Correction of Sequencing Errors

Summer school on bioinformatics data structures

Leena Salmela

University of Helsinki

August 9th, 2016

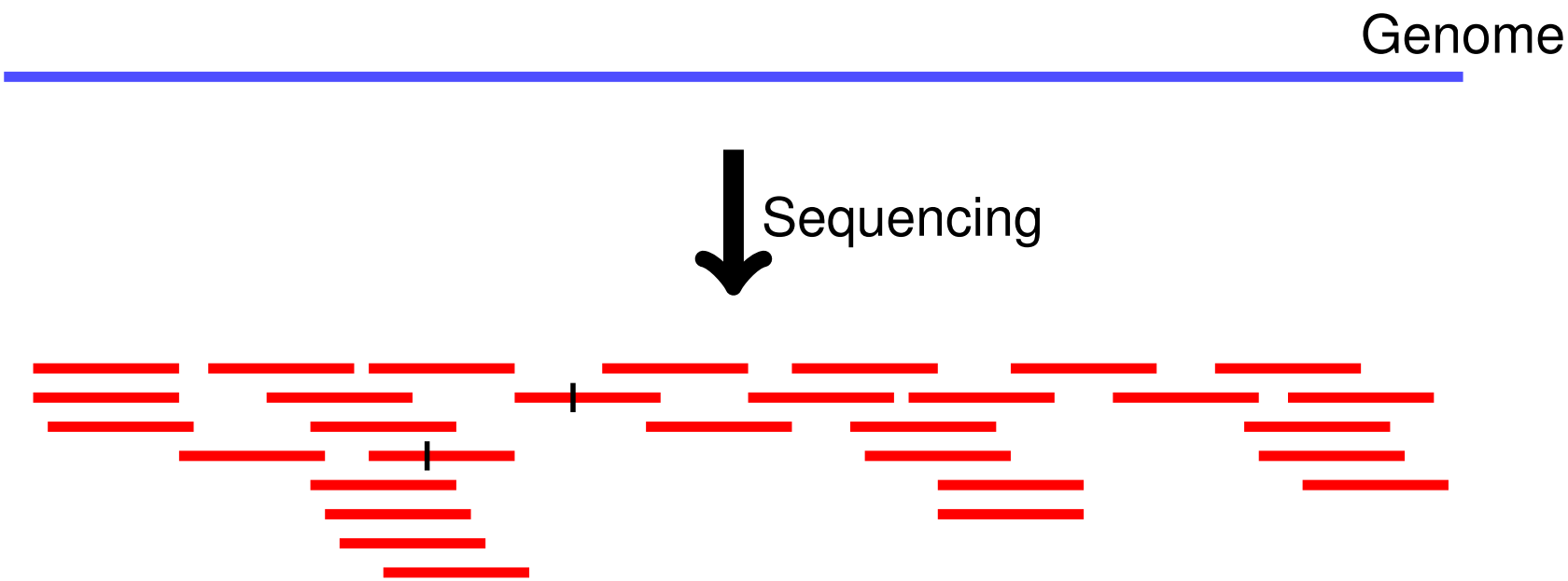
This lecture was part of the 1st Summer School on Bioinformatics Data Structures, funded by BIRDS project (www.birdsproject.eu)
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 690941

Outline

Outline

Sequencing technologies

Technology	Read length	Error rate	Typical errors
Illumina	150 - 300	$< 1\%$	substitutions
Pacific Biosciences	20000	$> 15\%$	indels
Oxford NanoPore MinION	up to 100000	10-30%	indels



Error correction problem

- High throughput sequencing produces large sets of short DNA sequences i.e. reads that may contain errors
- Sequencing errors greatly complicate *de novo* assembly
- Error correction aims at reducing the error rate prior assembly
- Input of error correction:
  - k reads i.e. strings usually containing characters ACGTN
  - Each read may be from the forward or the reverse strand
  - The length of reads can vary from read to read
  - All (or at least most) reads come from the target genome but each read may contain a small number of errors

How does error correction work?

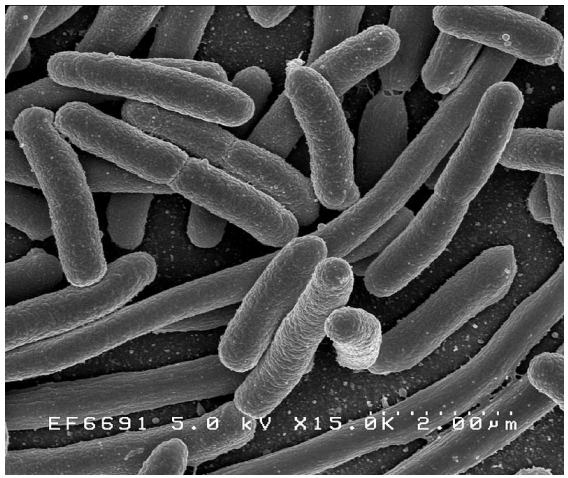
- DNA sequencing randomly samples reads from the genome
- The reads may contain errors
- Each position is sampled several times
- Errors can be detected by aligning the reads with each other
- SNPs or errors?

```
   A C G G T A G A T G C T A G         G G T A G T A G T ...
           T A G A T G C T A G C T A G G G
... A C G G T A         C T A A C T A G G G T A G T
     C G G T A G A T G C T A G C T A G       A G T A G T ...
           T A G A T G C T A G C T A G G G T A
```

Outline

Illumina: An example data set (E. coli K12)

- Read length: 100
- Number of reads: 2.3 million
- Coverage: 50x




```
 Score =  180 bits (97),  Expect = 2e-46
 Identities = 99/100 (99%), Gaps = 0/100 (0%)
 Strand=Plus/Plus

Query  1        TTATTGTACAGCGCCCAGACAATTAACACGACGGCATTCGCCACTGCCAGCAAAAAATAG  60
                ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1085516  TTATTGTACAGCGCCCAGACAATTAACACGACGGCATTCGCCACTGCCAGCAAAAAATAG  1085575

Query  61       AACTGAAGTCCGCTTCTGGCCTCGCTTTGCCAGTAATAAC  100
                |||||||||| ||||||||||||||||||||||||||||
Sbjct  1085576  AACTGAAGTCGGCTTCTGGCCTCGCTTTGCCAGTAATAAC  1085615
```

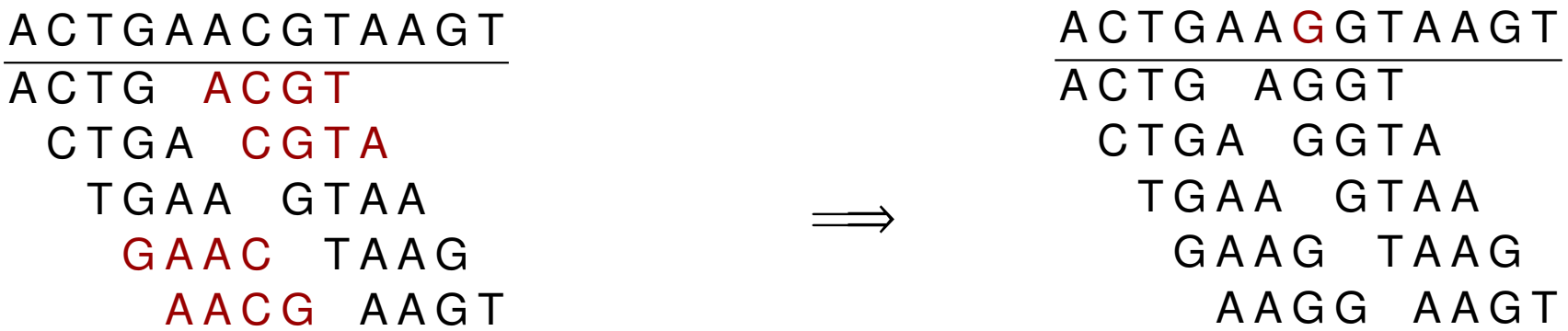
k-mer spectrum based methods

- Consider the set of k-mers occuring in the reads
  - Solid k-mer: occurs at least M times
  - Otherwise a k-mer is weak
- Find the minimum number of edits that make all k-mers solid in a read
  - Dynamic programming

```
ACTGAACGTAAGT
ACTG  ACGT
 CTGA  CGTA
  TGAA  GTAA
   GAAC  TAAG
    AACG  AAGT
```

k-mer spectrum based methods

- Consider the set of k-mers occuring in the reads
  - Solid k-mer: occurs at least M times
  - Otherwise a k-mer is weak
- Find the minimum number of edits that make all k-mers solid in a read
  - Dynamic programming



