

---

# De-Novo Genome Assembly and its Current State

Anne-Katrin Emde

April 17, 2013

Freie Universität Berlin, Algorithmische Bioinformatik

Max Planck Institut für Molekulare Genetik, Computational Molecular Biology

---



**International  
Max Planck Research School**  
for Computational Biology  
and Scientific Computing

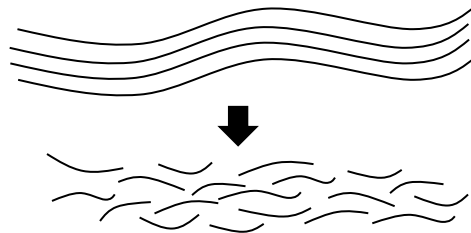
Freie Universität  Berlin

---



# What is genome assembly and why is it hard?

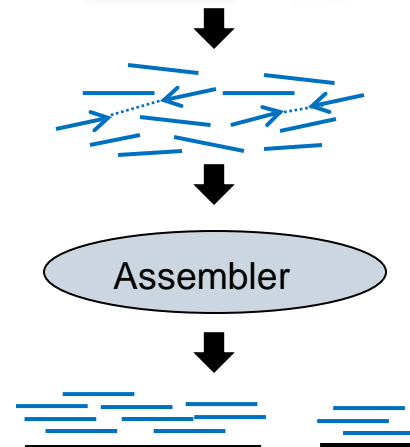
Original genome sequence  
(in multiple copies)



Sequence short fragments



Reconstruct original sequence from *reads*



Difficulties:

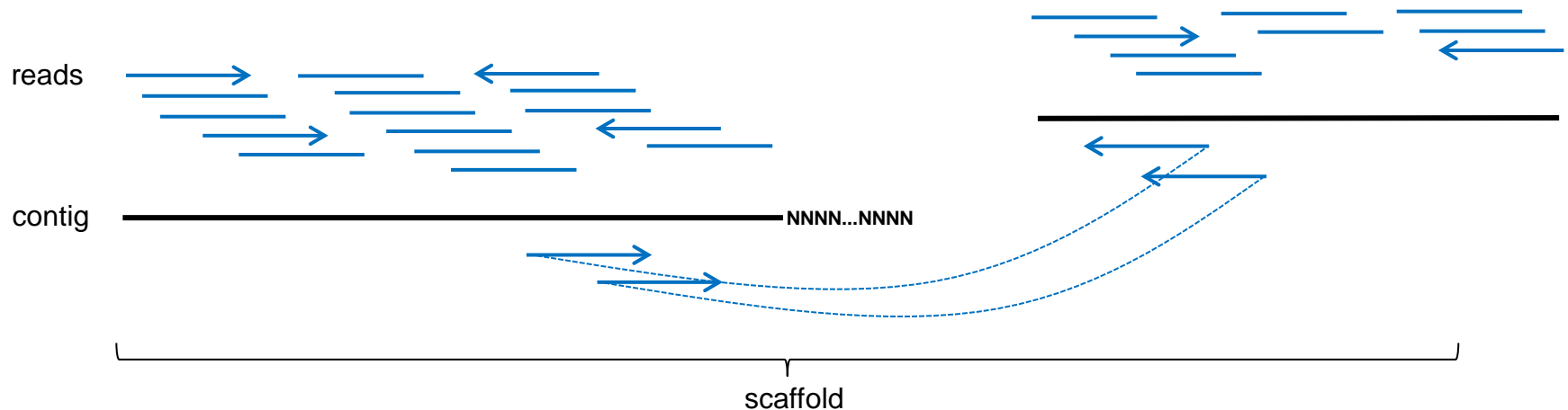


- Genomes are very long and repeat-rich
- Reads are very short and may contain errors and biases



# Assembly algorithms - introduction

Assemblers use **overlaps** between reads, to first produce **contigs**, that are then used to build **scaffolds**.



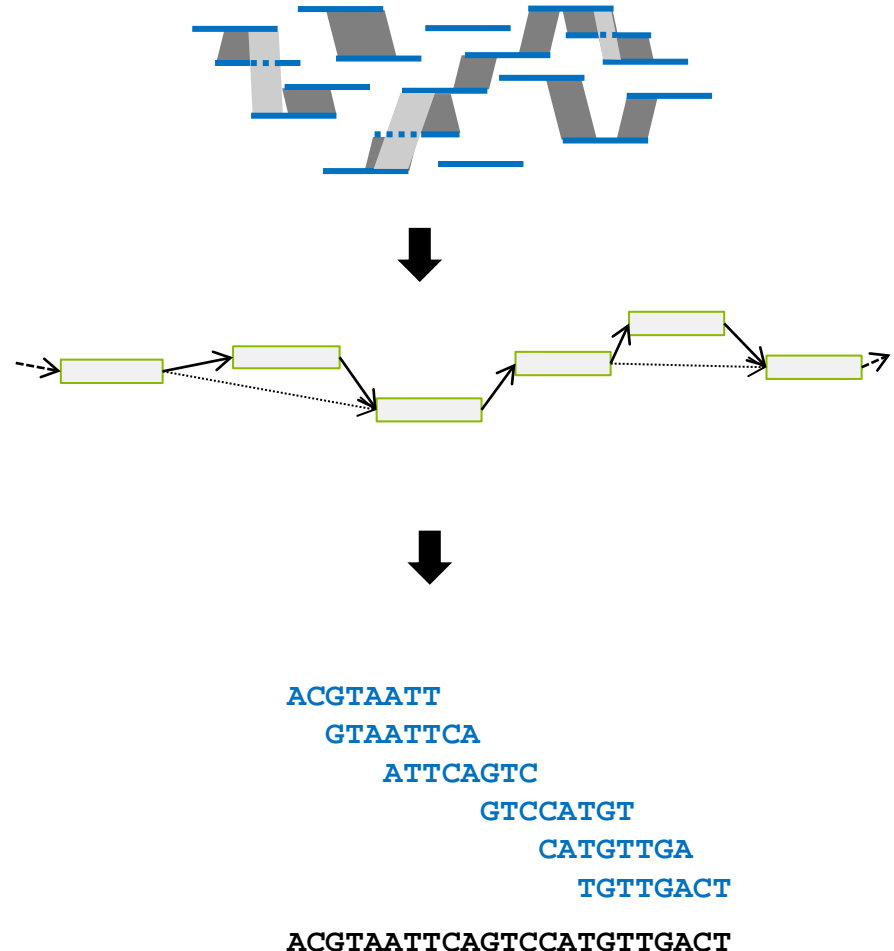
*Read pairs* contribute long-range linking information, especially in the scaffolding phase.

