read length	Coverage
A. baylyi	ADP1	CR543861	3.6 Mbp	89,011	4,284	106x	900,000	250	50x
E. coli	K-12 substr. MG1655	NC_000913	4.6 Mbp	22,270	5,999	29x	775,500	300	50x
S. cerevisae	S288C	NC_001133-001148	12.2 Mbp	205,923	5,698	96x	2,500,000	250	50x

Alignment-based comparison

Dataset	Method	# Reads	Average length	Average identity	Genome coverage	Runtime
A. baylyi	Original	89,011	4,284	70.09%	100%	N/A
	CoLoRMap	89,011	4,355	67.93%	100%	14h33min
	Jabba	17,476	10,260	99.40%	99.80%	12min30
	NaS	28,492	9,530	99.83%	100%	128h55min
	HG-CoLoR	25,436	11,619	99.70%	100%	1h59min
E. coli	Original	22,270	5,999	79.46%	100%	N/A
	CoLoRMap	22,270	6,219	89.02%	100%	8h26min
	Jabba	22,065	5,794	99.81%	99.41%	12min56
	NaS	22,144	8,307	99.86%	100%	81h30min
	HG-CoLoR	21,969	6,125	99.80%	100%	1h17min
S. cerevisae	Original	205,923	5,698	55.49%	99.90%	N/A
	CoLoRMap	205,923	5,737	39.93%	99.40%	37h36min
	Jabba	36,958	6,613	99.55%	93.21%	44min05
	NaS	85,432	6,770	99.16%	99.37%	$>$ 16 days
	HG-CoLoR	75,036	6,991	98.81%	99.47%	11h45min

Alignment-based comparison

Dataset	Method	# Reads	Average length	Average identity	Genome coverage	Runtime
A. baylyi	**Original**	**89,011**	**4,284**	**70.09%**	**100%**	**N/A**
	CoLoRMap	89,011	4,355	67.93%	100%	14h33min
	Jabba	17,476	10,260	99.40%	99.80%	12min30
	NaS	28,492	9,530	99.83%	100%	128h55min
	HG-CoLoR	25,436	11,619	99.70%	100%	1h59min
E. coli	**Original**	**22,270**	**5,999**	**79.46%**	**100%**	**N/A**
	CoLoRMap	22,270	6,219	89.02%	100%	8h26min
	Jabba	22,065	5,794	99.81%	99.41%	12min56
	NaS	22,144	8,307	99.86%	100%	81h30min
	HG-CoLoR	21,969	6,125	99.80%	100%	1h17min
S. cerevisae	**Original**	**205,923**	**5,698**	**55.49%**	**99.90%**	**N/A**
	CoLoRMap	205,923	5,737	39.93%	99.40%	37h36min
	Jabba	36,958	6,613	99.55%	93.21%	44min05
	NaS	85,432	6,770	99.16%	99.37%	$>$ 16 days
	HG-CoLoR	75,036	6,991	98.81%	99.47%	11h45min

Alignment-based comparison

Dataset	Method	# Reads	Average length	Average identity	Genome coverage	Runtime
A. baylyi	Original	89,011	4,284	70.09%	100%	N/A
	CoLoRMap	**89,011**	**4,355**	**67.93%**	**100%**	**14h33min**
	Jabba	17,476	10,260	99.40%	99.80%	12min30
	NaS	28,492	9,530	99.83%	100%	128h55min
	HG-CoLoR	25,436	11,619	99.70%	100%	1h59min
E. coli	Original	22,270	5,999	79.46%	100%	N/A
	CoLoRMap	**22,270**	**6,219**	**89.02%**	**100%**	**8h26min**
	Jabba	22,065	5,794	99.81%	99.41%	12min56
	NaS	22,144	8,307	99.86%	100%	81h30min
	HG-CoLoR	21,969	6,125	99.80%	100%	1h17min
S. cerevisae	Original	205,923	5,698	55.49%	99.90%	N/A
	CoLoRMap	**205,923**	**5,737**	**39.93%**	**99.40%**	**37h36min**
	Jabba	36,958	6,613	99.55%	93.21%	44min05
	NaS	85,432	6,770	99.16%	99.37%	$>$ 16 days
	HG-CoLoR	75,036	6,991	98.81%	99.47%	11h45min

