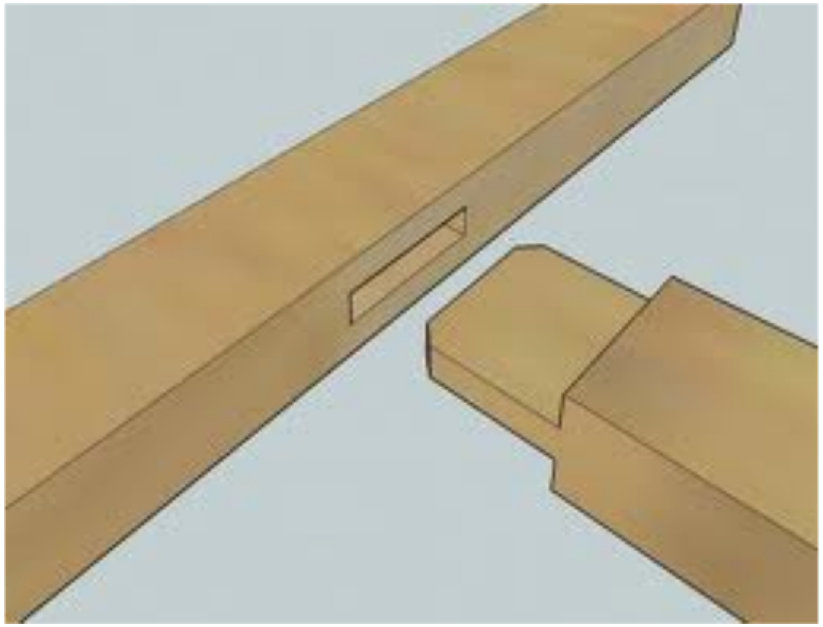
25/09/2019

RNA Seq analysis

de-novo transcriptome assembly

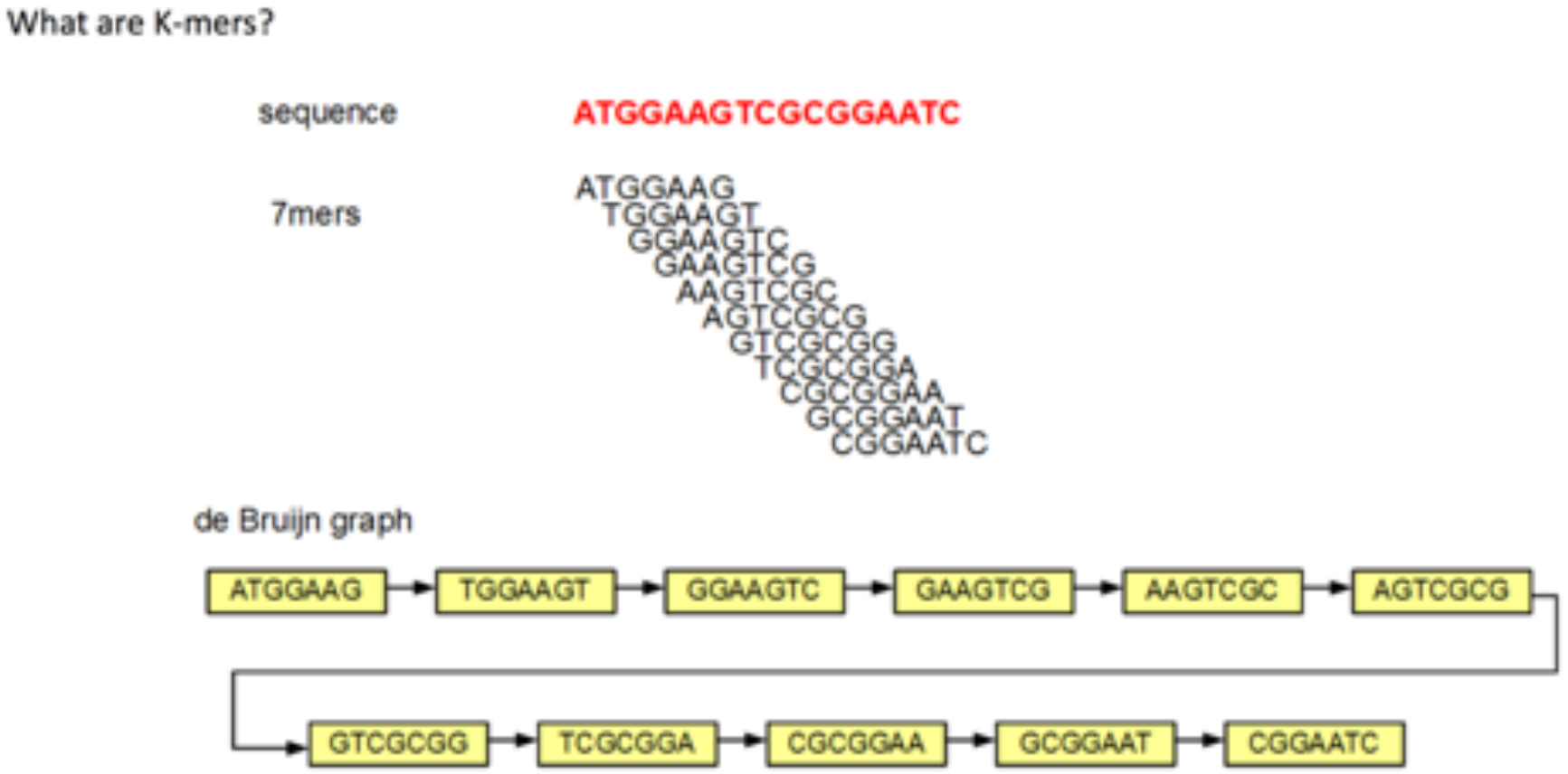
ABiMS – Station Biologique Roscoff



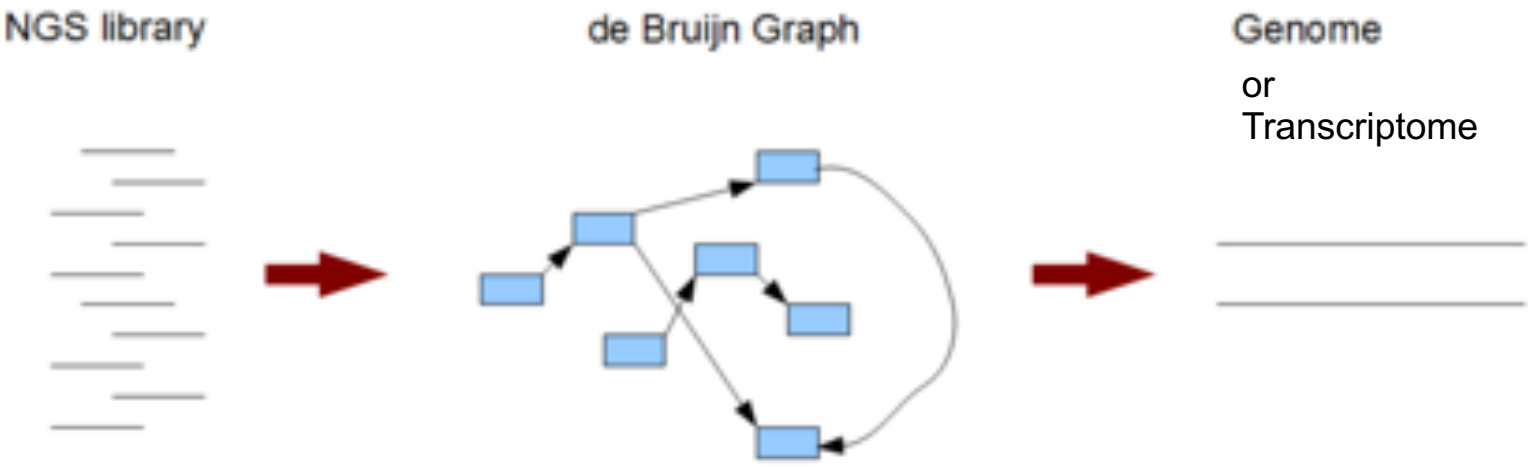
ASSEMBLY ALGORITHMS

De novo transcriptome assembly

Most of the modern assemblers works in K-mer space



De novo transcriptome assembly



Complications:

- Sequencing errors add nodes (and complexity)
- Short or long K-mers?
- No positional info on repeats (less important for transcriptomes)

From: http://www.homolog.us/Tutorials/index.php

Assembly algorithms

- Data model
  - Overlap-Layout-Consensus (OLC)
  - Eulerian / de Bruijn Graph (DBG)
- Search method
  - Greedy
  - Non-greedy
- Parallelizability
  - Multithreaded
  - Distributable

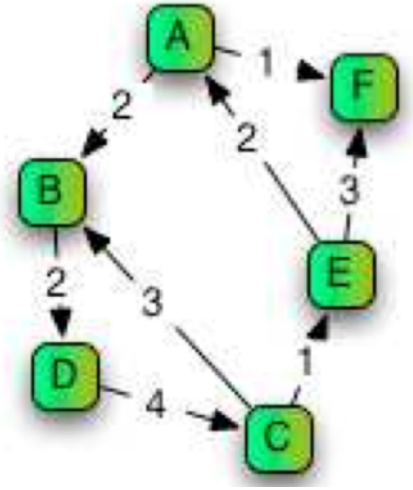
What is a “k-mer” ?

- A k-mer is a sub-string of length k
- A string of length L has (L-k+1) k-mers
- Example read L=8 has 5 k-mers when k=4

```
AGATCCGT
AGAT
  GATC
    ATCC
      TCCG
        CCGT
```

What is a graph ?

- Not an Excel chart!
- Nodes/Vertices
  - A,B,E,G,H,K,M
- Edges/Arcs
  - (lines between nodes)
- Directed graph
  - Arrow head on edge
- Weighted graph
  - Numerals on edges



Overlap - Layout – Consensus

- Overlap
  - All against all pair-wise comparison
  - Build graph: nodes=reads, edges=overlaps
- Layout
  - Analyse/simplify/clean the overlap graph
  - Determine Hamiltonian path (NP-hard)
- Consensus
  - Align reads along assembly path
  - Call bases using weighted voting

OLC : Pairwise Overlap

- All against all pair-wise comparison
  - 1⁄2 N(N-1) alignments to perform [N=no. reads]
- In practice, use smarter heuristics
  - Index all k-mers from all reads
  - Only check pairs that share enough k-mers
  - Similar approach to BLAST algorithm
- Both approaches parallelizable
  - Each comparison is independent

OLC: Overlap Example

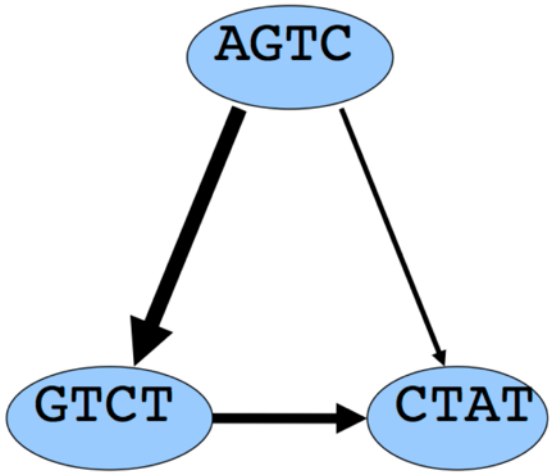
- True sequence (7bp) : AGTCTAT
- Reads (3 x 4bp) : AGTC, GTCT, CTAT
- Pairs to align (3)
  AGTC+GTCT , AGTC+CTAT , GTCT+CTAT
- Best overlaps

```
AGTC-        AGTC---        GTCT—
-GTCT        ---CTAT        --CTAT
```

(good) *(poor)* *(ok)*

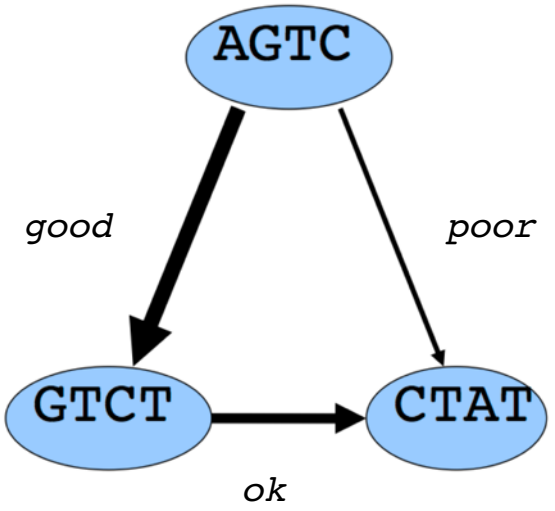
OLC: Overlap Graph

- Nodes are the 3 reads sequences
- Edges are the overlap alignment with orientation



OLC: Overlap Graph

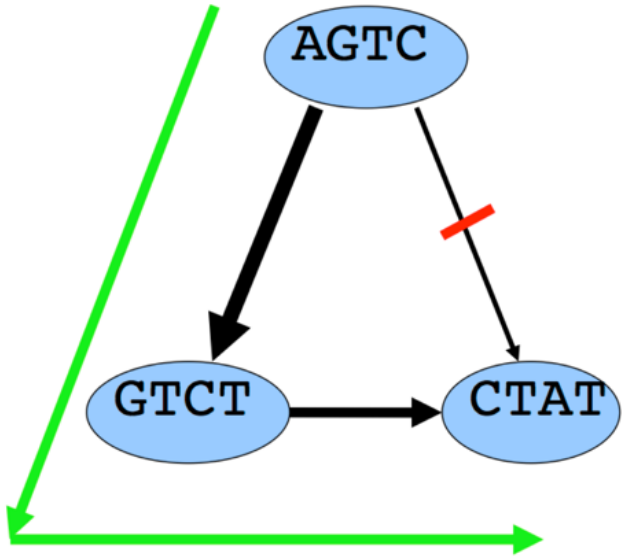
- Nodes are the 3 reads sequences
- Edges are the overlap alignment with orientation
- Edge thickness represents score of overlap




```
AGTC-      AGTC---      GTCT—
-GTCT      ---CTAT      --CTAT
(good)     (poor)        (ok)
```

OLC: Layout – Consensus

- Optimal path shown in green
- Un-traversed weak overlap in red
- Consensus is read by outputting the overlapped nodes along the path
- `aGTCTCTat`