## 2 Paradigmen

- 1) Overlap – Layout – Consensus: Focus is on sequence reads
- 2) De Bruijn graph: Focus is on k-tuples occurring in sequence reads

# Algorithms – Overlap Graph

- 1) **Overlap phase:** pairwise overlap alignments
- 2) **Layout phase:** overlap graph construction and finding relative placement of reads
- 3) **Consensus phase:** Produce multiple read alignment and compute contig consensus sequences

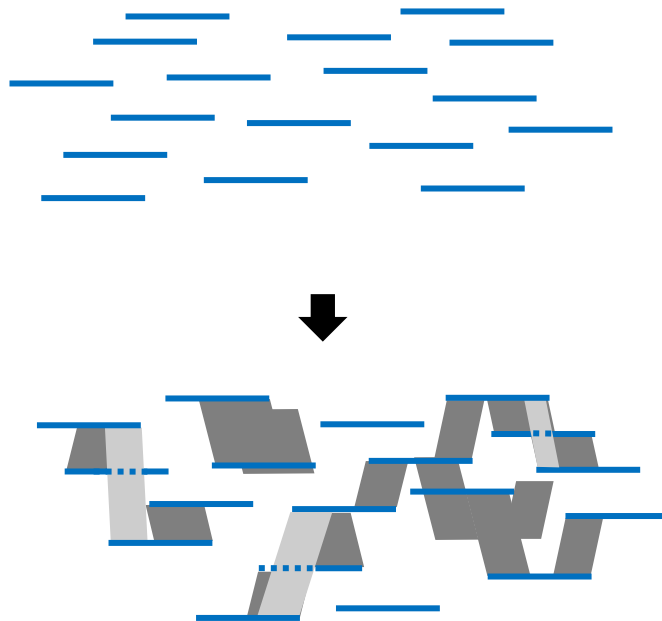


*Kececioğlu and Myers, 1995*

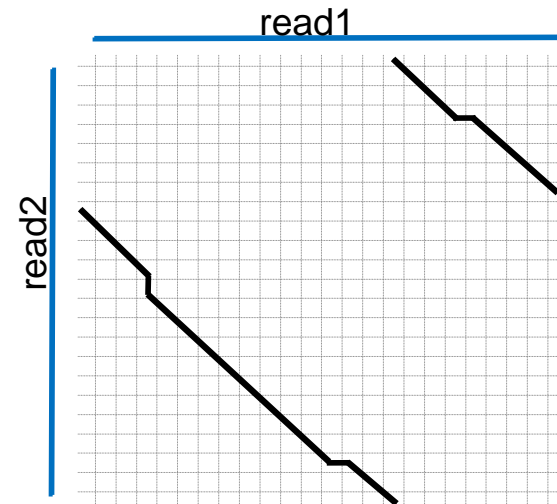
# Algorithms – Overlap Graph

## 1) Overlap phase:

for all pairs of reads



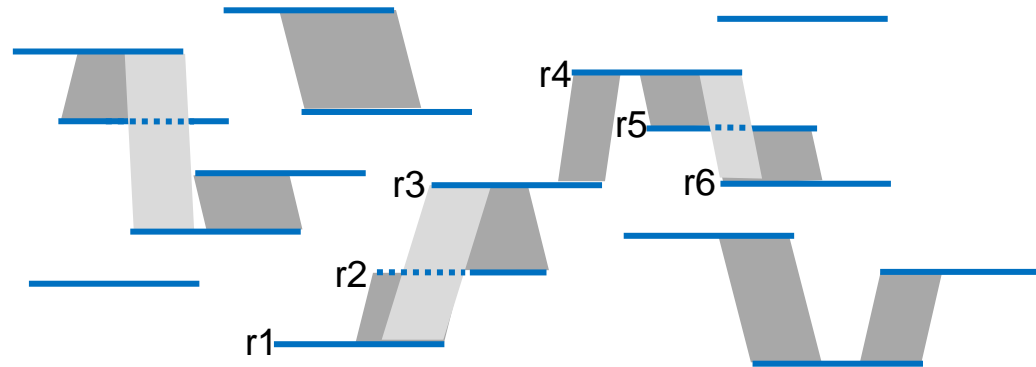
pairwise overlap alignments



Computationally expensive!

# Algorithms – Overlap Graph

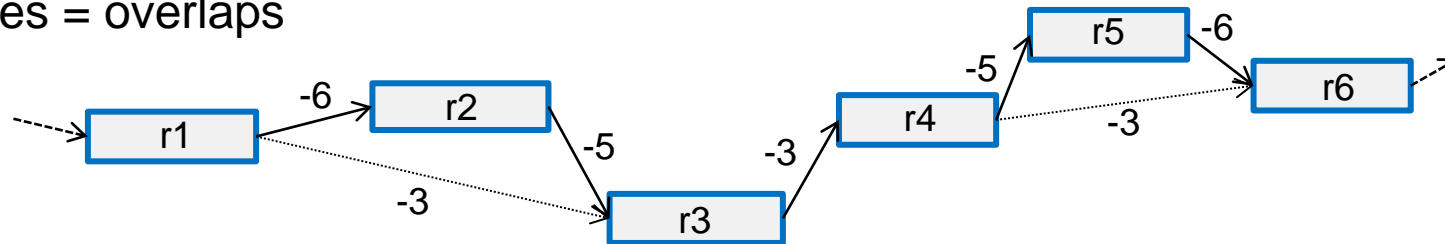
## 1) Overlap phase:



## 2) Layout phase:

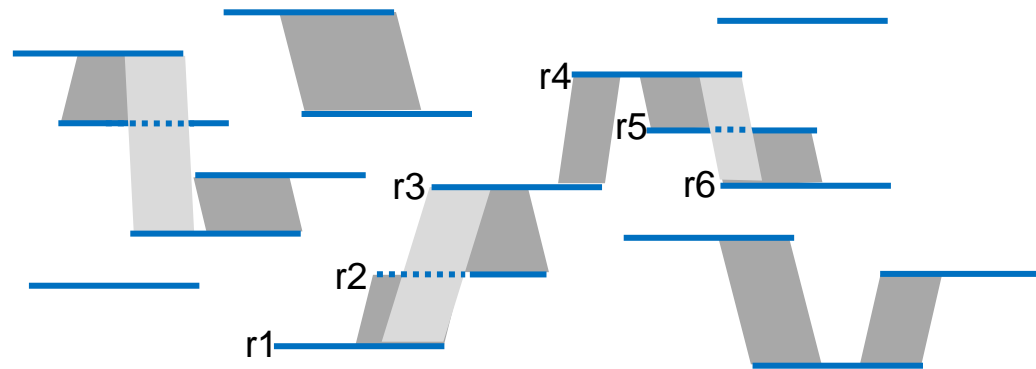
nodes = reads

edges = overlaps



# Algorithms – Overlap Graph

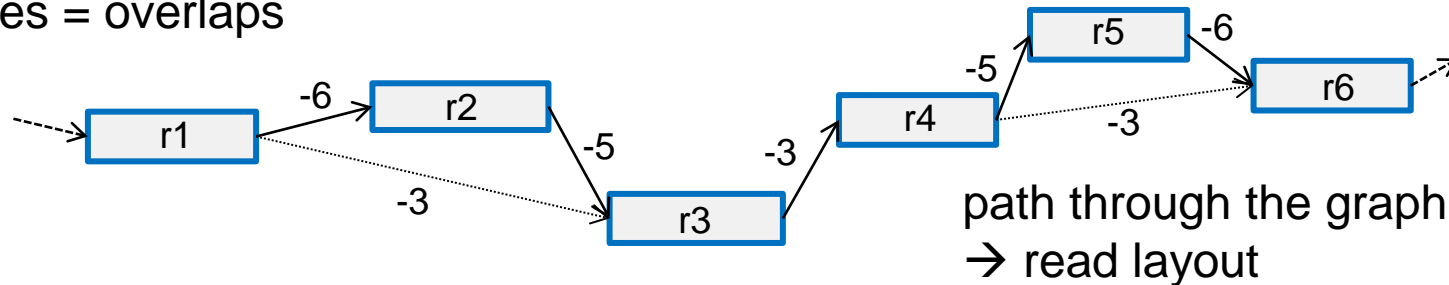
## 1) Overlap phase:



## 2) Layout phase:

nodes = reads

edges = overlaps



In theory Hamiltonian path (NP-complete), in practice heuristics

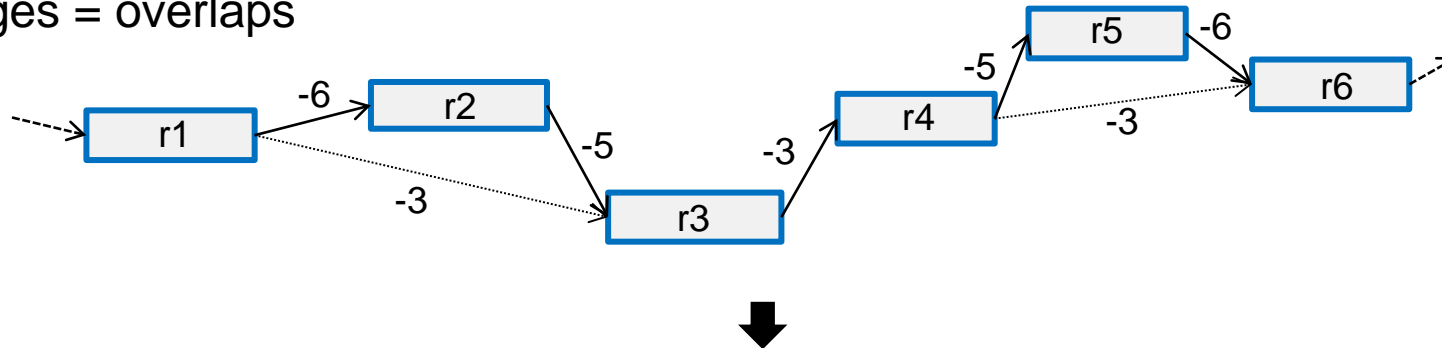


# Algorithms – Overlap Graph

## 2) Layout phase:

nodes = reads

edges = overlaps



## 3) Consensus phase:

multiple read alignment

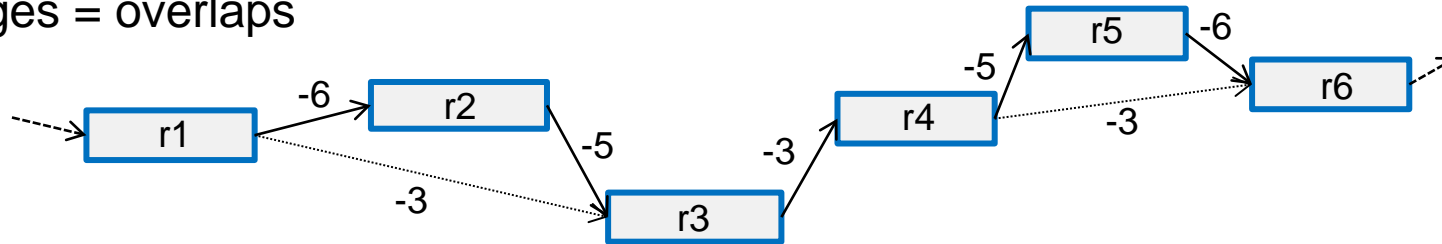
r1 **ACGTAATT**  
 r2 **GTAATTCA**  
 r3 **ATTCAGTC**  
 r4 **GTCCATGT**  
 r5 **CATGTTGA**  
 r6 **TGTTGACT**  
 contig **ACGTAATTCAGTCCATGTTGACT**

# Algorithms – Overlap Graph

## 2) Layout phase:

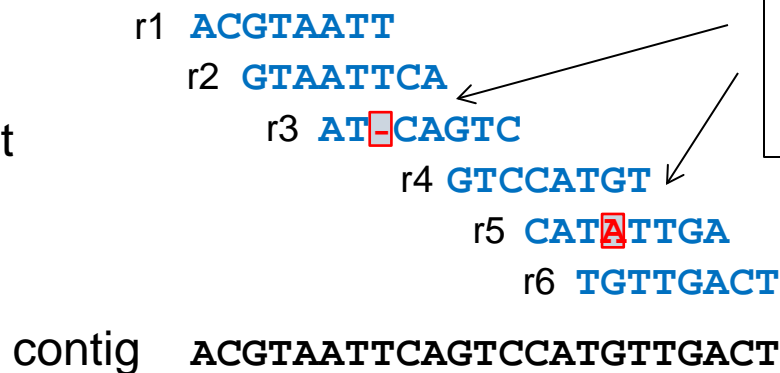
nodes = reads

edges = overlaps

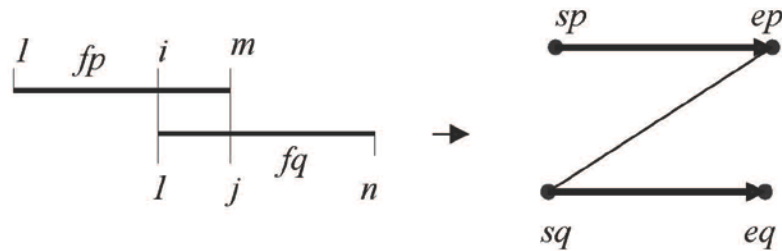


## 3) Consensus phase:

multiple read alignment



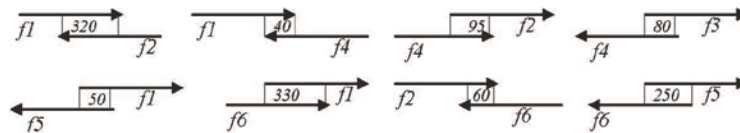
robust with alignment errors



The label (or "length") of the overlap edge  $e$  is defined to be  $-1$  times the overlap length, e.g.  $-(\frac{m-i+j-1}{2} + 1)$  in the figure.

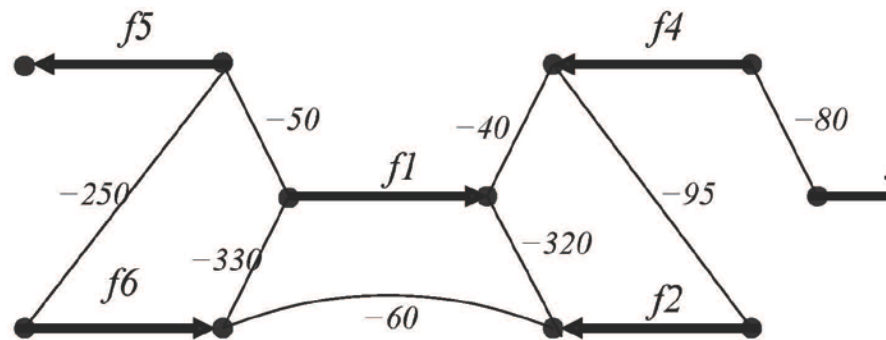
### 10.14 Example

Assume we are given 6 reads  $\mathcal{F} = \{f_1, f_2, \dots, f_6\}$ , each of length 500, together with the following overlaps:



Here, for example, the last 320 bases of read  $f_1$  align to the first 320 bases of the reverse complement  $\bar{f}_2$  of  $f_2$ , whereas  $f_1$  and  $\bar{f}_5$  overlap in the first 50 bases of each.