Alignment based methods

Two subproblems:

- Identify clusters of reads originating from same genomic position
- Form a multiple alignment of a cluster of reads

Identifying clusters of reads

- Typically a k-mer index is used
- Construct a hash table that associates each k-mer to the reads where it occurs either in forward or reverse orientation
- Index only those k-mers that are lexicographically smaller than their reverse complements

GTCAGAAGTCGTGGTAACCCTTGATA
GGTATCAAGGGTTACCACGACTTTCTG
CAGAAGTCGTGGTAACCCTTGATACC
GTCGTGGTAACCTTGATACCACCGG
GAACCGGTGGTATCAAGGGTTACCA
GGTACCCCTTGATACCACCGGTTCA

$\Rightarrow$

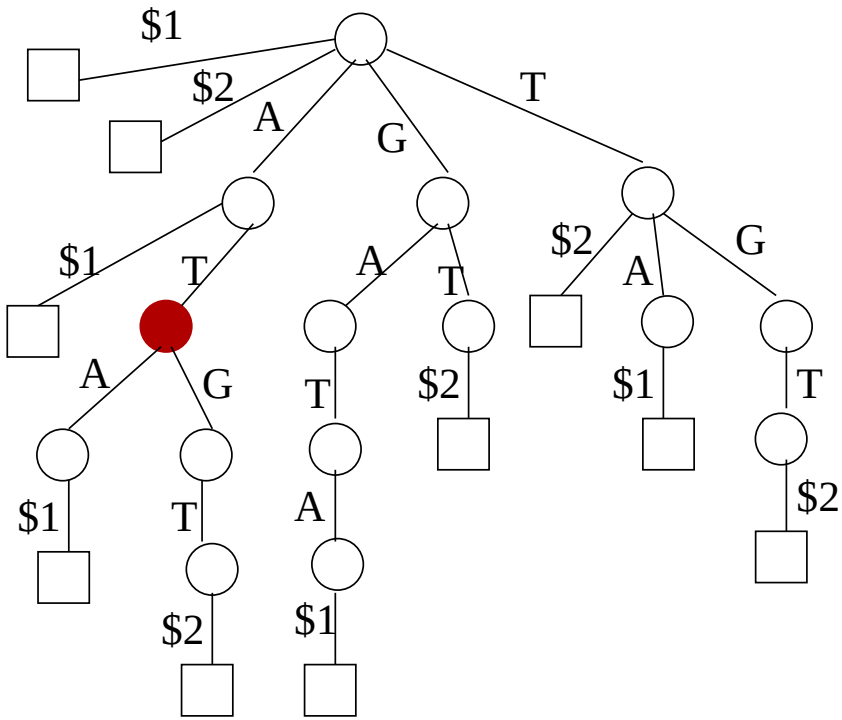
⋮
CAGAAGTC: 1f 3f
⋮
CTTCTGAC: 1r
⋮
GGTTACCA: 1r 2f 3r 4r 5f
⋮
TCAGAAGT: 1f
⋮

Forming multiple alignments

- Each multiple alignment is based on one read called the *base read*
- Retrieve from the index all reads that share at least one k-mer with the base read (*k-mer neighborhood* of the base read)
- Heuristics needed to speed up construction of the multiple alignment

Suffix trie/array based methods

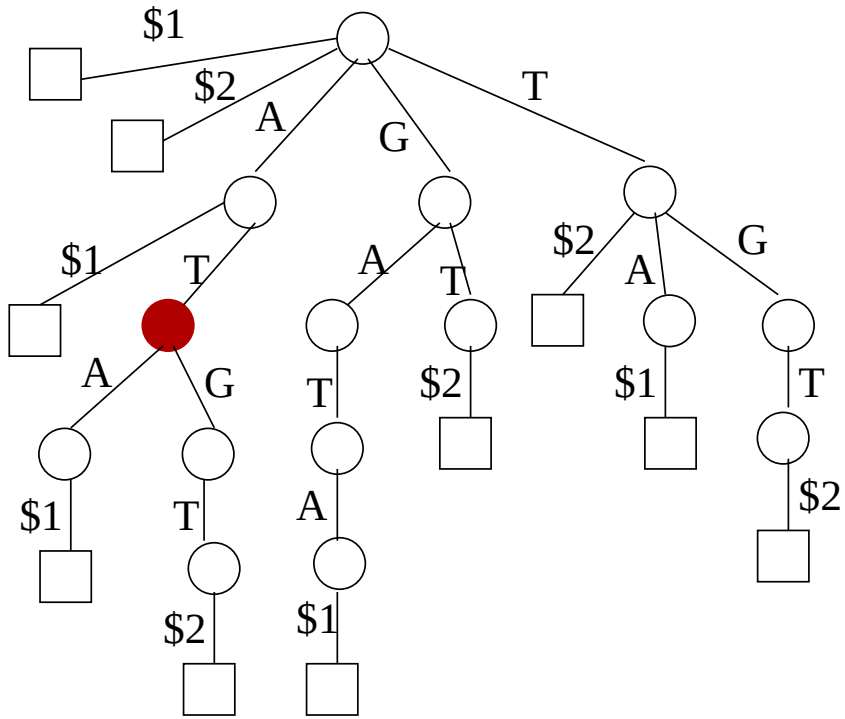
- Computing alignments is time consuming
- *k*-mer spectrum methods do not take full advantage of context
- How about using a suffix trie for presenting alignments implicitly?



Generalized suffix trie

- Generalized suffix trie of a set of reads is a tree containing all the suffixes of the reads
- Concatenate a unique symbol $\$_i$ to each read so that all the suffixes are unique
- Path label of a node is the concatenation of edge labels on the path from the root to the node
- Level of a node is the length of the path from the root to the node
- Weight of a node is the number of leaves in the subtrie rooted at that node

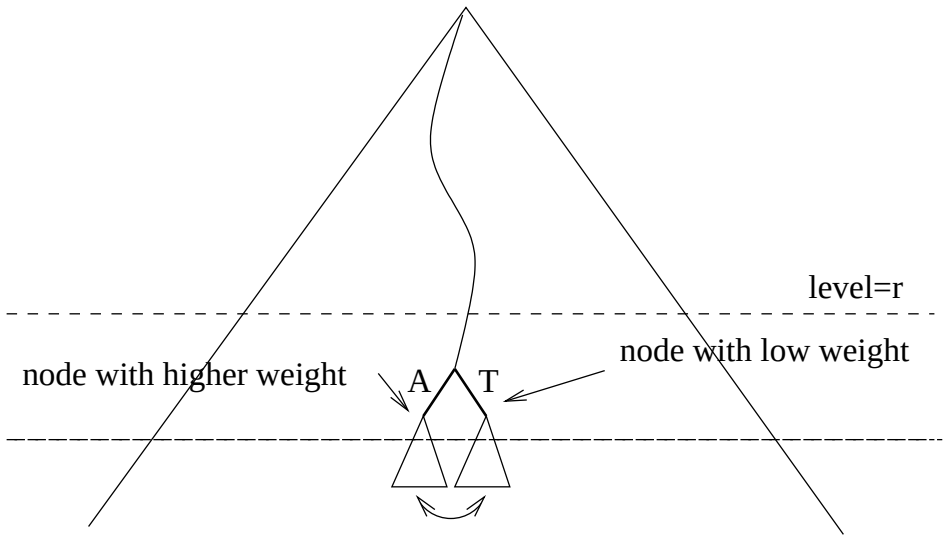
Generalized suffix trie for GATA$\$_1$ and ATGT$\$_2$