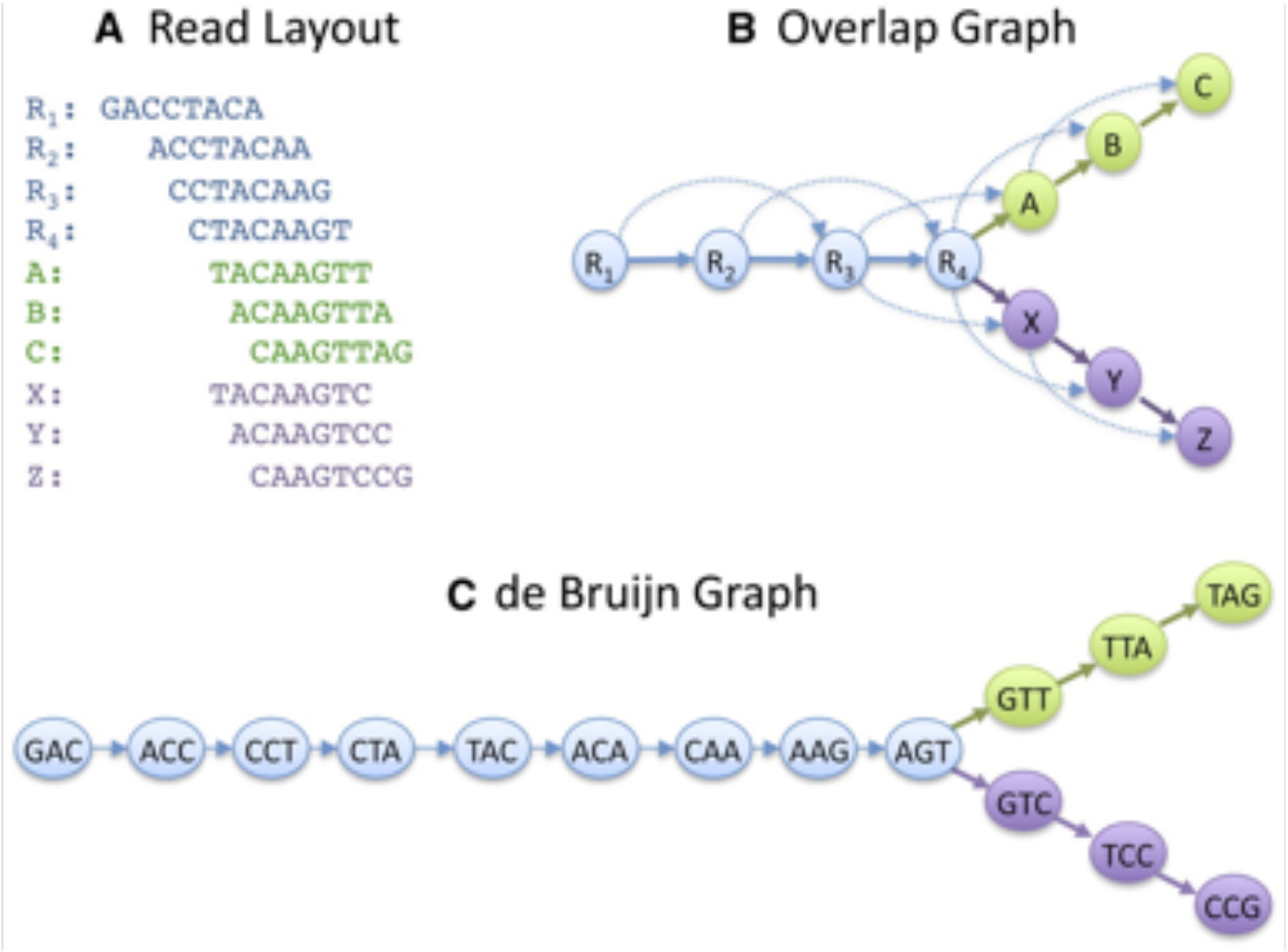
OLC : Softwares

- Phrap, PCAP, CAP3
  - Smaller scale assemblers
- Celera Assembler
  - Sanger-era assembler for large genomes
- Arachne, Edena, CABOG, Mira 4
  - Modern Sanger/hybrid assemblers
- Newbler (gsAssembler)
  - Used for 454 NGS “long” reads
  - Can be used for IonTorrent flowgrams too

- More recently : TraRECo: a greedy approach based de novo transcriptome assembler with read error correction using consensus matrix. Yoon BMC Genomics. 2018; 19: 653. (doi: 10.1186/s12864-018-5034-x)

Eulerian approach

- Break all reads (length L) into (L-k+1) k-mers
  - L=36, k=31 gives 6 k-mers per read
- Construct a *de Bruijn* graph (DBG)
  - Nodes = one for each unique k-mer
  - Edges = k-1 exact overlap between two nodes
- Graph simplification
  - Merge chains, remove bubbles and tips
- Find a Eulerian path through the graph
  - Linear time algorithm, unlike Hamiltonian


A Read Layout
R1: GACCTACA
R2: ACCTACAA
R3: CCTACAAG
R4: CTACAAGT
A: TACAAGTT
B: ACAAGTTA
C: CAAGTTAG
X: TACAAGTC
Y: ACAAGTCC
Z: CAAGTCCG
B Overlap Graph
R1
R2
R3
R4
A
B
C
X
Y
Z
C de Bruijn Graph
GAC
ACC
CCT
CTA
TAC
ACA
CAA
AAG
AGT
GTT
TTA
TAG
GTC
TCC
CCG

DBG : simple

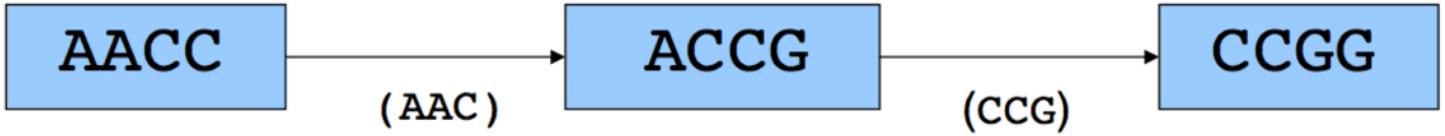
- Sequence

  `AACCGG`

- K-mers (k=4)

  `AACC  ACCG  CCGG`

- Graph



DBG : repeated k-mer

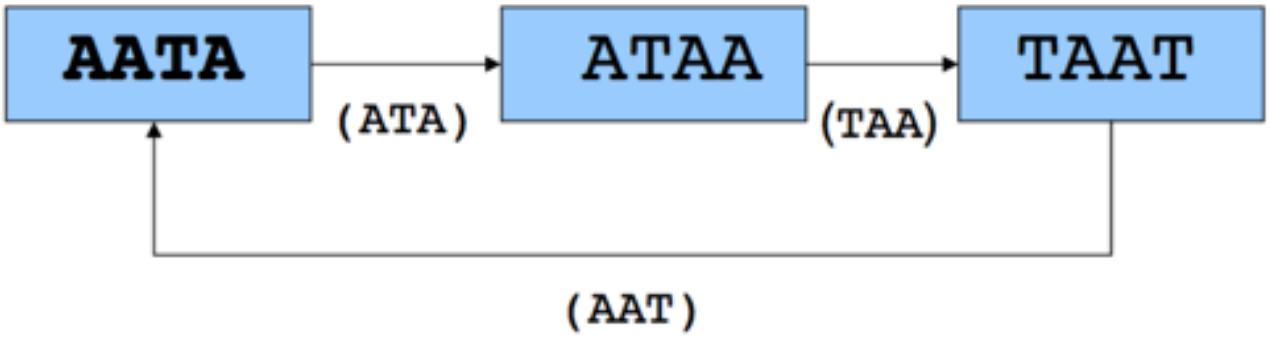
- Sequence

  `AATAATA`

- K-mers (k=4)