We obtain the following overlap graph OG:

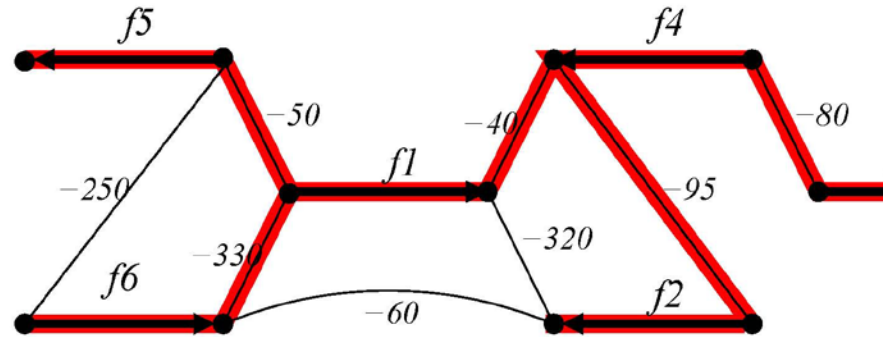


Each read  $f_p$  is represented by a read edge  $(s_p, e_p)$  of length  $|f_p|$ . Overlaps off the start  $s_p$ , or end  $e_p$ , of  $f_p$  are represented by overlap edges starting at the node  $s_p$ , or  $e_p$ , respectively. Each overlap edge is labeled by  $-1$  times the overlap length.

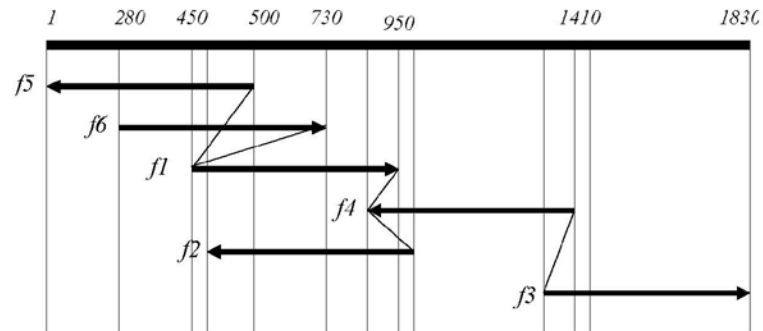
## 10.15 The layout phase

The goal of the layout phase is to arrange all reads into an approximate multi-alignment. This involves assigning coordinates to all nodes of the overlap graph  $OG$ , and thus, determining the value of  $s_i$  and  $e_i$  for each read  $f_i$ .

A simple heuristic is to select a *spanning forest* of the overlap graph  $OG$  that contains all read edges.<sup>1</sup>



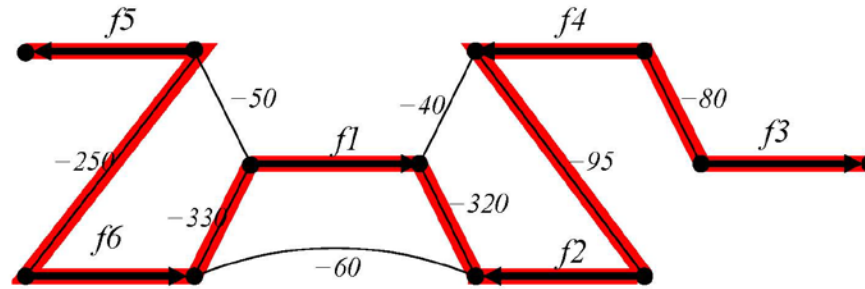
Such a subset of edges positions every read with respect to every other, within a given connected component of the graph:



Such a putative alignment of reads is called a *contig*.

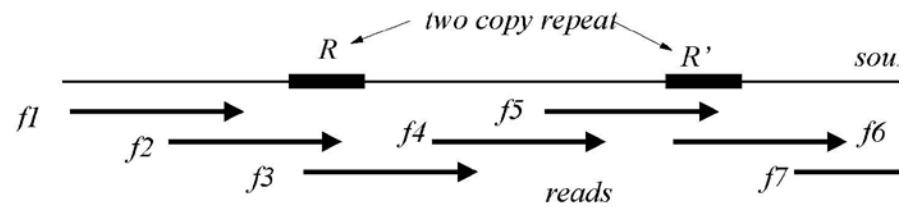
The spanning tree is usually constructed using a *greedy heuristic* in which the overlap edges are chosen in order of decreasing overlap length (i.e., increasing edge “length”).

<sup>1</sup>(A spanning forest is a set  $F$  of edges such that any two nodes in the same connected component of  $OG$  are connected by a unique simple, unoriented path of edges in  $F$ .)

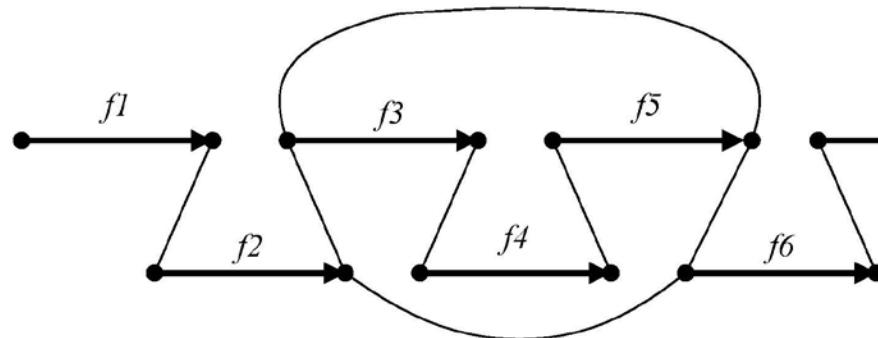


### 10.16 Repeats and the layout phase

Consider the following situation:



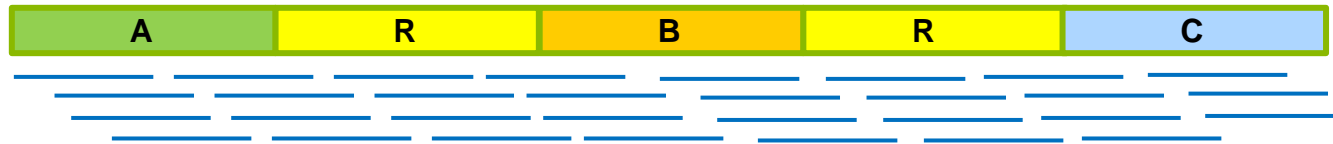
This gives rise to the following overlap graph:



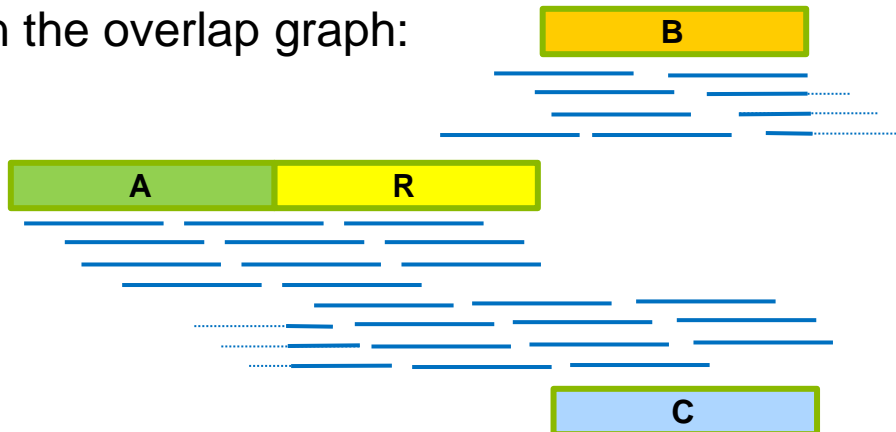
Consider this spanning tree:

# Overlap Graph

There are not only simple paths...