For the red node:
level=2, weight=2,
path label=AT

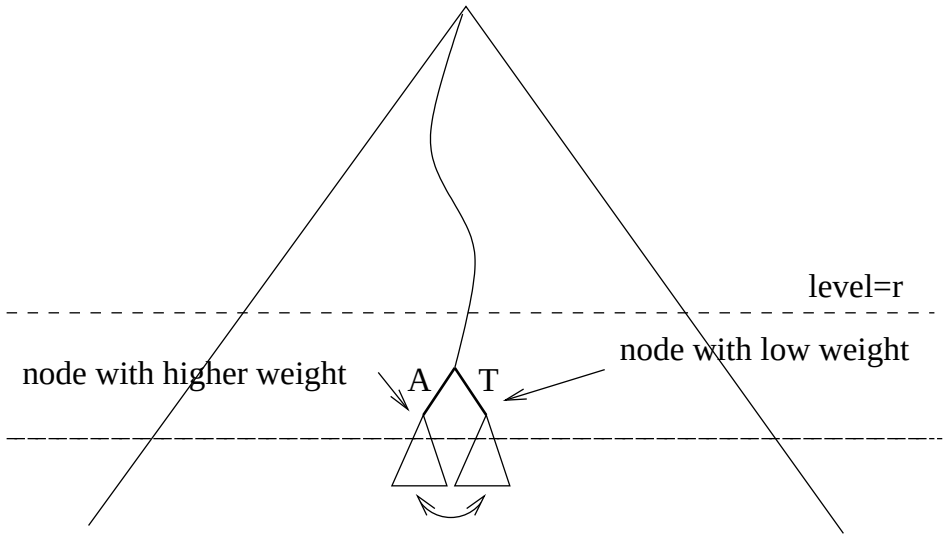
Suffix trie for error correction

- Build a suffix trie of the reads and their reverse complements
  - Top levels:
  - Intermediate levels:
  - Lowest levels:



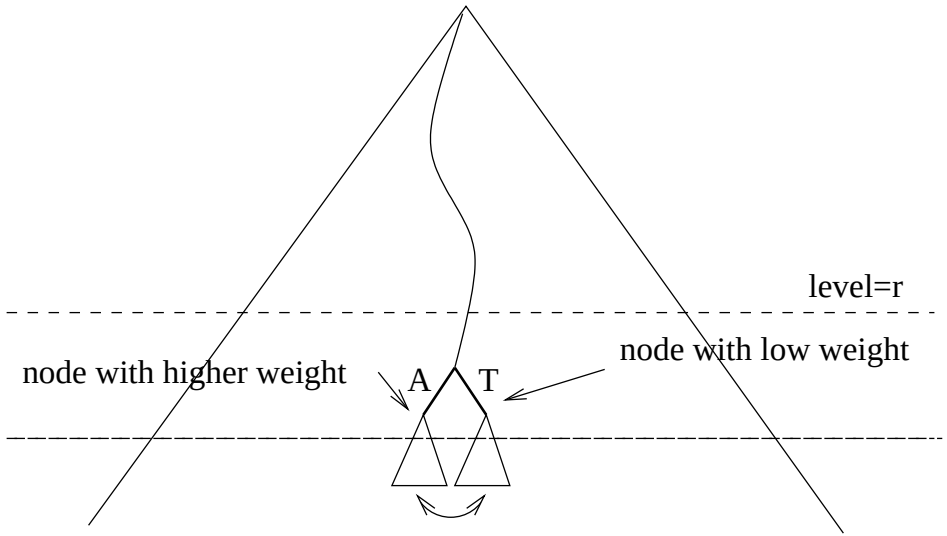
Suffix trie for error correction

- Build a suffix trie of the reads and their reverse complements
  - Top levels: Almost all nodes have four children
  - Intermediate levels: Most nodes have only one child. If there are more children, the branching is likely caused by sequencing errors.
  - Lowest levels: The weights are too small to distinguish between erroneous and correct children



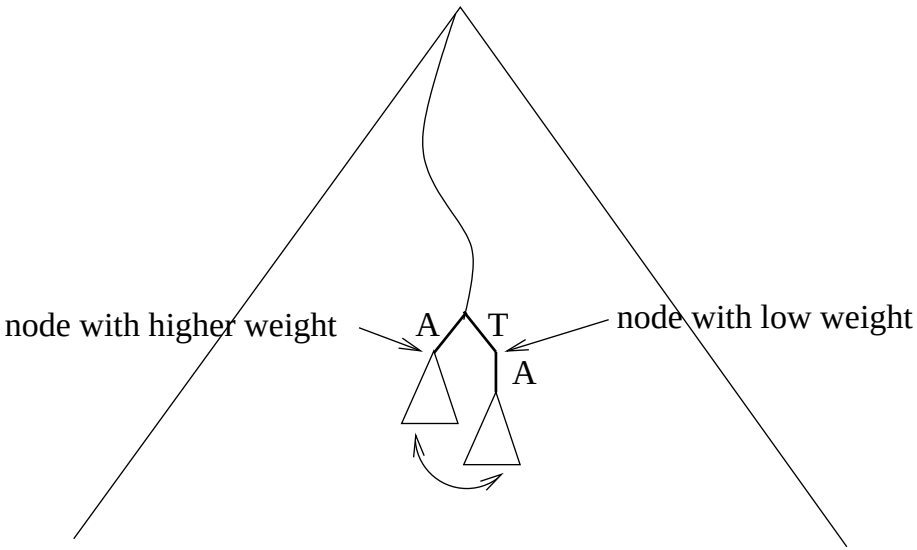
Algorithm for correcting substitutions

- Traverse the nodes at the intermediate level of the trie
  - Find a node that has more than one child and some of the children have lower than expected weight.
  - Compare the subtries rooted at the low weight node and its sibling nodes.
  - If the sibling subtrie contains the low weight subtrie, correct the error by substituting the base of the low weight node with the base of the sibling.
  - Transfer the correction to the reads.

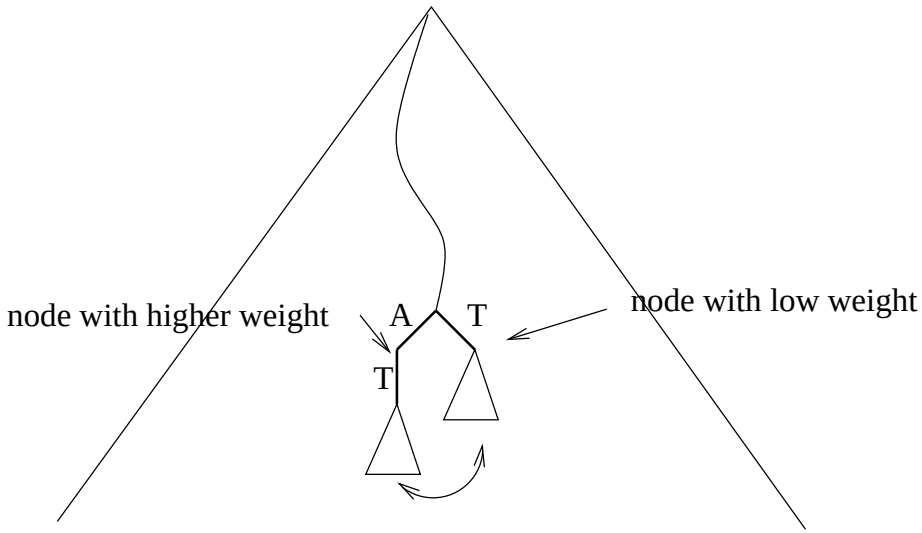


Algorithm for correcting insertions and deletions

- Insertions and deletions also cause extra branching in the suffix trie.
- These errors can thus be similarly detected and corrected.