  `AATA ATAA TAAT AATA (repeat)`

- Graph



DBG: alternate paths

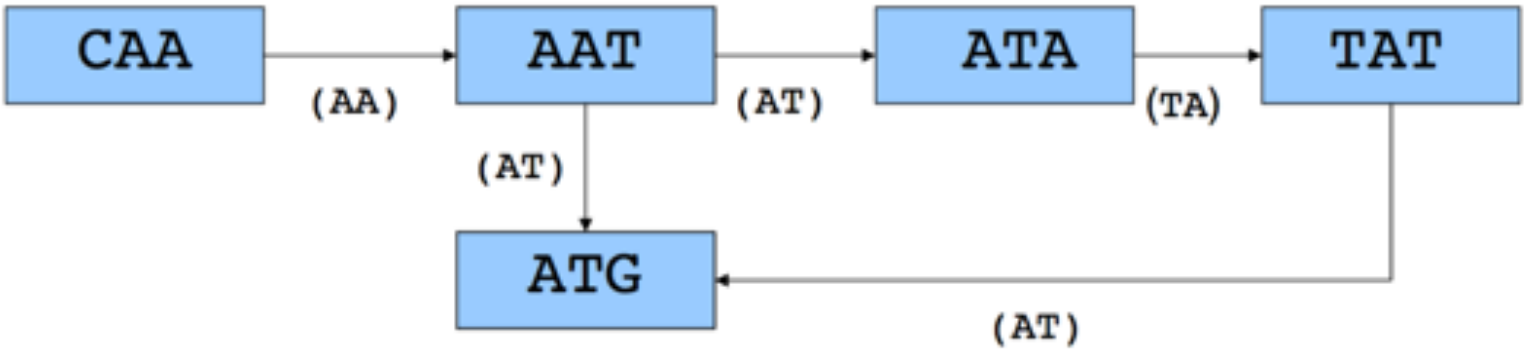
- Sequence

  `CAATATG`

- K-mers (k=3)

  `CAA AAT ATA TAT ATG`

- Graph



DBG: graph simplification

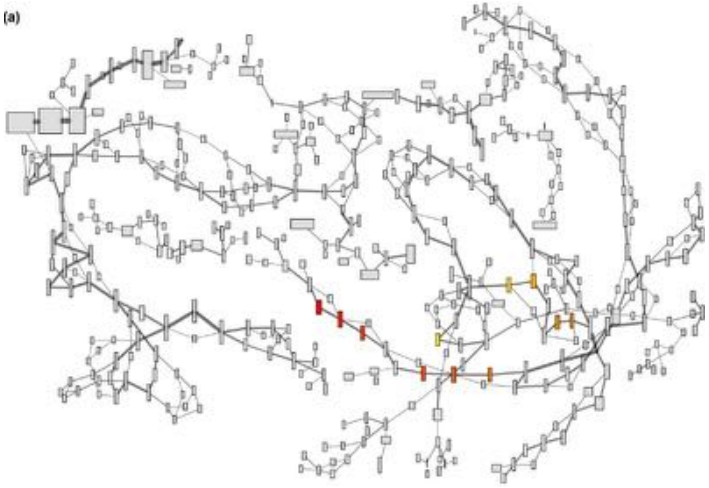
- This problem is known to be NP-complete
- In practice, heuristics are used which consist in simplifying the graph to « make it linear »
- However, the structures that are removed may correspond to relevant biological structures (SNPs, alternative splicing)

DBG: graph simplification

- Remove tips or spurs
  - Dead ends in graph due to errors at read end
- Collapse bubbles
  - Errors in middle of reads
  - But could be true SNPs or diploidity
- Remove low coverage paths
  - Possible contamination
- Makes final Eulerian path easier
  - And hopefully more accurate contigs

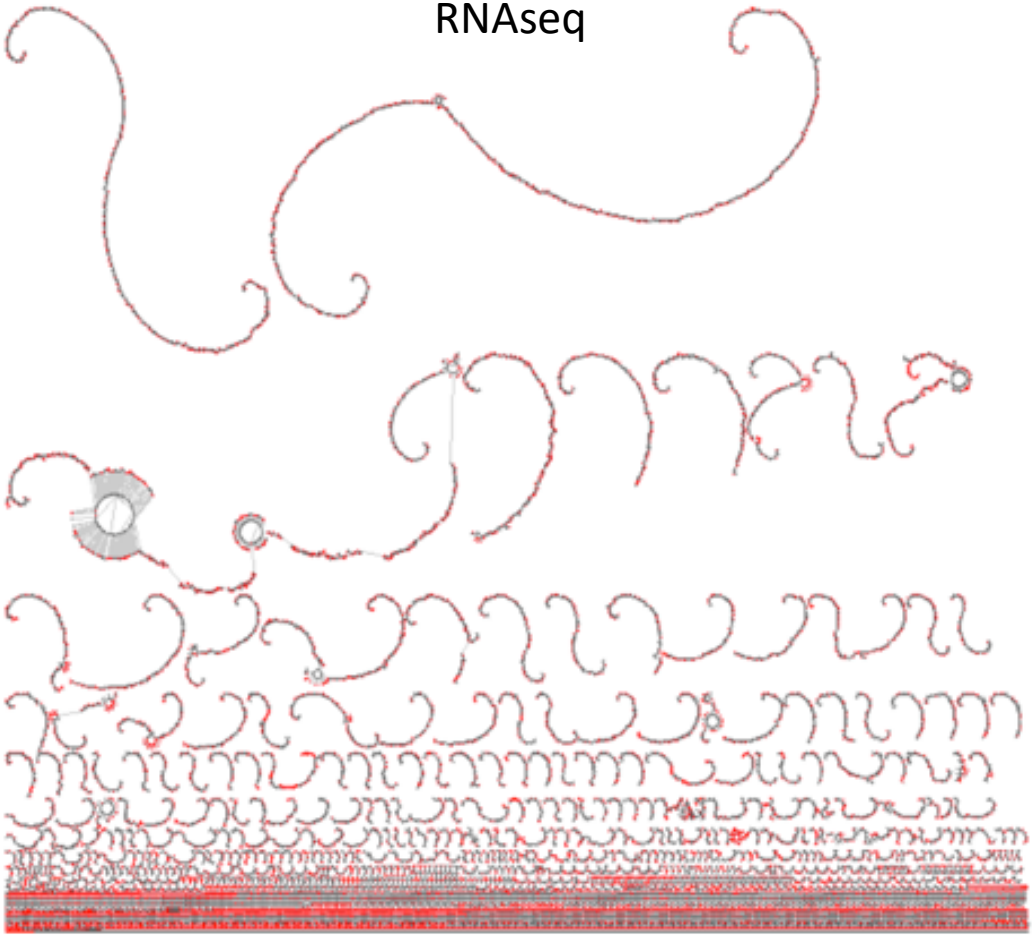
Example of DBG built from Genome data - RNA-seq data