Approximate ordering  
in the overlap graph:

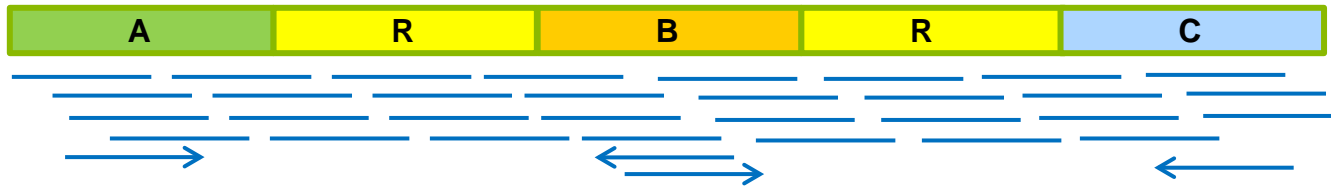


- Coverage is high
- Path branches

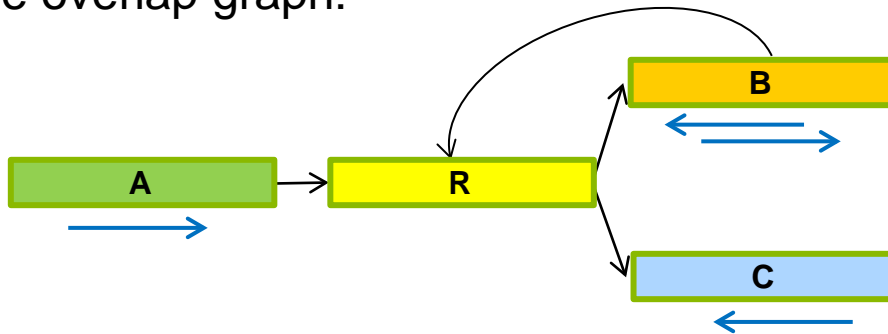
**R** is a repeat!

# Overlap Graph

There are not only simple paths...

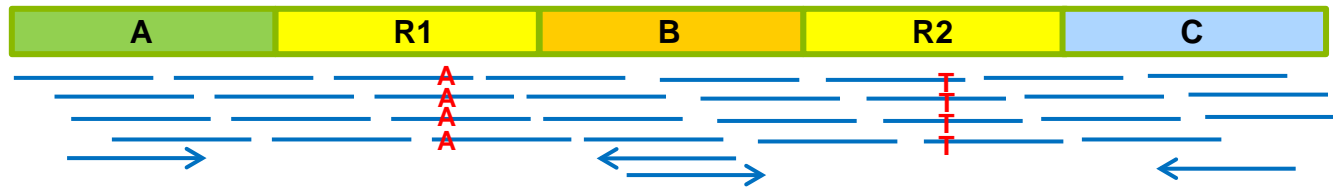


Approximate ordering  
in the overlap graph:

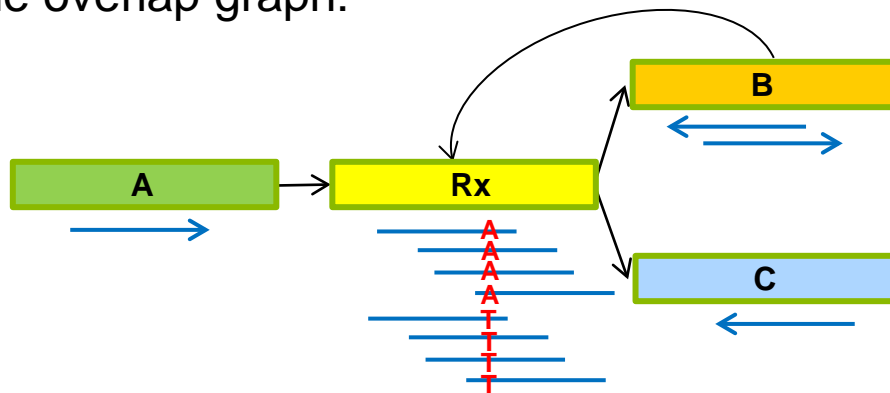


# Overlap Graph

Now: R1 and R2 are nearly identical



Approximate ordering  
in the overlap graph:

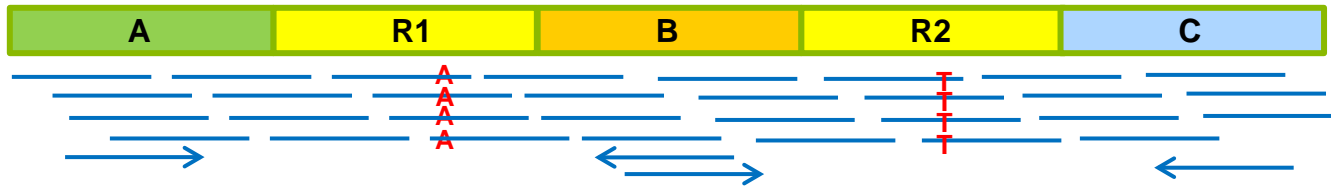


**Overlap strictness:**  
Tradeoff between error tolerance  
and “natural” repeat separation

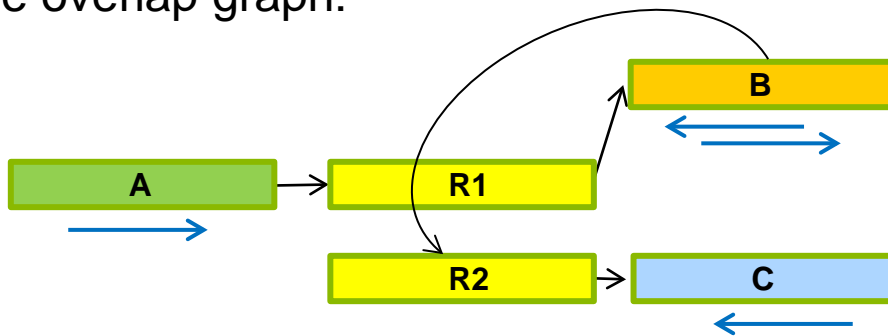


# Overlap Graph

Now: R1 and R2 are nearly identical



Approximate ordering  
in the overlap graph:



**Overlap strictness:**  
Tradeoff between error tolerance  
and “natural” repeat separation

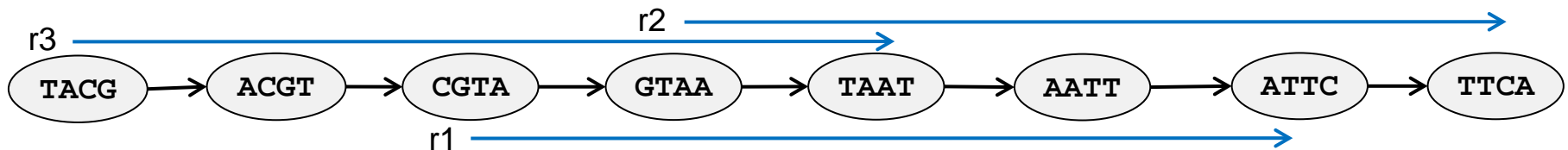
# Algorithms - de Bruijn graph (Euler assembler)

- No overlap phase, no consensus phase, basically just a layout phase

Nodes =  $k$ -mers  
 Edges =  $(k+1)$ -mers

Given  $k = 4$  and three read sequences:

r1 **CGTAATTC**  
 r2 **GTAATTCA**  
 r3 **TACGTAAT**



contig: **TACGTAATTCA**

r3 **TACGTAAT**  
 r1 **CGTAATTC**  
 r2 **GTAATTCA**

In theory „de Bruijn Superwalk Problem“ (NP-hard), in practice heuristics

# de Bruijn graph

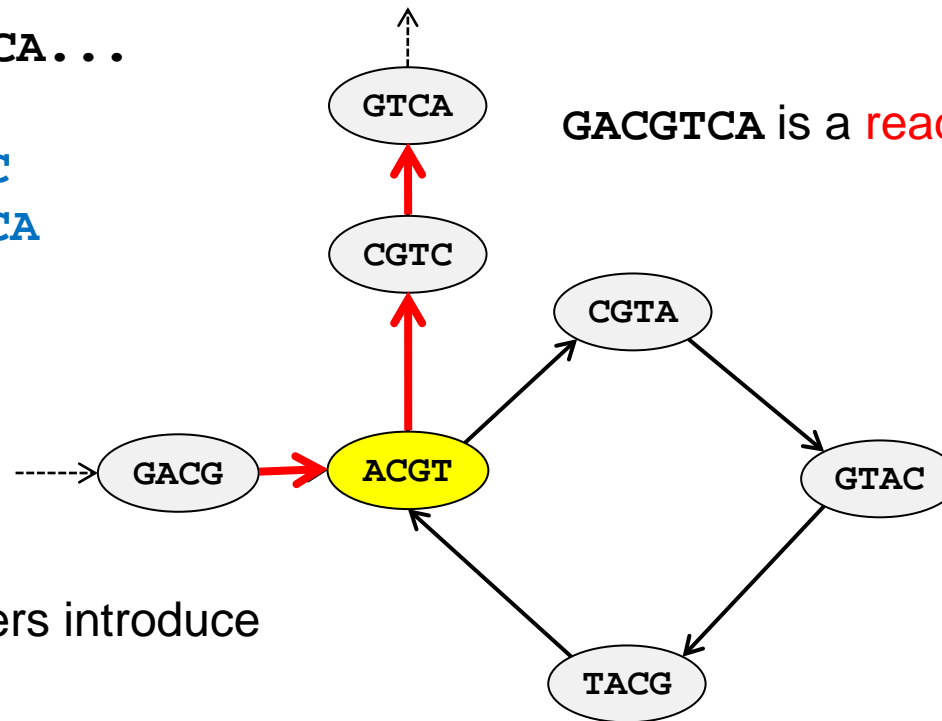
Again, there are not only simple paths...

...G**ACGT****ACGT**CA...

GACGTACG

CGTACGTC

GTACGTCA



GACGTCA is a **read-incoherent** path

Repetitive k-mers introduce cycles

# de Bruijn graph

What if we increase  $k$  to 5?

...GACGTACGTCA...

GACGTACG

CGTACGTC

GTACGTCA



→ Back to a linear graph structure

Increasing  $k$  leads to better repeat resolution

# de Bruijn graph

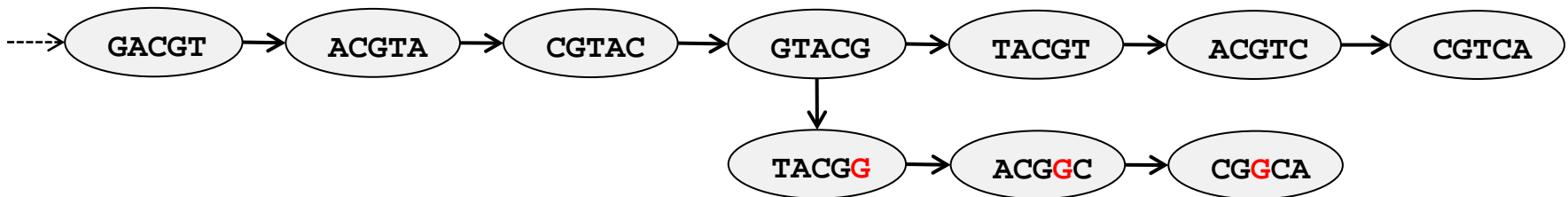
What if we have sequencing errors?

...GACGTACGTCA...