Insertion



Deletion

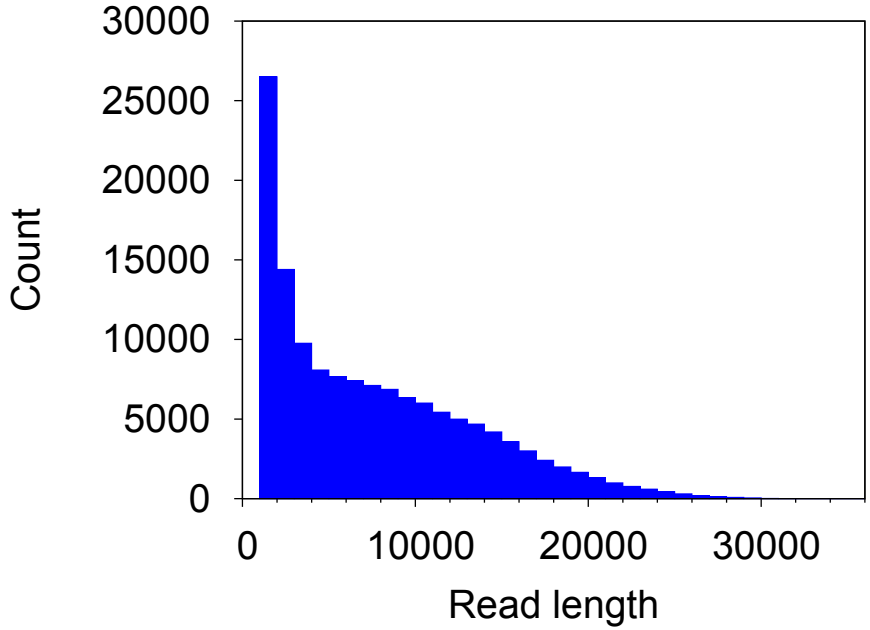
Summary of short read error correction

- k-mer spectrum based methods
- Alignment based methods
- Suffix trie/array based methods

Outline

PacBio: An example data set (E. coli K12)

- Mean read length: 7630
- Median read length: 6280
- Max read length: 35422
- Error rate: $\sim$ 15 %

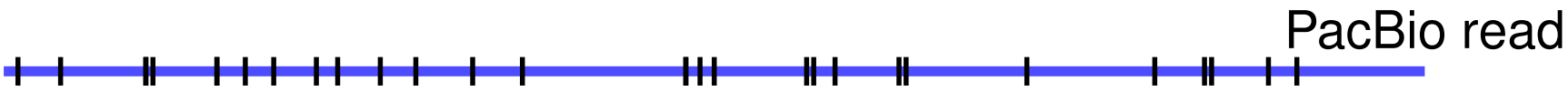



```
 Score = 2002 bits (1084),  Expect = 0.0
 Identities = 1977/2341 (84%), Gaps = 330/2341 (14%)
 Strand=Plus/Plus

Query  10122    ATCCAGTCCCCGGCAAGCTTGCTGC-AGAACTGCTCCGTGCTAAAATAGAAAGTTGCGGA  10180
                |||||||||||||||| ||| ||||| |||||||||||||||||| || ||||||||| |
Sbjct  3817612  ATCCAGTCCCCGGCA-GCT-GCTGCCAGAACTGCTCCGTGCTAAA-TA-AAAGTTGCG-A  3817666

Query  10181    ACCAGGACCCCTTCACCAC-GTTCATTCAATGCATTAGCGCGCCCGG-TTAGCGGTATTC  10238
                 ||||||||  |||| ||| |  |||| || ||    || ||||||| ||||||||||||
Sbjct  3817667  -CCAGGACCG-TTCATCACTGG-CATT-AA-GC----GC-CGCCCGGGTTAGCGGTATTC  3817716

Query  10239    CCATTGCCATCACCCAGCGAGTAAAAGGTGCTGCTTACGAGCCAGAAATAGAAACTGATG  10298
                |||||||||||||||||||||||||  ||||||||||||||||  || || ||||||
Sbjct  3817717  CCATTGCCATCACCCAGCGAGTAAAAGG--CTGCTTACGAGCCAGAT-TA-AA-CTGATG  3817771
```

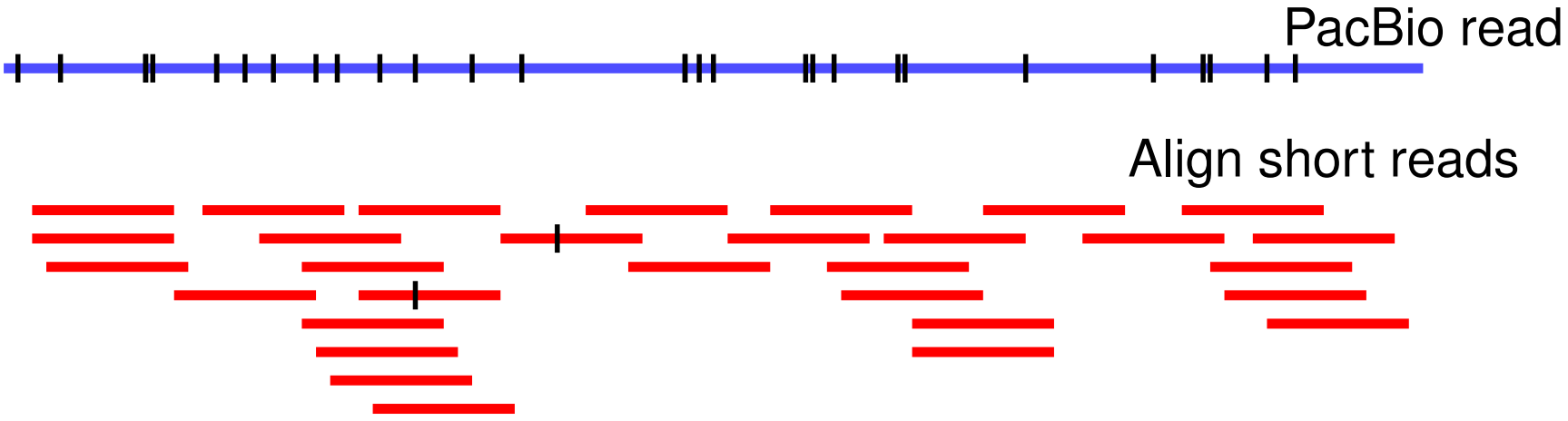
Long read error correction

- Error rate of PacBio reads is high, ~15%
- Correcting the errors simplifies further analysis like *de novo* assembly
- Two approaches:
  - Use PacBio reads only, challenge: error rate of aligning two PacBio reads is ~30%
  - Use short accurate reads (typically Illumina) to correct the PacBio reads