Genome



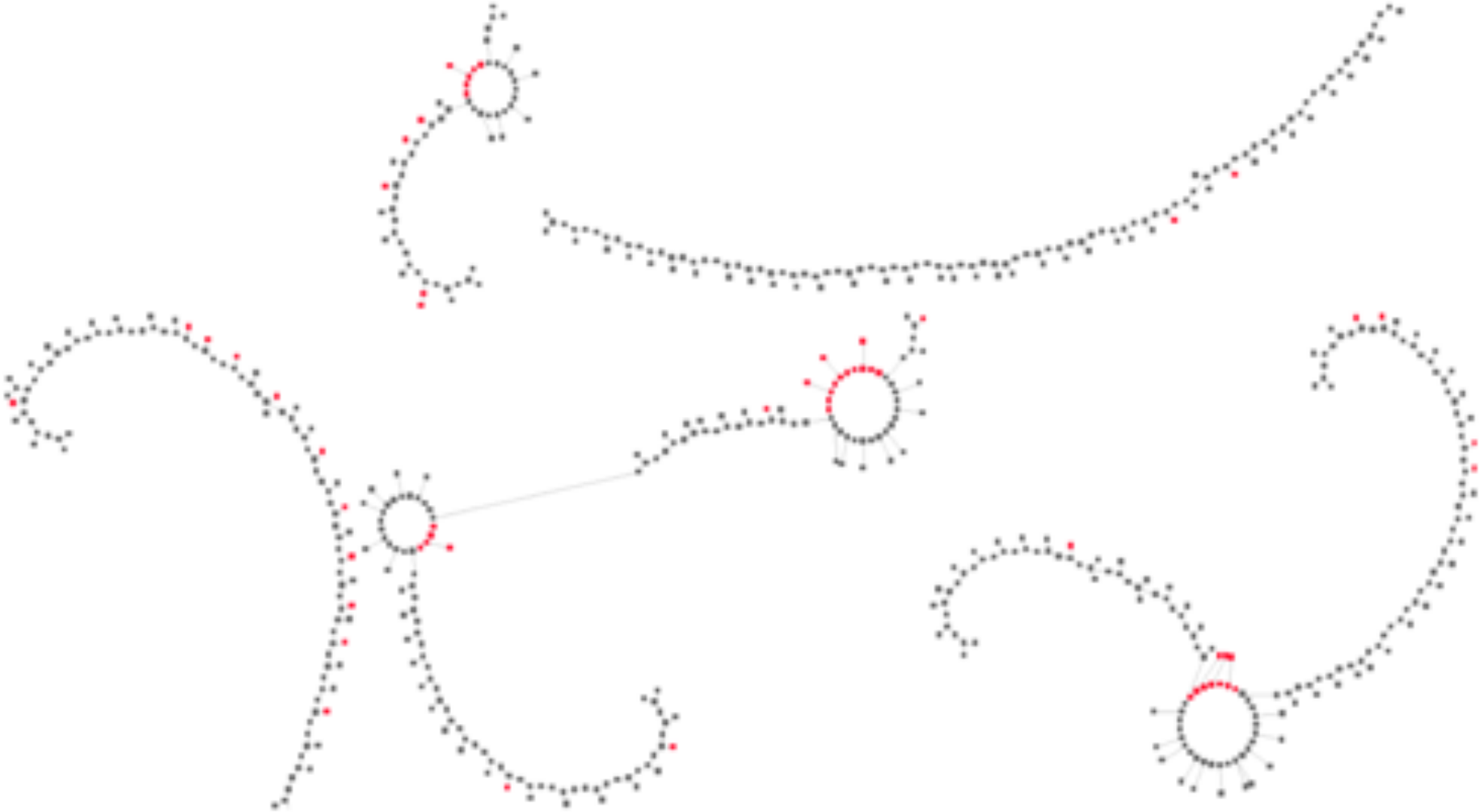
Entire chromosomes represented

RNAseq



Ideally, one graph per expressed gene

Polymorphism in RNA-seq data



DBG: softwares

- **Velvet/Oases**
  - Velvet (Zerbino, Birney 2008) is a sophisticated set of algorithms that constructs de Bruijn graphs, simplifies the graphs, and corrects the graphs for errors and repeats.
  - Oases (Schulz et al. 2012) post-processes Velvet assemblies (minus the repeat correction) with different k-mer sizes.
- **Trans-ABySS**
  - Trans-ABySS (Robertson et al. 2010) takes multiple ABySS assemblies (Simpson et al. 2009)
- **CLC bio Genomics Workstation**
- **Trinity**

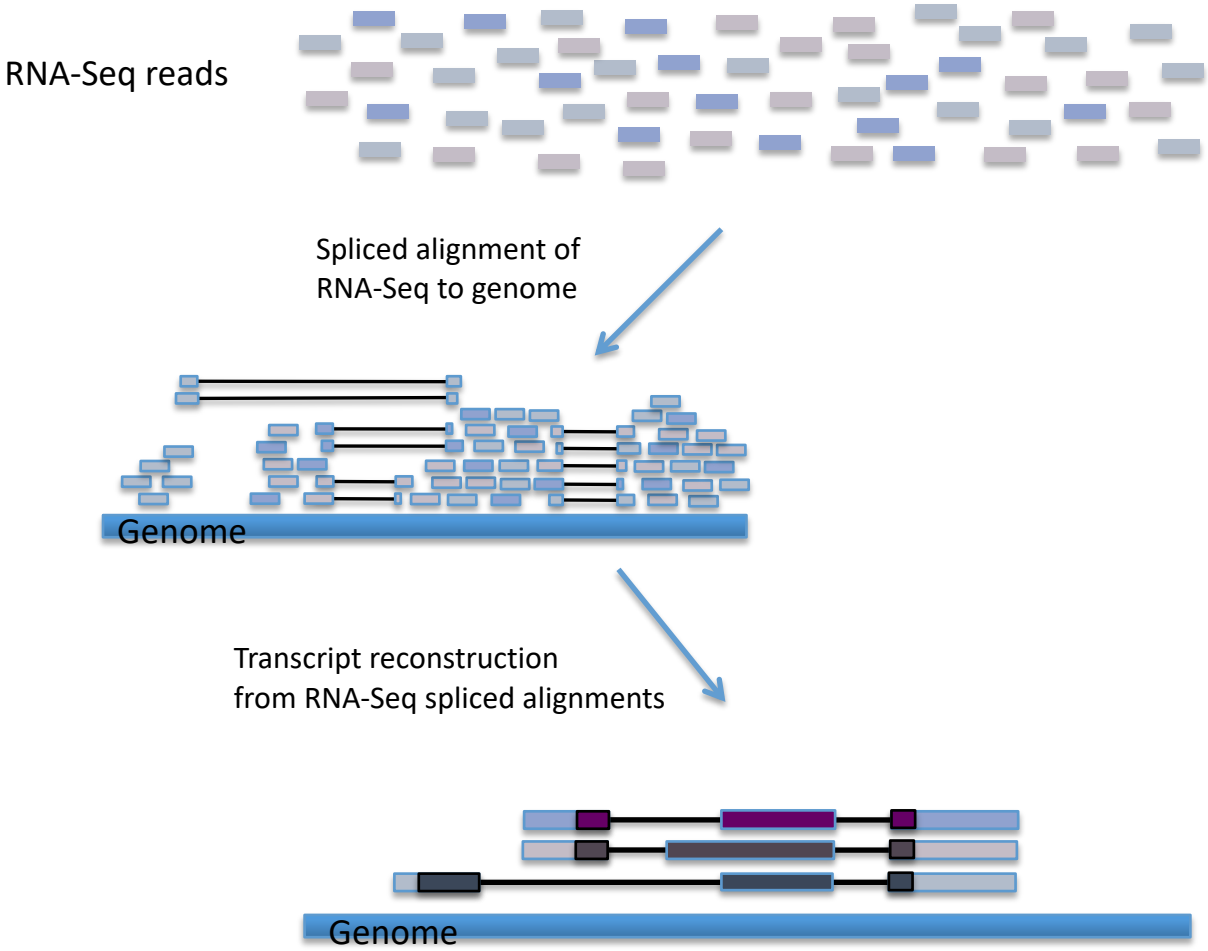
OLC vs DBG

- DBG
  - More sensitive to repeats and read errors
  - Graph converges at repeats of length k
  - One read error introduces k false nodes
  - Parameters: kmer_size cov_cutoff ...

!!Underline the importance of reads errors correction!!