GACGTACG

CGTACGTC

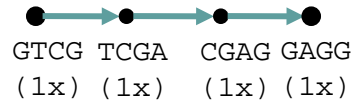
GTACG**G**CA



→ Additional nodes

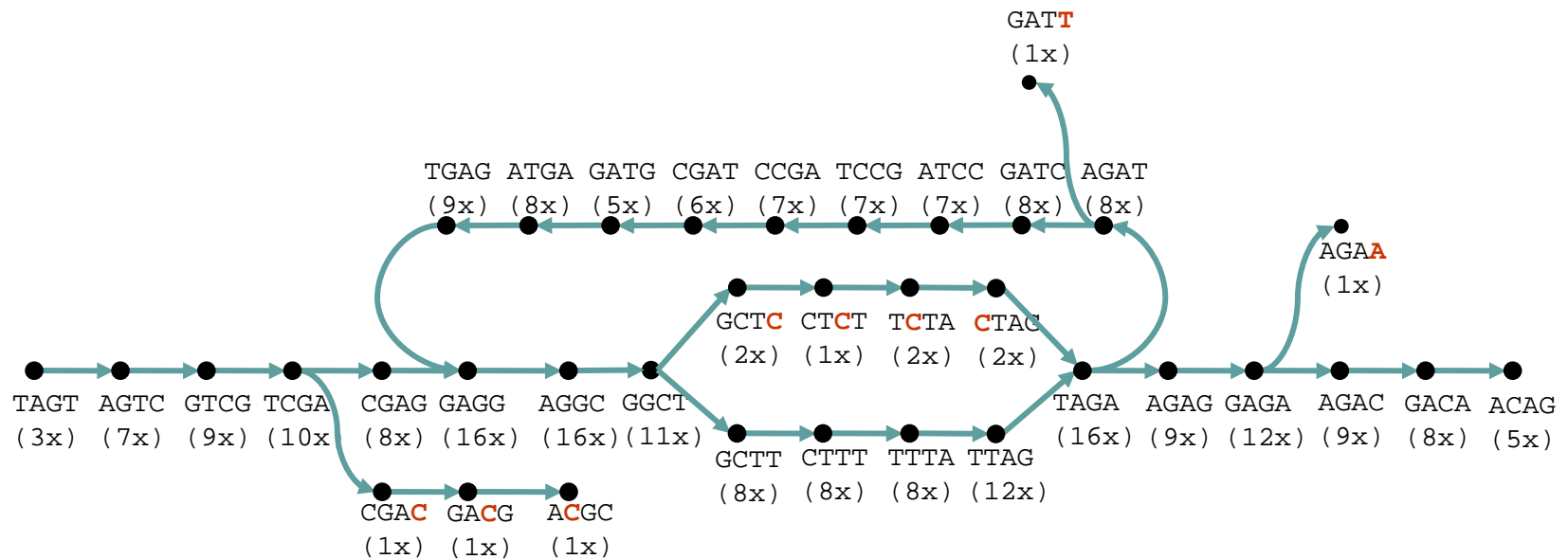
# A quick example

One read: GTCGAGG



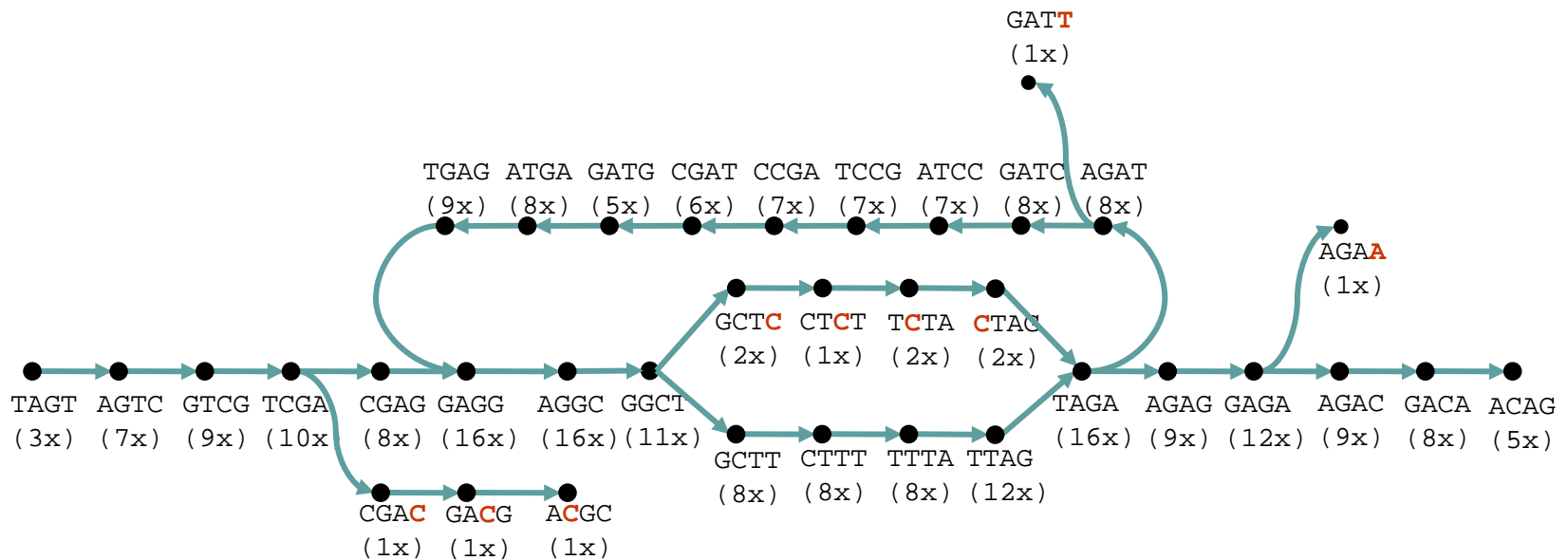
# A quick example

All the others...



# A quick example

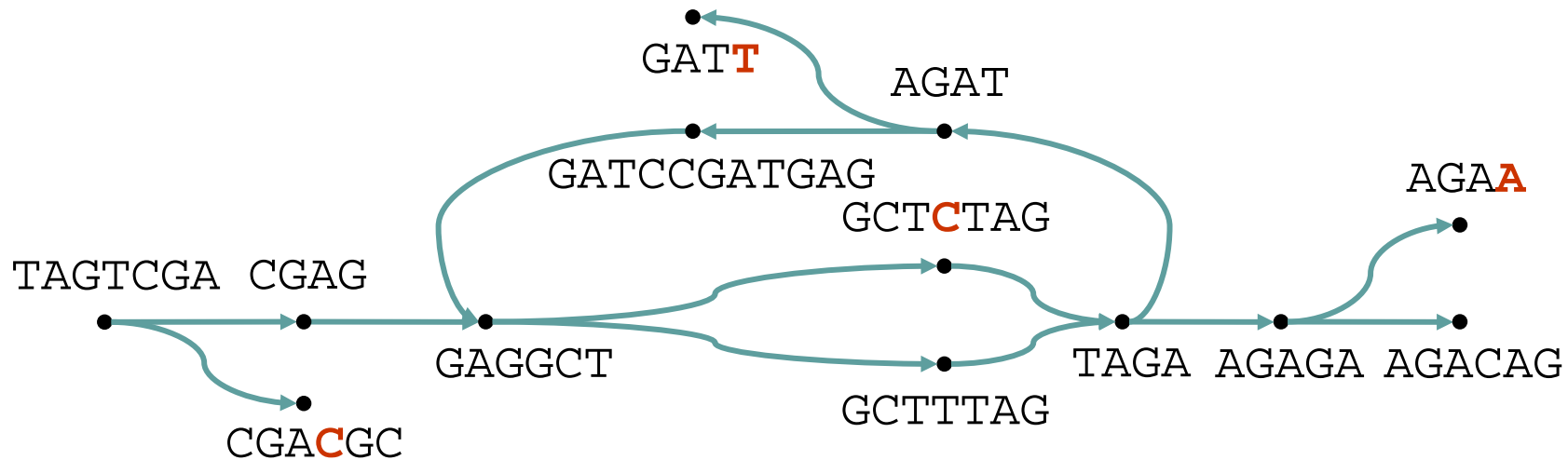
All the others...





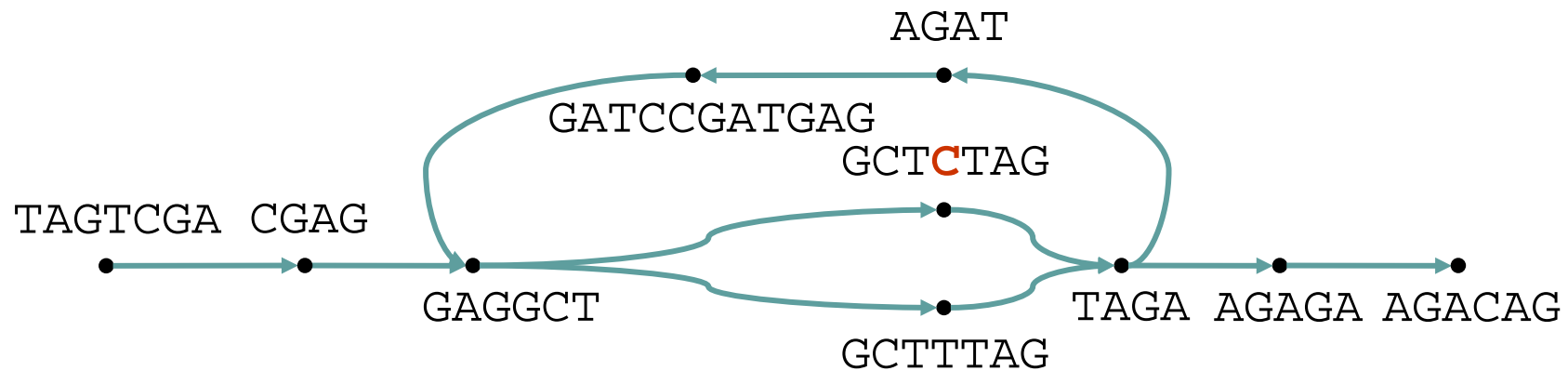
# A quick example

After simplification...