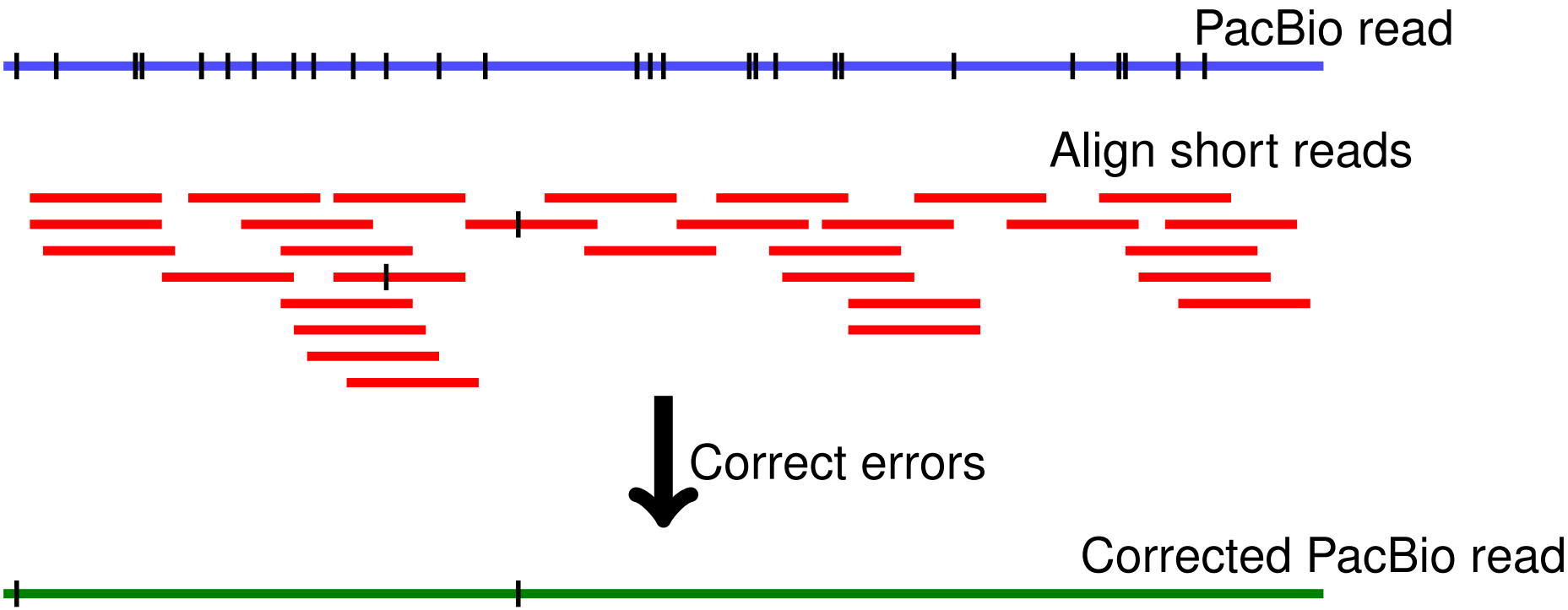
Alignment based approaches



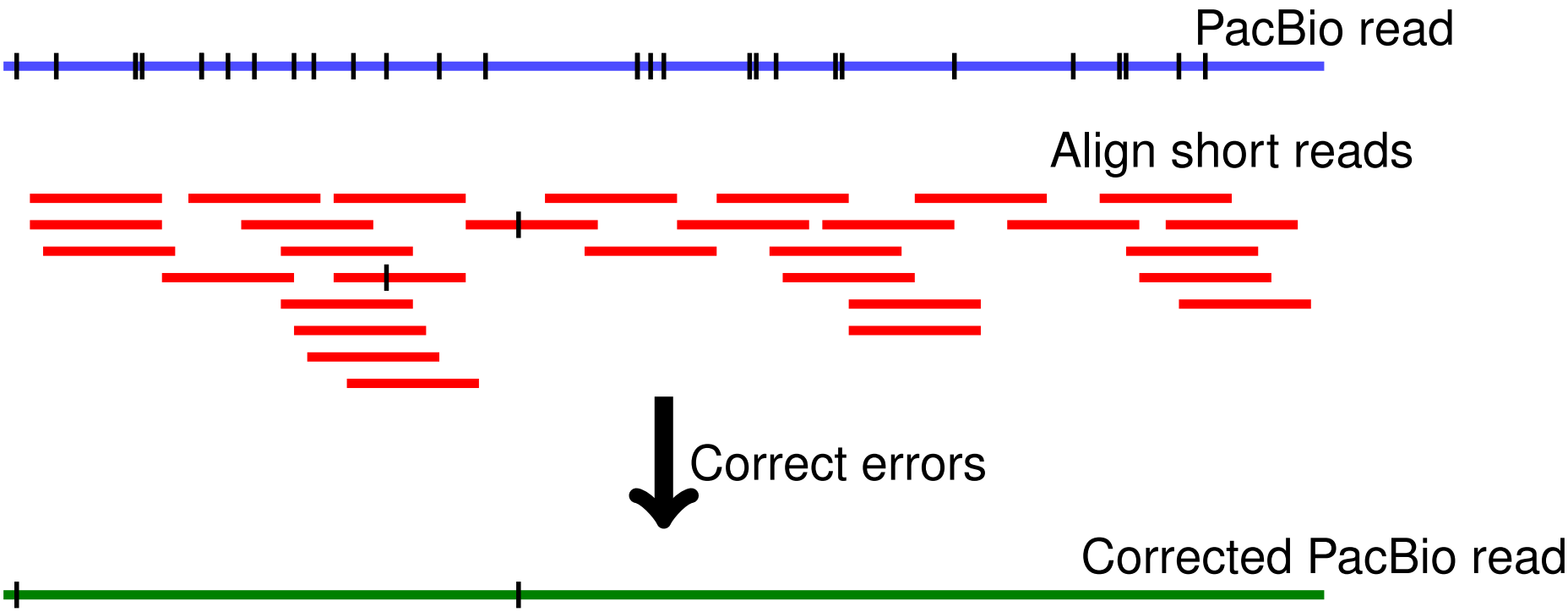
Alignment based approaches



Alignment based approaches



Alignment based approaches



- E.g. PacBioToCA, LSC
- Problem: Aligning short reads to the PacBio reads allowing for high error rate is slow.

De Bruijn graph (DBG)

- Given a set of reads $R_1 ... R_n$
- Extract all k-mers that occur in the reads
- Form a graph:
  - the k-mers are the vertices in the graph
  - draw an edge between two k-mers if they overlap by $k - 1$ bases

De Bruijn graph (DBG)

- Given a set of reads $R_1 ... R_n$
- Extract all k-mers that occur in the reads
- Form a graph:
  - the k-mers are the vertices in the graph
  - draw an edge between two k-mers if they overlap by $k - 1$ bases
- (Alternative representation: $(k - 1)$-mers are vertices, an edge between two vertices if the k-mer occurs in the reads)

De Bruijn graph: Example

- Reads: ATGCGT, GCGTGG, GTGGCA
- 3-mers:

De Bruijn graph: Example

- Reads: ATGCGT, GCGTGG, GTGGCA
- 3-mers: ATG

De Bruijn graph: Example

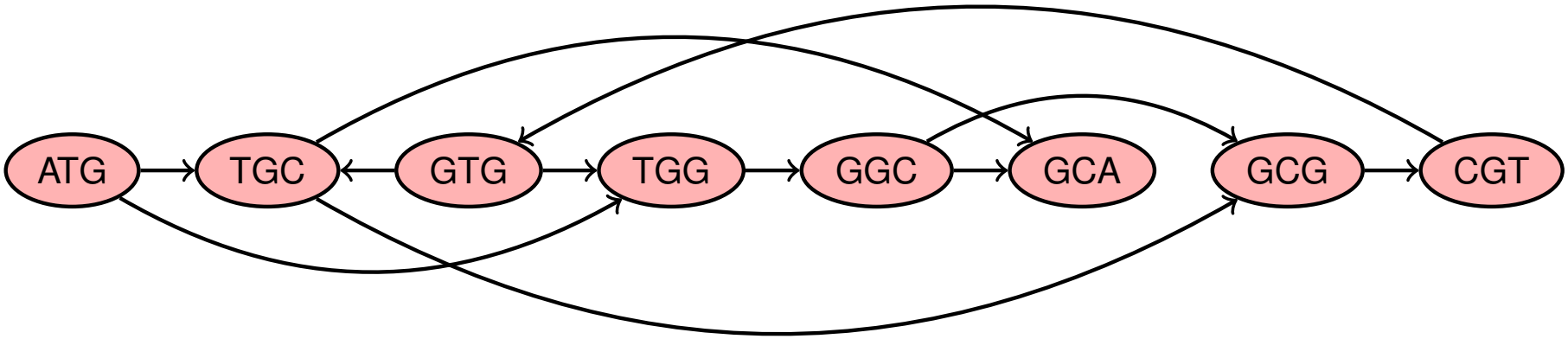
- Reads: ATGCGT, GCGTGG, GTGGCA
- 3-mers: ATG,TGC

De Bruijn graph: Example

- Reads: ATGCGT, GCGTGG, GTGGCA
- 3-mers: ATG,TGC,GCG, CGT, GTG, TGG, GGC, GCA

De Bruijn graph: Example

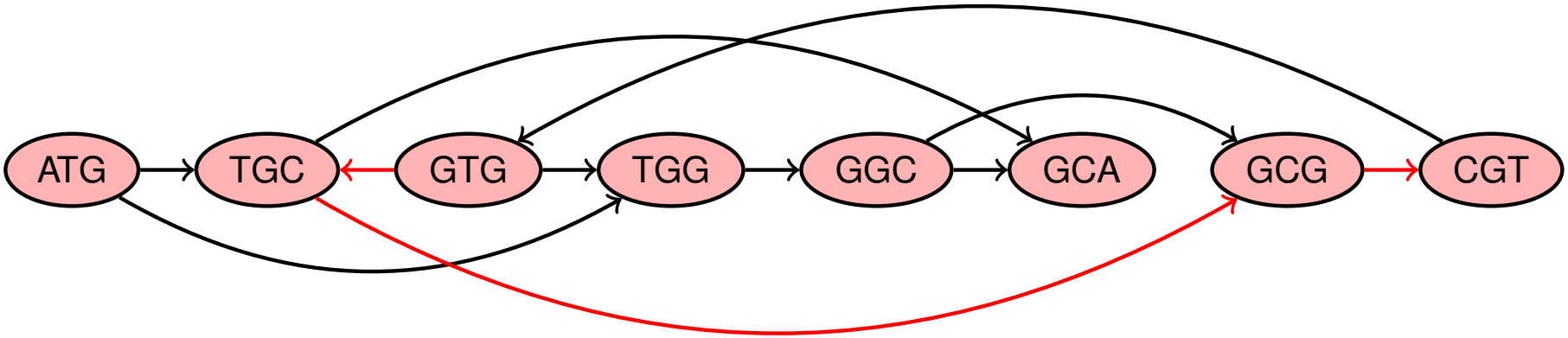
- Reads: ATGCGT, GCGTGG, GTGGCA
- 3-mers: ATG,TGC,GCG, CGT, GTG, TGG, GGC, GCA
- DBG:



DBG: Paths spell strings

- Consider the red path in the graph
- The string spelled by this path is

```
G T G
  T G C
    G C G
      C G T
-----------
G T G C G T
```

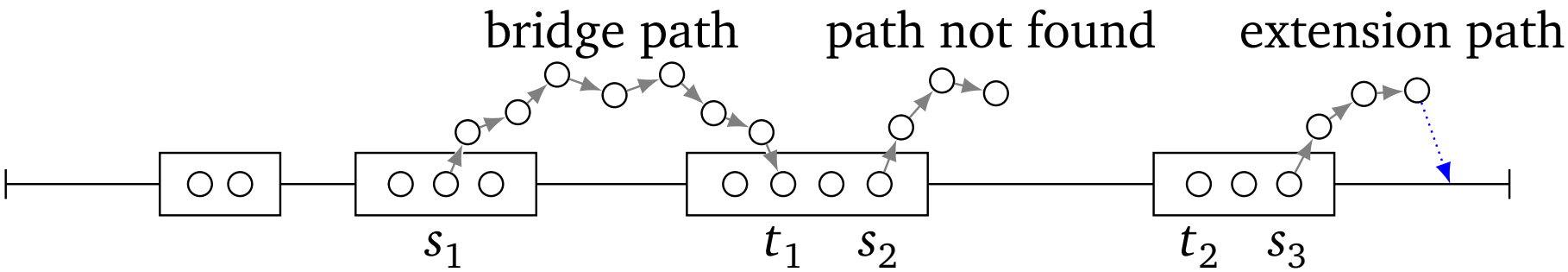


DBG based error correction: overview

- Build a DBG of the short accurate reads
- Align the long PacBio reads against the DBG
- Correct the long read based the alignment path in the DBG

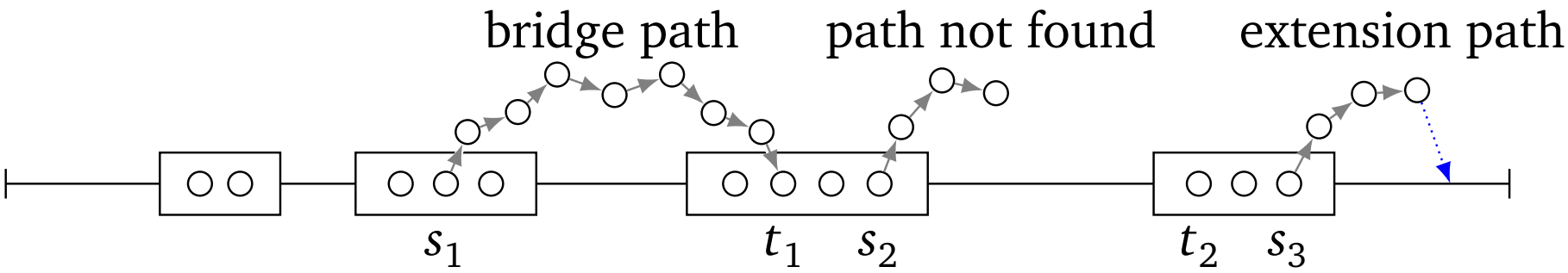
Solid and weak regions in long reads

- Classify k-mers in the long reads:
  - solid: in the DBG
  - weak: not in the DBG
- The long read now consists of solid and weak regions



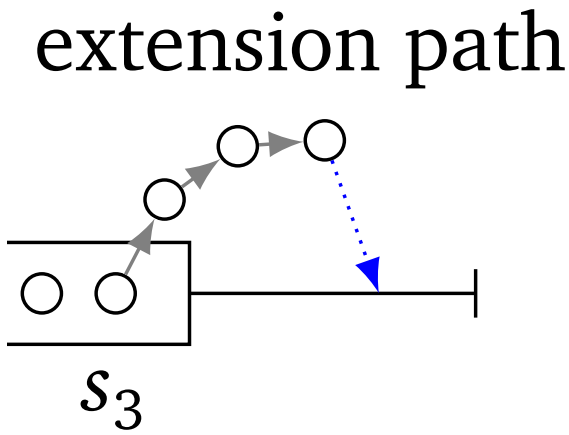
Correction a weak region in a long read

- Find paths in the DBG between the flanking solid k-mers
- Minimize edit distance between the long read and the string spelled by the path.
- Allow only limited branching



Correcting weak heads/tails of long reads

- Find a path in DBG starting from the extreme solid k-mer
- Maximize length of the prefix of the end to correct
- Minimize edit distance between the path and the prefix of the end
- Find best extension maximizing an alignment score



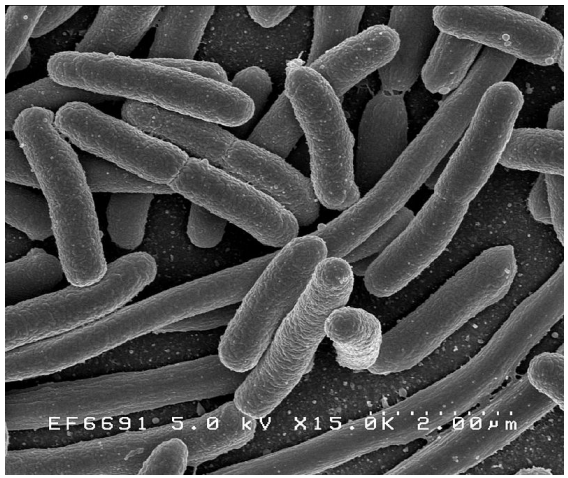
What if short reads are not available?

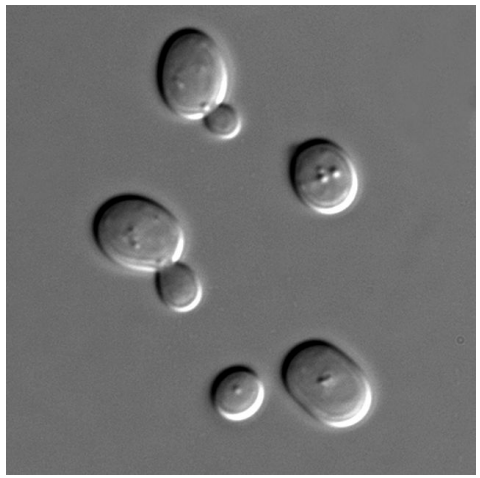
- We can use the DBG based approach if the graph can be built on long reads only
- This is possible if
  - k is small
  - coverage of the read set is high

  $\Longrightarrow$ k-mers are present in the reads with high enough abundancy
- Iterative approach with increasing k is beneficial
- Additional step based on multiple alignments is beneficial to take advantage of the long range information