Alignment-based comparison

Dataset	Method	# Reads	Average length	Average identity	Genome coverage	Runtime
A. baylyi	Original	89,011	4,284	70.09%	100%	N/A
	CoLoRMap	89,011	4,355	67.93%	100%	14h33min
	Jabba	**17,476**	**10,260**	**99.40%**	**99.80%**	**12min30**
	NaS	28,492	9,530	99.83%	100%	128h55min
	HG-CoLoR	25,436	11,619	99.70%	100%	1h59min
E. coli	Original	22,270	5,999	79.46%	100%	N/A
	CoLoRMap	22,270	6,219	89.02%	100%	8h26min
	Jabba	**22,065**	**5,794**	**99.81%**	**99.41%**	**12min56**
	NaS	22,144	8,307	99.86%	100%	81h30min
	HG-CoLoR	21,969	6,125	99.80%	100%	1h17min
S. cerevisae	Original	205,923	5,698	55.49%	99.90%	N/A
	CoLoRMap	205,923	5,737	39.93%	99.40%	37h36min
	Jabba	**36,958**	**6,613**	**99.55%**	**93.21%**	**44min05**
	NaS	85,432	6,770	99.16%	99.37%	$>$ 16 days
	HG-CoLoR	75,036	6,991	98.81%	99.47%	11h45min

Alignment-based comparison

Dataset	Method	# Reads	Average length	Average identity	Genome coverage	Runtime
A. baylyi	Original	89,011	4,284	70.09%	100%	N/A
	CoLoRMap	89,011	4,355	67.93%	100%	14h33min
	Jabba	17,476	10,260	99.40%	99.80%	12min30
	NaS	**28,492**	**9,530**	**99.83%**	**100%**	**128h55min**
	HG-CoLoR	25,436	11,619	99.70%	100%	1h59min
E. coli	Original	22,270	5,999	79.46%	100%	N/A
	CoLoRMap	22,270	6,219	89.02%	100%	8h26min
	Jabba	22,065	5,794	99.81%	99.41%	12min56
	NaS	**22,144**	**8,307**	**99.86%**	**100%**	**81h30min**
	HG-CoLoR	21,969	6,125	99.80%	100%	1h17min
S. cerevisae	Original	205,923	5,698	55.49%	99.90%	N/A
	CoLoRMap	205,923	5,737	39.93%	99.40%	37h36min
	Jabba	36,958	6,613	99.55%	93.21%	44min05
	NaS	**85,432**	**6,770**	**99.16%**	**99.37%**	**$>$ 16 days**
	HG-CoLoR	75,036	6,991	98.81%	99.47%	11h45min

Alignment-based comparison

Dataset	Method	# Reads	Average length	Average identity	Genome coverage	Runtime
A. baylyi	Original	89,011	4,284	70.09%	100%	N/A
	CoLoRMap	89,011	4,355	67.93%	100%	14h33min
	Jabba	17,476	10,260	99.40%	99.80%	12min30
	NaS	28,492	9,530	99.83%	100%	128h55min
	HG-CoLoR	**25,436**	**11,619**	**99.70%**	**100%**	**1h59min**
E. coli	Original	22,270	5,999	79.46%	100%	N/A
	CoLoRMap	22,270	6,219	89.02%	100%	8h26min
	Jabba	22,065	5,794	99.81%	99.41%	12min56
	NaS	22,144	8,307	99.86%	100%	81h30min
	HG-CoLoR	**21,969**	**6,125**	**99.80%**	**100%**	**1h17min**
S. cerevisae	Original	205,923	5,698	55.49%	99.90%	N/A
	CoLoRMap	205,923	5,737	39.93%	99.40%	37h36min
	Jabba	36,958	6,613	99.55%	93.21%	44min05
	NaS	85,432	6,770	99.16%	99.37%	$>$ 16 days
	HG-CoLoR	**75,036**	**6,991**	**98.81%**	**99.47%**	**11h45min**