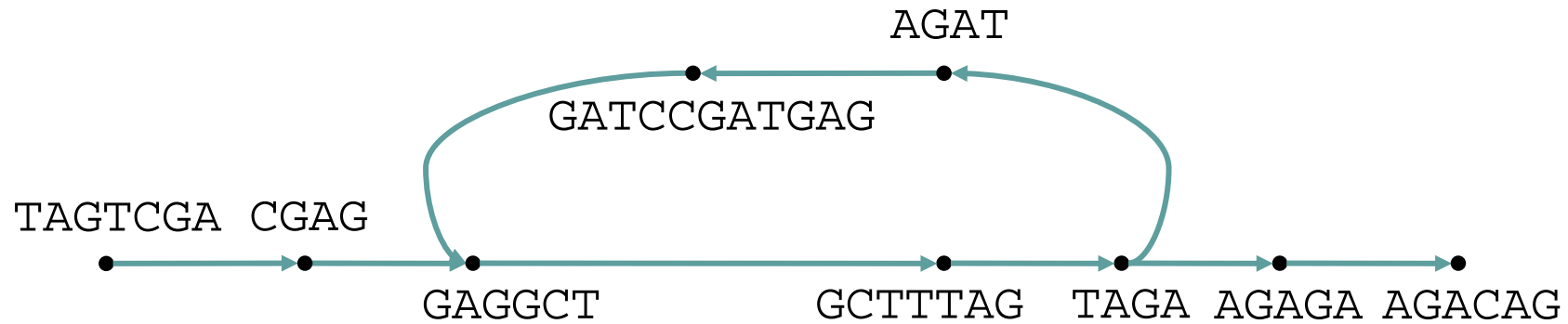
# Error removal

Tips removed...



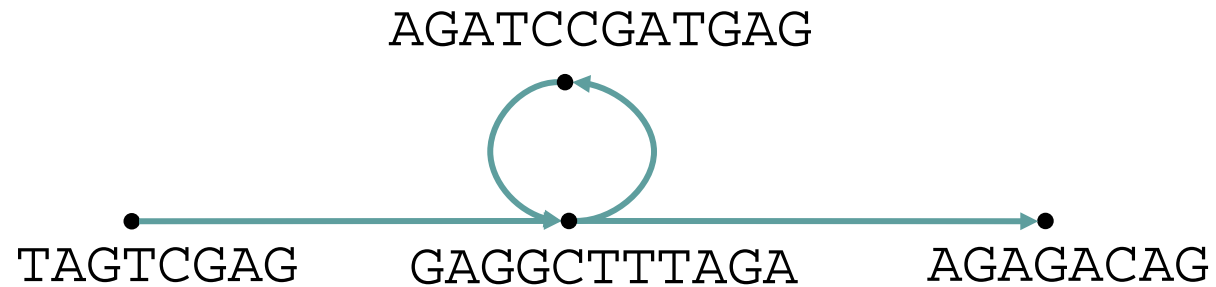
# Error removal

Bubbles removed ...



# Error removal

Final simplification...



# de Bruijn graph

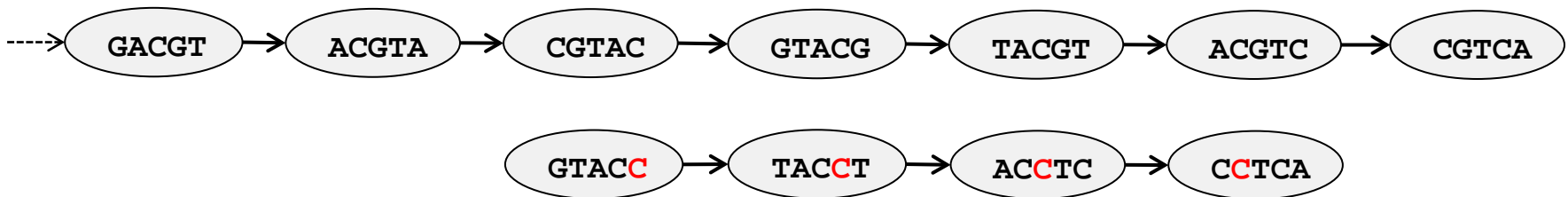
What if we have sequencing errors?

...GACGTACGTCA...

GACGTACG

CGTACGTC

GTACCTCA



→ Additional nodes  
& graph disconnected

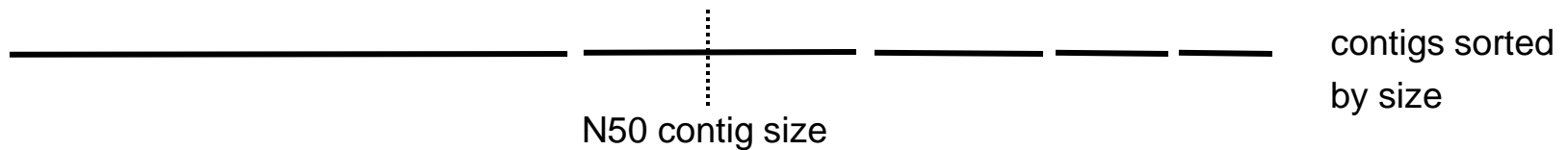
### Choice of $k$ :

Tradeoff between error tolerance  
and repeat resolution

# What is a good assembly?

Some assembly evaluation metrics:

- High N50 contig size (most commonly used metric)



- Low number of assembly errors (sequence errors, structural misjoins)



Only if a reference sequence is known!

# Conclusions

- Most assemblers use either the OLC or de Bruijn graph paradigm, both lead to NP-hard assembly models.
- However, assembler performance is independent of the underlying paradigm and mainly depends on heuristics for repeat resolution and handling noise.
- How to measure assembly accuracy is another important aspect, in general tradeoff between assembly contiguity and correctness.
- Evaluations show that assembly is far from solved, assembler performance still quite inconsistent.

*“For large genomes, the choice of assemblers is often limited to those that will run without crashing” (GAGE paper, 2012)*

# References

- GAGE: a critical evaluation of genome assemblies and assembly algorithms. Steven L. Salzberg et al., Genome Res. 2012
- Assemblathon 2: evaluating de-novo of genome assembly in three vertebrate species, Bradnam et al., not yet published
- Assembly algorithms for next-generation sequencing data. Jason R. Miller, Sergey Koren, Granger Sutton, Genomics, 2010.
- Fragment assembly string graph, Myers, 2005

## Figures:

- [geneed.nlm.nih.gov](http://geneed.nlm.nih.gov)
- <http://paper-shredding-services-review.toptenreviews.com/>
- [www.illumina.com](http://www.illumina.com)
- A Practical Comparison of *De Novo* Genome Assembly Software Tools for Next-Generation Sequencing Technologies, Zhang et al., PLOS one, 2011