Data sets

	E. coli	Yeast
Genome size	4.6 Mbp	12 Mbp
PacBio coverage	208x	129x
Illumina coverage	50x	38x





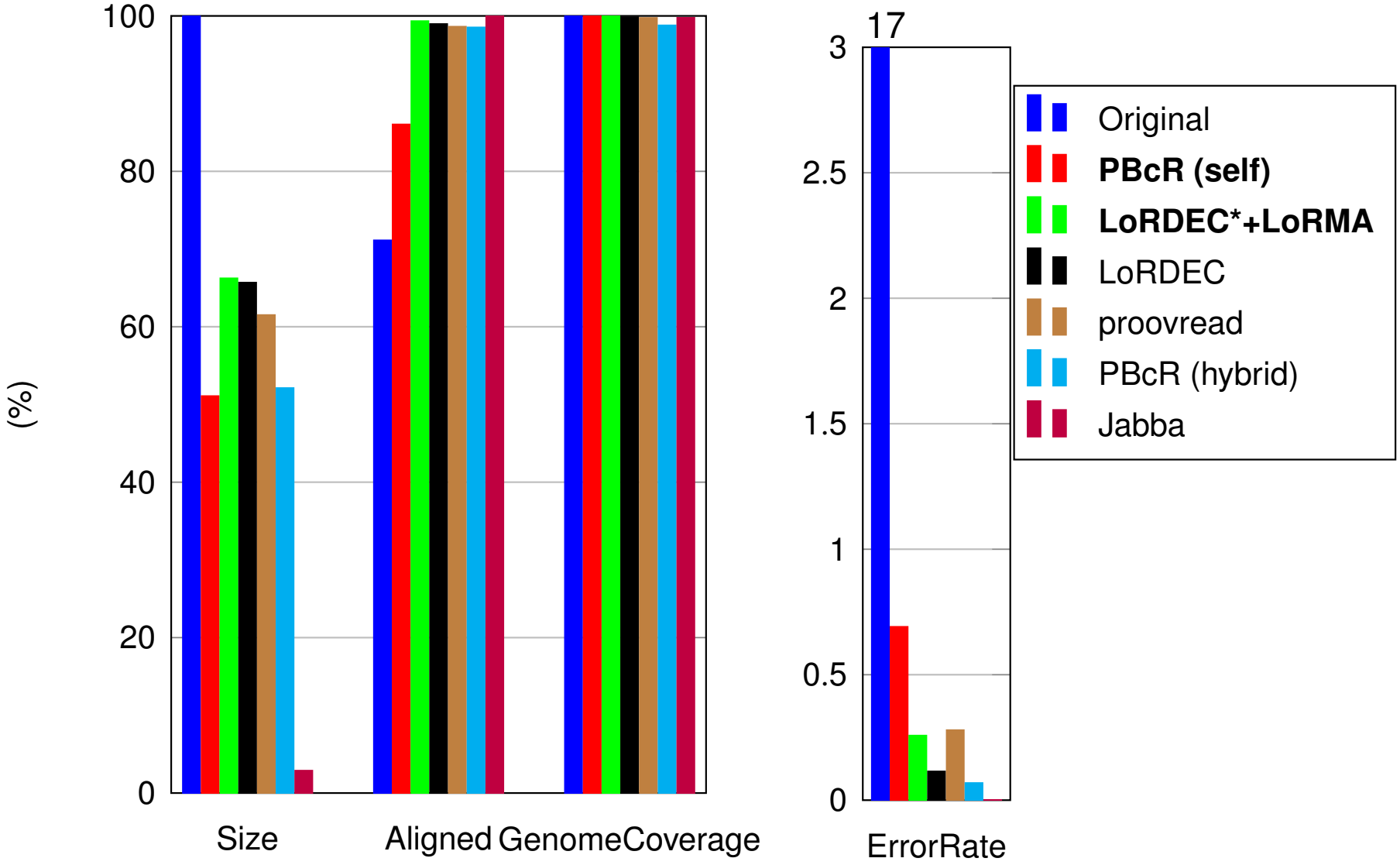
Error correction tools

- Selfcorrection:
  - LoRDEC*+LoRMA (DBG + alignment)
  - PBcR (Alignment)
- Hybrid correction
  - LoRDEC and Jabba (DBG)
  - PBcR (Alignment)
  - proovread (Alignment)

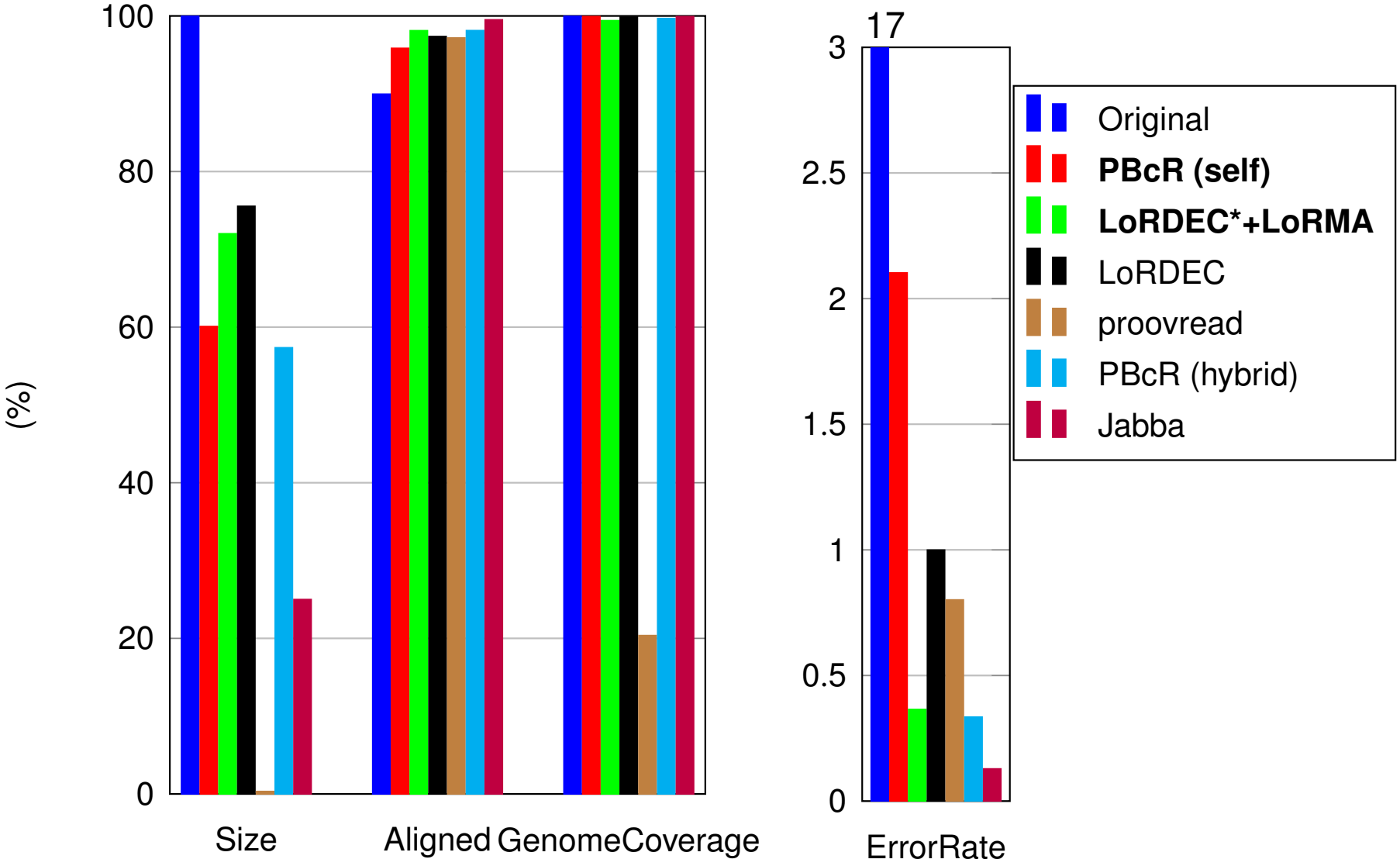
Aligning against reference

- Reads were aligned against the reference with BLASR.
- We measured
  - The proportion of reads that were corrected
  - The proportion of reads that was aligned
  - The genome coverage
  - The error rate