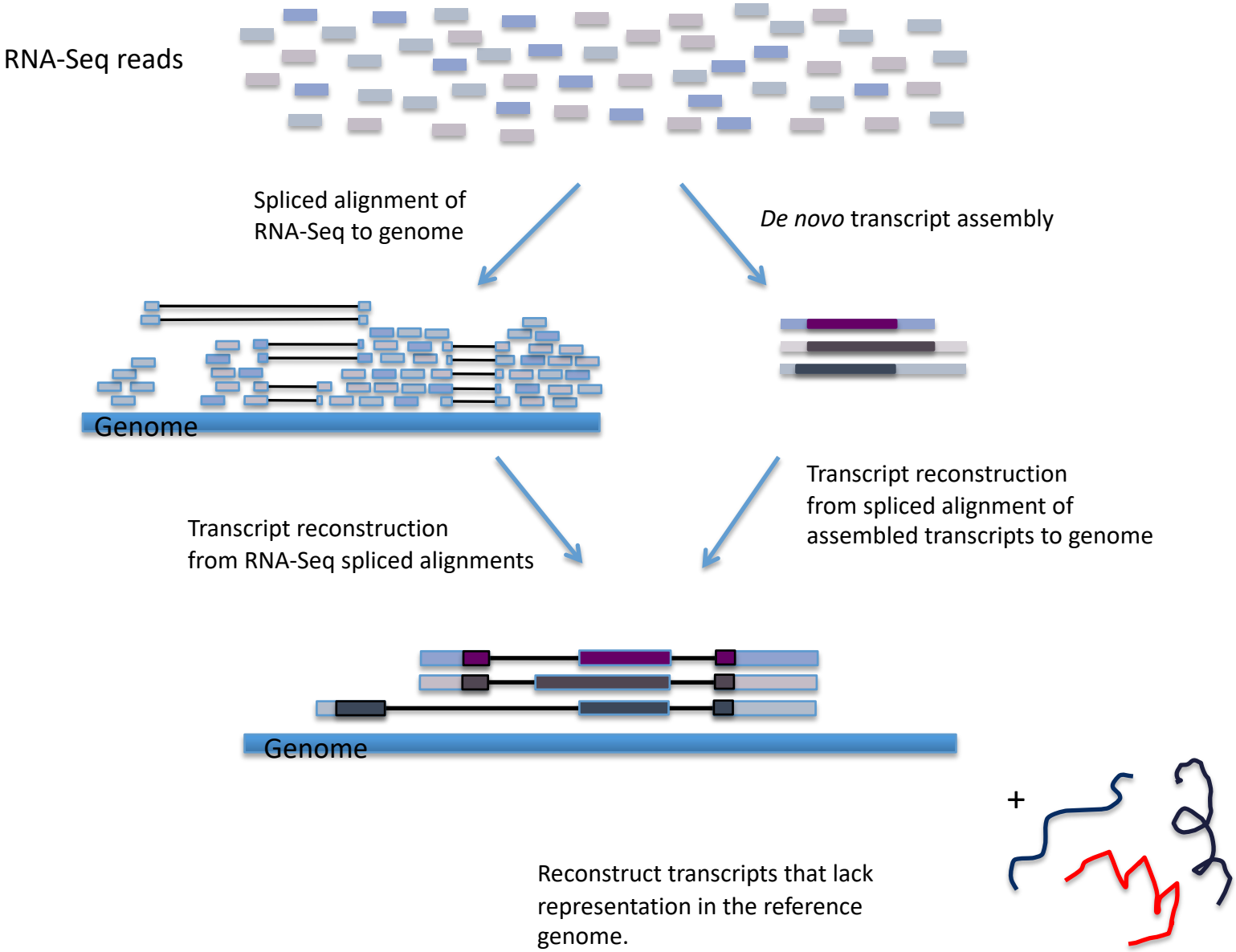
- OLC
  - Less sensitive to repeats and read errors
  - Graph construction more demanding
  - Doesn't scale to voluminous short reads
  - Parameters: minOverlapLen %id ...
  - OLC assembly is best suited to lower coverage, longer read data such as Sanger, 454, or PacBio.

Contemporary strategies for transcript reconstruction from RNA-Seq



The short read alignments, instead of the reads themselves, are assembled into gene structures

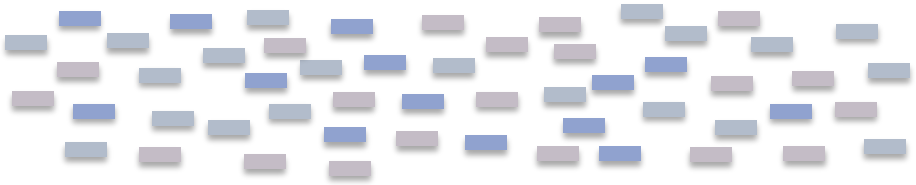
Contemporary strategies for transcript reconstruction from RNA-Seq



Reconstruct transcripts that lack representation in the reference genome.

Contemporary strategies for transcript reconstruction from RNA-Seq

RNA-Seq reads

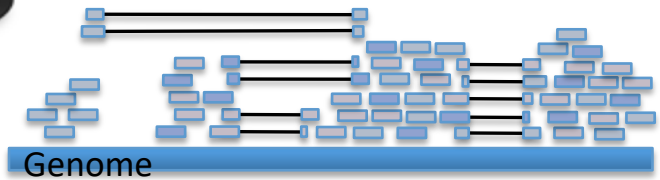


Oases
SOAPdenovo-trans
TransABYSS

Spliced alignment of RNA-Seq to genome

De novo transcript assembly

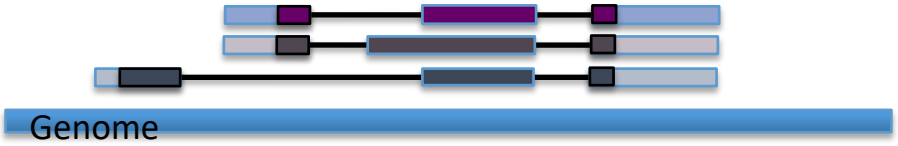
Tophat
STaR
HiSAT2



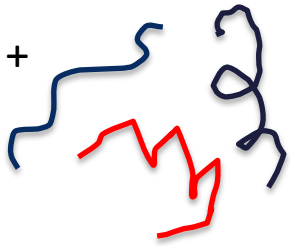
Transcript reconstruction from RNA-Seq spliced alignments

Transcript reconstruction from spliced alignment of assembled transcripts to genome