Assembly-based comparison

Dataset	Method	# Reads	Coverage	# Expected contigs	# Obtained contigs	Genome coverage
A. baylyi	CoLoRMap	89,011	108x	1	_	_
	Jabba	17,476	50x	1	13	89.43%
	NaS	28,492	75x	1	1	100%
	HG-CoLoR	25,436	82x	1	1	99.99%
E. coli	CoLoRMap	22,270	30x	1	29	97,74%
	Jabba	22,065	28x	1	41	95.76%
	NaS	22,144	40x	1	1	100%
	HG-CoLoR	21,969	29x	1	1	100%
S. cerevisae	CoLoRMap	205,923	98x	16	_	_
	Jabba	36,958	20x	16	134	70.52%
	NaS	85,432	48x	16	122	96.72%
	HG-CoLoR	75,036	43x	16	81	96.11%

Assembly-based comparison

Dataset	Method	# Reads	Coverage	# Expected contigs	# Obtained contigs	Genome coverage
A. baylyi	**CoLoRMap**	**89,011**	**108x**	**1**	–	–
	Jabba	17,476	50x	1	13	89.43%
	NaS	28,492	75x	1	1	100%
	HG-CoLoR	25,436	82x	1	1	99.99%
E. coli	**CoLoRMap**	**22,270**	**30x**	**1**	**29**	**97,74%**
	Jabba	22,065	28x	1	41	95.76%
	NaS	22,144	40x	1	1	100%
	HG-CoLoR	21,969	29x	1	1	100%
S. cerevisae	**CoLoRMap**	**205,923**	**98x**	**16**	–	–
	Jabba	36,958	20x	16	134	70.52%
	NaS	85,432	48x	16	122	96.72%
	HG-CoLoR	75,036	43x	16	81	96.11%

Assembly-based comparison

Dataset	Method	# Reads	Coverage	# Expected contigs	# Obtained contigs	Genome coverage
A. baylyi	CoLoRMap	89,011	108x	1	_	_
	Jabba	**17,476**	**50x**	**1**	**13**	**89.43%**
	NaS	28,492	75x	1	1	100%
	HG-CoLoR	25,436	82x	1	1	99.99%
E. coli	CoLoRMap	22,270	30x	1	29	97,74%
	Jabba	**22,065**	**28x**	**1**	**41**	**95.76%**
	NaS	22,144	40x	1	1	100%
	HG-CoLoR	21,969	29x	1	1	100%
S. cerevisae	CoLoRMap	205,923	98x	16	_	_
	Jabba	**36,958**	**20x**	**16**	**134**	**70.52%**
	NaS	85,432	48x	16	122	96.72%
	HG-CoLoR	75,036	43x	16	81	96.11%

Assembly-based comparison

Dataset	Method	# Reads	Coverage	# Expected contigs	# Obtained contigs	Genome coverage
A. baylyi	CoLoRMap	89,011	108x	1	_	_
	Jabba	17,476	50x	1	13	89.43%
	NaS	**28,492**	**75x**	**1**	**1**	**100%**
	HG-CoLoR	25,436	82x	1	1	99.99%
E. coli	CoLoRMap	22,270	30x	1	29	97,74%
	Jabba	22,065	28x	1	41	95.76%
	NaS	**22,144**	**40x**	**1**	**1**	**100%**
	HG-CoLoR	21,969	29x	1	1	100%
S. cerevisae	CoLoRMap	205,923	98x	16	_	_
	Jabba	36,958	20x	16	134	70.52%
	NaS	**85,432**	**48x**	**16**	**122**	**96.72%**
	HG-CoLoR	75,036	43x	16	81	96.11%

Assembly-based comparison

Dataset	Method	# Reads	Coverage	# Expected contigs	# Obtained contigs	Genome coverage
A. baylyi	CoLoRMap	89,011	108x	1	_	_
	Jabba	17,476	50x	1	13	89.43%
	NaS	28,492	75x	1	1	100%
	HG-CoLoR	**25,436**	**82x**	**1**	**1**	**99.99%**
E. coli	CoLoRMap	22,270	30x	1	29	97,74%
	Jabba	22,065	28x	1	41	95.76%
	NaS	22,144	40x	1	1	100%
	HG-CoLoR	**21,969**	**29x**	**1**	**1**	**100%**
S. cerevisae	CoLoRMap	205,923	98x	16	_	_
	Jabba	36,958	20x	16	134	70.52%
	NaS	85,432	48x	16	122	96.72%
	HG-CoLoR	**75,036**	**43x**	**16**	**81**	**96.11%**