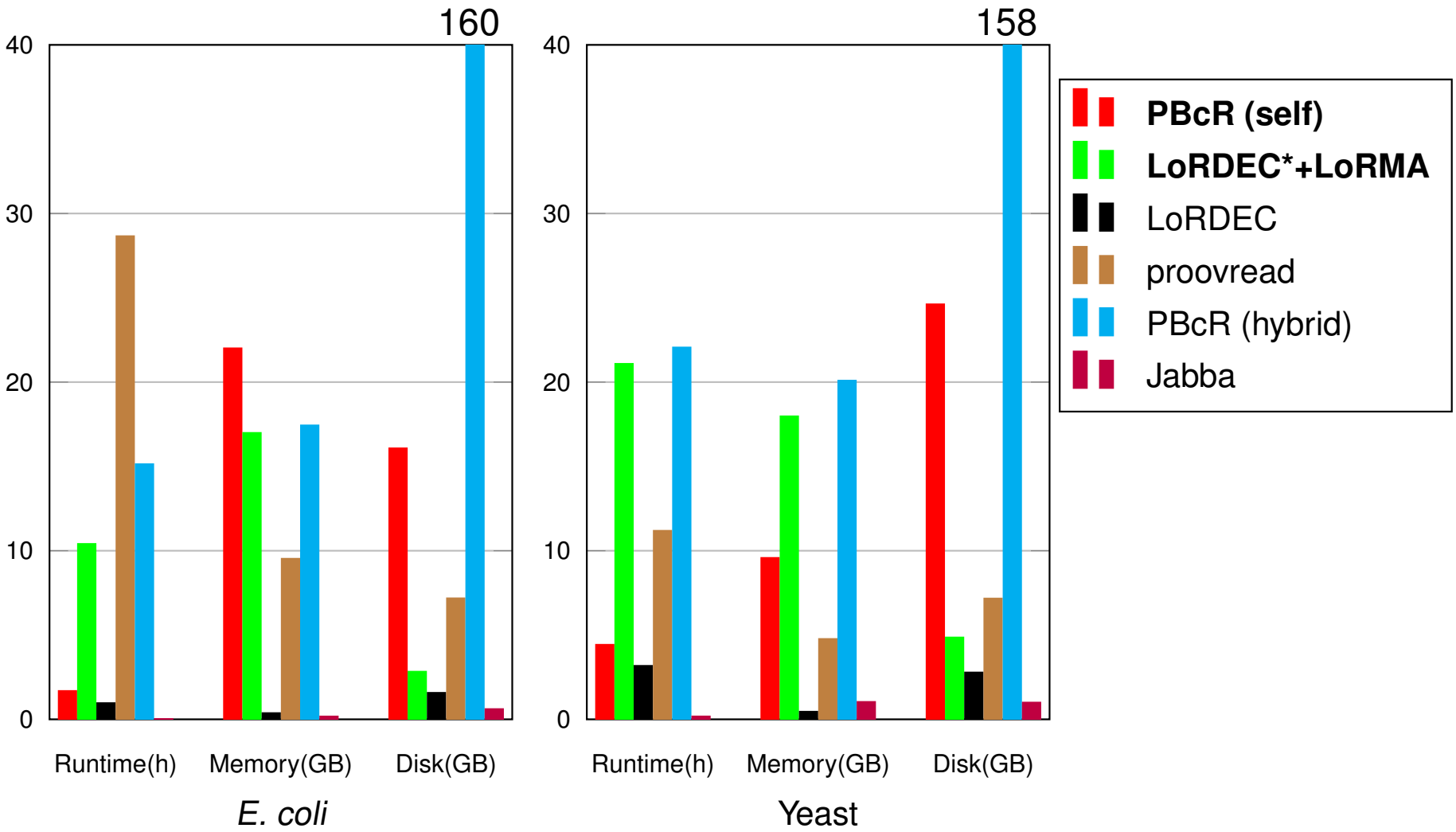
E. coli


100
80
60
40
20
0
(%)
Size
Aligned
GenomeCoverage
17
3
2.5
2
1.5
1
0.5
0
ErrorRate
Original
PBcR (self)
LoRDEC*+LoRMA
LoRDEC
proovread
PBcR (hybrid)
Jabba

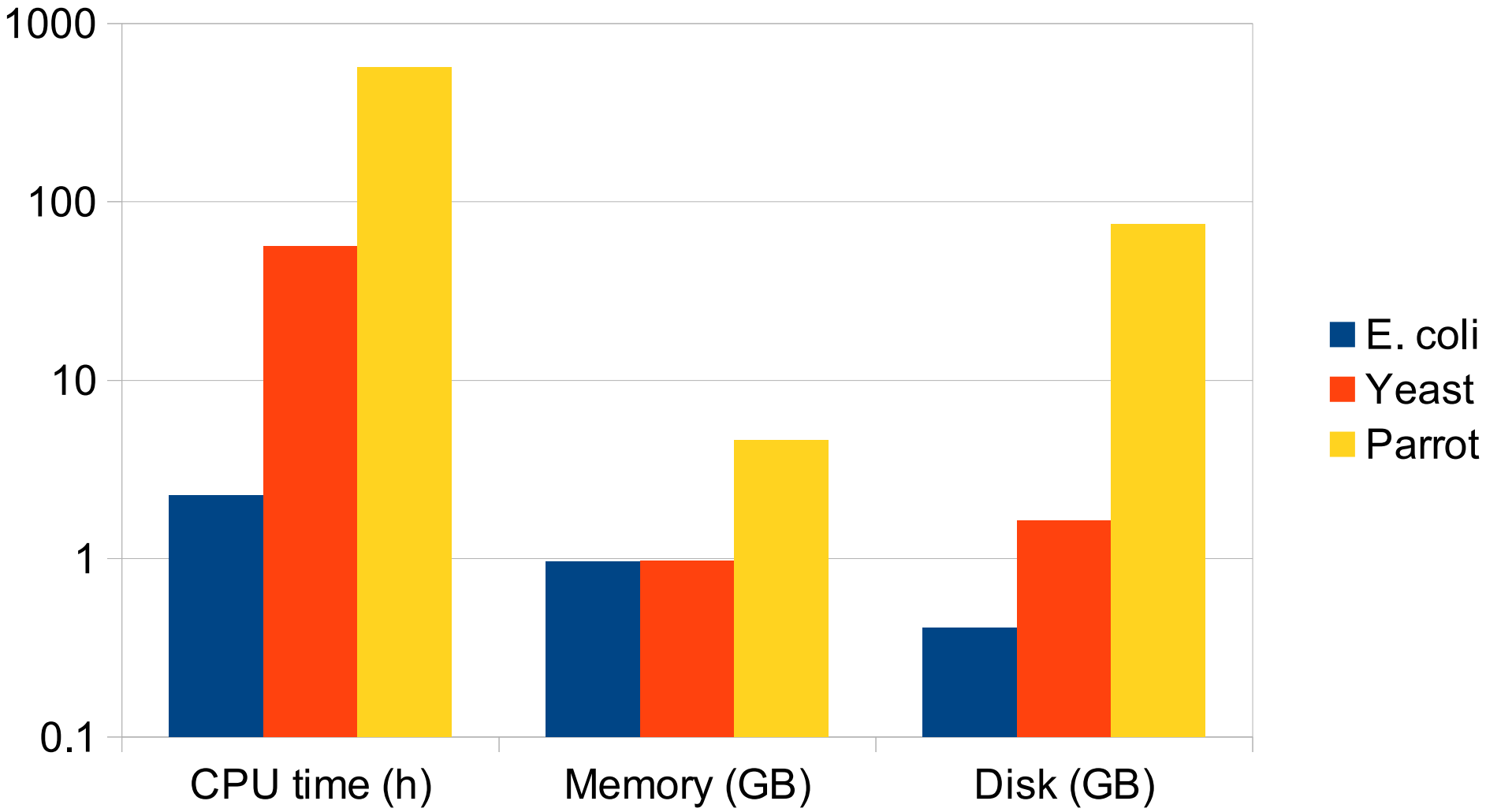
Yeast


100
80
60
40
20
0
(%)
Size
Aligned
GenomeCoverage
17
3
2.5
2
1.5
1
0.5
0
ErrorRate
Original
PBcR (self)
LoRDEC*+LoRMA
LoRDEC
proovread
PBcR (hybrid)
Jabba

Resources


160
158
40
30
20
10
0
Runtime(h)
Memory(GB)
Disk(GB)
E. coli
Yeast
PBcR (self)
LoRDEC*+LoRMA
LoRDEC
proovread
PBcR (hybrid)
Jabba

Scaling to the parrot data (LoRDEC)



Summary of long read error correction

- Hybrid correction: using also short reads
- Selfcorrection: using only long reads
- Alignment based approach
- DBG based approach

Acknowledgements

- Eric Rivals, University of Montpellier
- Esko Ukkonen, University of Helsinki
- Riku Walve, University of Helsinki
- Jan Schröder, Walter and Eliza Hall Institute of Medical Research, Melbourne