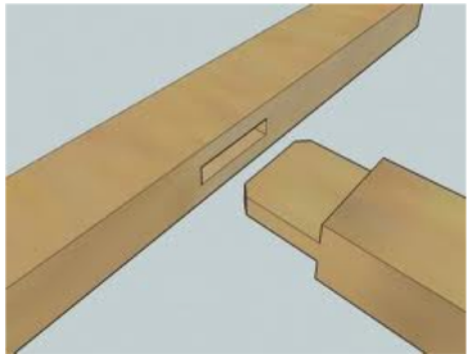
Cufflinks
Scripture



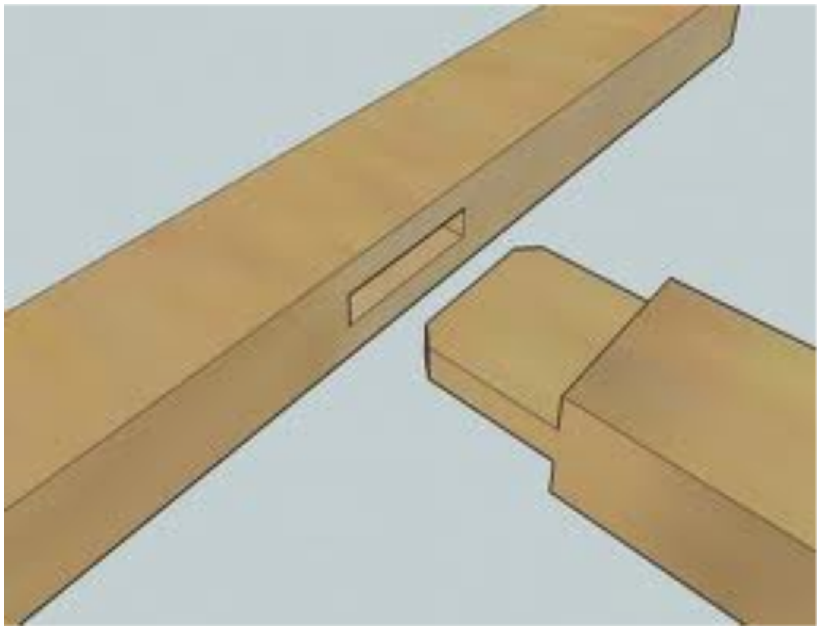
Gmap



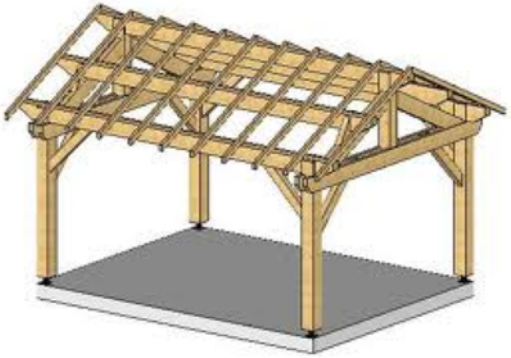
RNA-Seq analysis



De novo transcriptome assembly



?



Transcriptome assembly with Trinity

Brian Haas
Moran Yassour
Kerstin Lindblad-Toh
Aviv Regev
Nir Friedman
David Eccles
Alexie Papanicolaou
Michael Ott
…

developed at the Broad Institute

Additional tools, plug-ins, and documentation continually added to the Trinity Suite

Trinity workflow

- Compress data (inchworm):
  - Cut reads into k-mers (k consecutive nucleotides)
  - Overlap and extend (greedy)
  - Report all sequences (“contigs”)

- Build de Bruijn graph (chrysalis):
  - Collect all contigs that share k-1-mers
  - Build graph (disjoint “components”)
  - Map reads to components

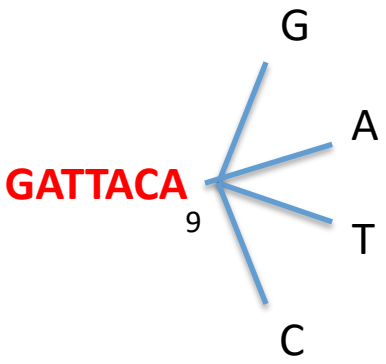
- Enumerate all consistent possibilities (butterfly):
  - Unwrap graph into linear sequences
  - Use reads and pairs to eliminate false sequences
  - Use dynamic programming to limit compute time (SNPs!!)

Inchworm Algorithm

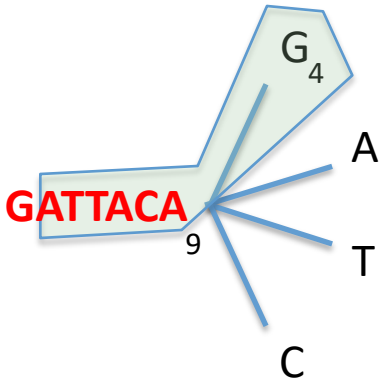
Decompose all reads into overlapping Kmers (25-mers) and count them : Jellyfish

Identify seed kmer as most abundant Kmer, ignoring low-complexity kmers.

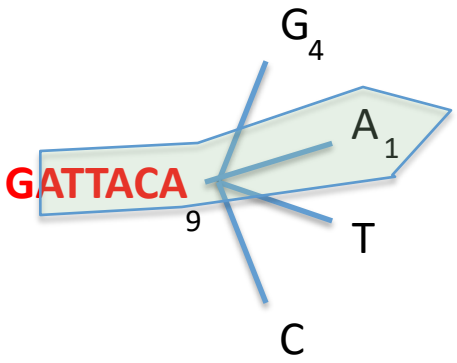
Extend kmer at 3’ end, guided by coverage.