Conclusion

- Uses long reads as templates instead of locally correcting them
- Exploits the advantages of the enhanced de Bruijn Graph
- Oriented towards assembly
- Several orders of magnitude faster than NaS, while achieving comparable results
- Provides the best trade off between runtime and quality, when compared to state-of-the-art methods
- HG-CoLoR is available from: `https://github.com/pierre-morisse/HG-CoLoR`

Conclusion

- Uses long reads as templates instead of locally correcting them
- Exploits the advantages of the enhanced de Bruijn Graph
- Oriented towards assembly
- Several orders of magnitude faster than NaS, while achieving comparable resutls
- Provides the best trade off between runtime and quality, when compared to state-of-the-art methods
- HG-CoLoR is available from: `https://github.com/pierre-morisse/HG-CoLoR`

Conclusion

- Uses long reads as templates instead of locally correcting them
- Exploits the advantages of the enhanced de Bruijn Graph
- Oriented towards assembly
- Several orders of magnitude faster than NaS, while achieving comparable resutls
- Provides the best trade off between runtime and quality, when compared to state-of-the-art methods
- HG-CoLoR is available from: `https://github.com/pierre-morisse/HG-CoLoR`

Conclusion

- Uses long reads as templates instead of locally correcting them
- Exploits the advantages of the enhanced de Bruijn Graph
- Oriented towards assembly
- Several orders of magnitude faster than NaS, while achieving comparable resutls
- Provides the best trade off between runtime and quality, when compared to state-of-the-art methods
- HG-CoLoR is available from:
  `https://github.com/pierre-morisse/HG-CoLoR`

Conclusion

- Uses long reads as templates instead of locally correcting them
- Exploits the advantages of the enhanced de Bruijn Graph
- Oriented towards assembly
- Several orders of magnitude faster than NaS, while achieving comparable resutls
- Provides the best trade off between runtime and quality, when compared to state-of-the-art methods
- HG-CoLoR is available from: `https://github.com/pierre-morisse/HG-CoLoR`

Conclusion

- Uses long reads as templates instead of locally correcting them
- Exploits the advantages of the enhanced de Bruijn Graph
- Oriented towards assembly
- Several orders of magnitude faster than NaS, while achieving comparable resutls
- Provides the best trade off between runtime and quality, when compared to state-of-the-art methods
- HG-CoLoR is available from:
  `https://github.com/pierre-morisse/HG-CoLoR`

Future work

- Run HG-CoLoR on larger genomes
- Build a proper assembly tool from the enhanced de Bruijn graph
- Adapt HG-CoLoR to self-correction

Future work

- Run HG-CoLoR on larger genomes
- Build a proper assembly tool from the enhanced de Bruijn graph
- Adapt HG-CoLoR to self-correction

Future work

- Run HG-CoLoR on larger genomes
- Build a proper assembly tool from the enhanced de Bruijn graph
- Adapt HG-CoLoR to self-correction

References I

Chaisson, M. J. and Tesler, G. (2012).
Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory.
BMC bioinformatics, 13(1):238.

Haghshenas, E., Hach, F., Sahinalp, S. C., and Chauve, C. (2016).
CoLoRMap: Correcting Long Reads by Mapping short reads.
Bioinformatics, 32(17):i545–i551.

Kowalski, T., Grabowski, S., and Deorowicz, S. (2015).
Indexing arbitrary-length k-mers in sequencing reads.
PLoS ONE, 10(7):1–14.

References II

Madoui, M.-A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., and Aury, J.-M. (2015).
Genome assembly using Nanopore-guided long and error-free DNA reads.
BMC Genomics, 16:327.

Marçais, G., Yorke, J. A., and Zimin, A. (2015).
QuorUM: An Error Corrector for Illumina Reads.
PLOS ONE, 10(6):1–13.

Miclotte, G., Heydari, M., Demeester, P., Rombauts, S., Van de Peer, Y., Audenaert, P., and Fostier, J. (2016).
Jabba: hybrid error correction for long sequencing reads.
Algorithms Mol Biol, 11:10.

Thanks for your attention.