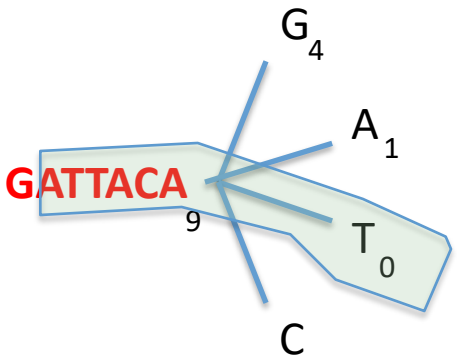
Inchworm Algorithm



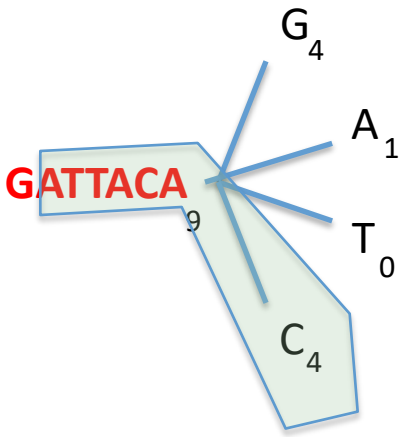
Inchworm Algorithm



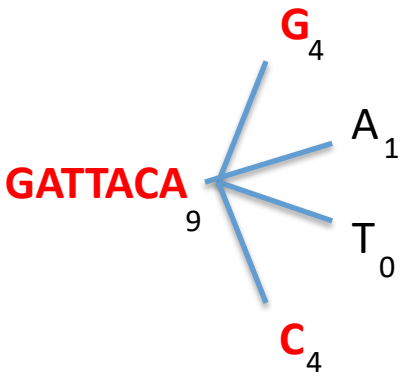
Inchworm Algorithm



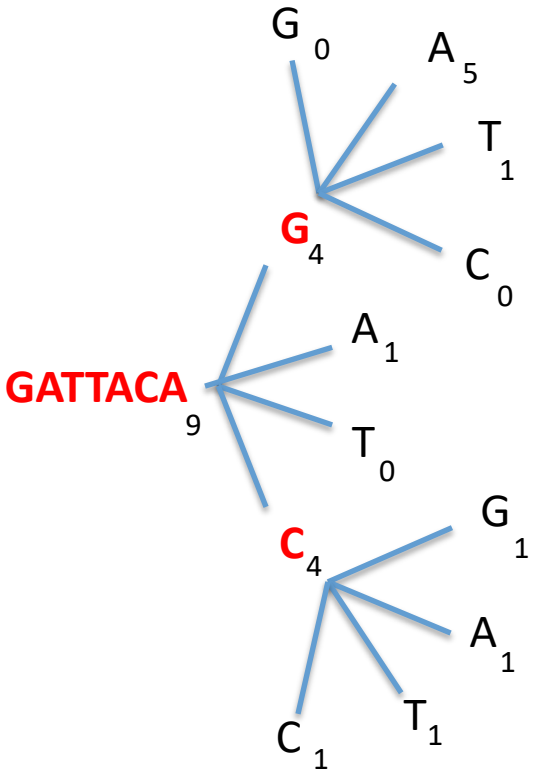
Inchworm Algorithm



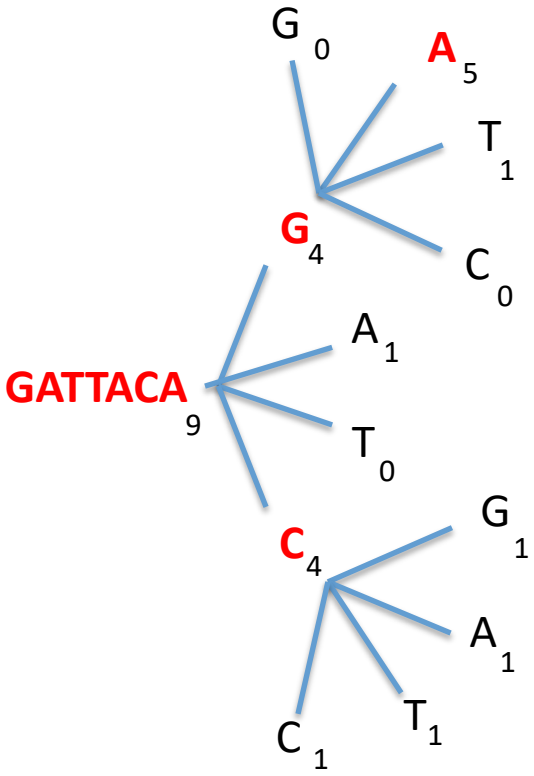
Inchworm Algorithm



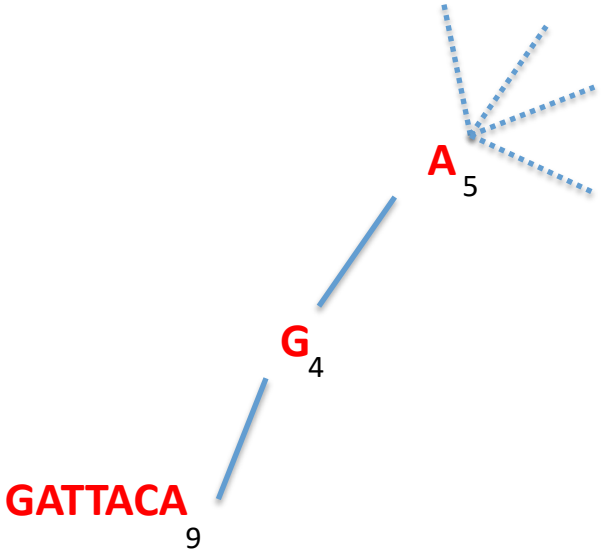
Inchworm Algorithm



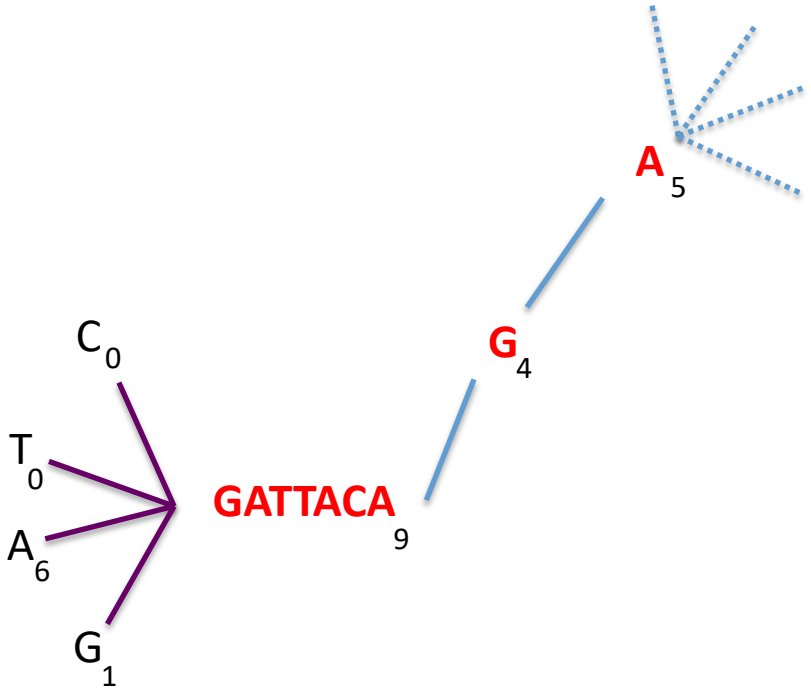
Inchworm Algorithm



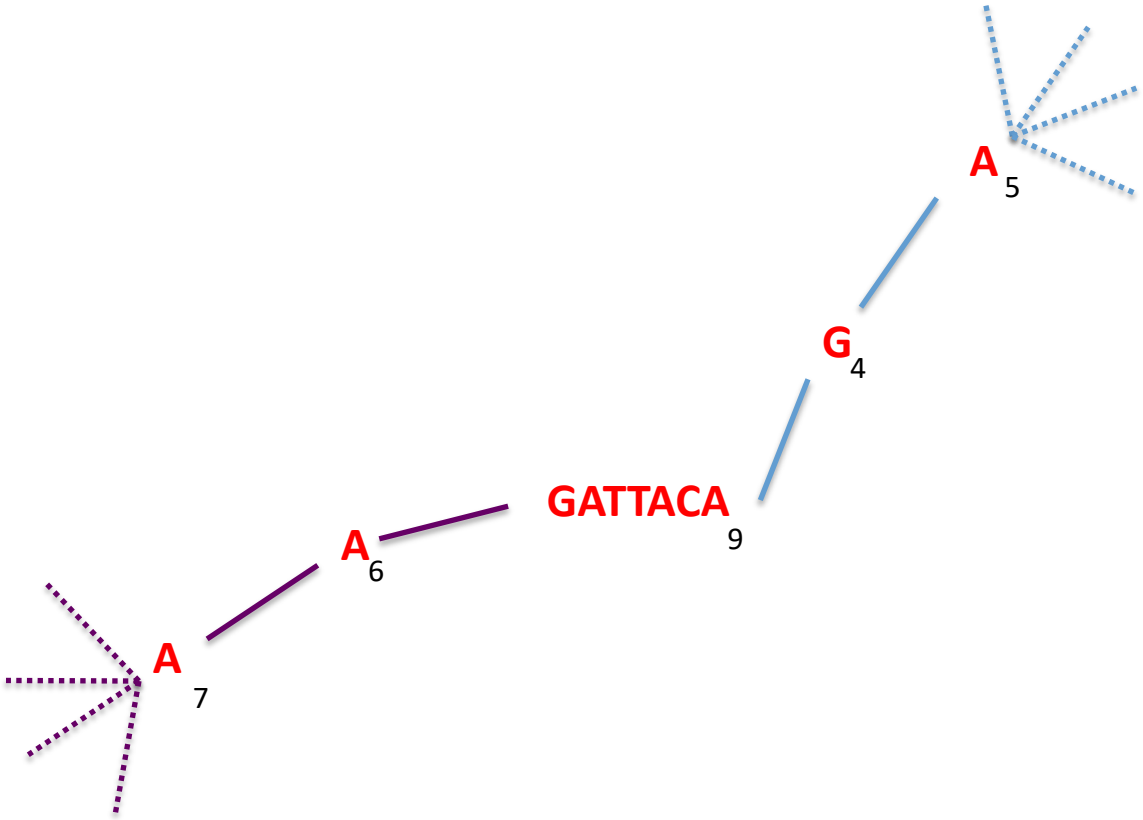
Inchworm Algorithm



Inchworm Algorithm



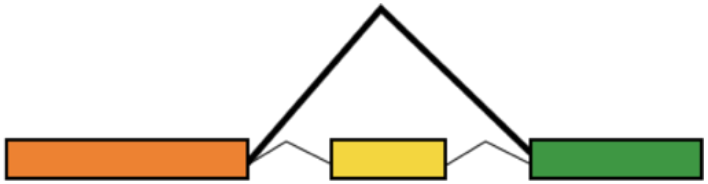
Inchworm Algorithm



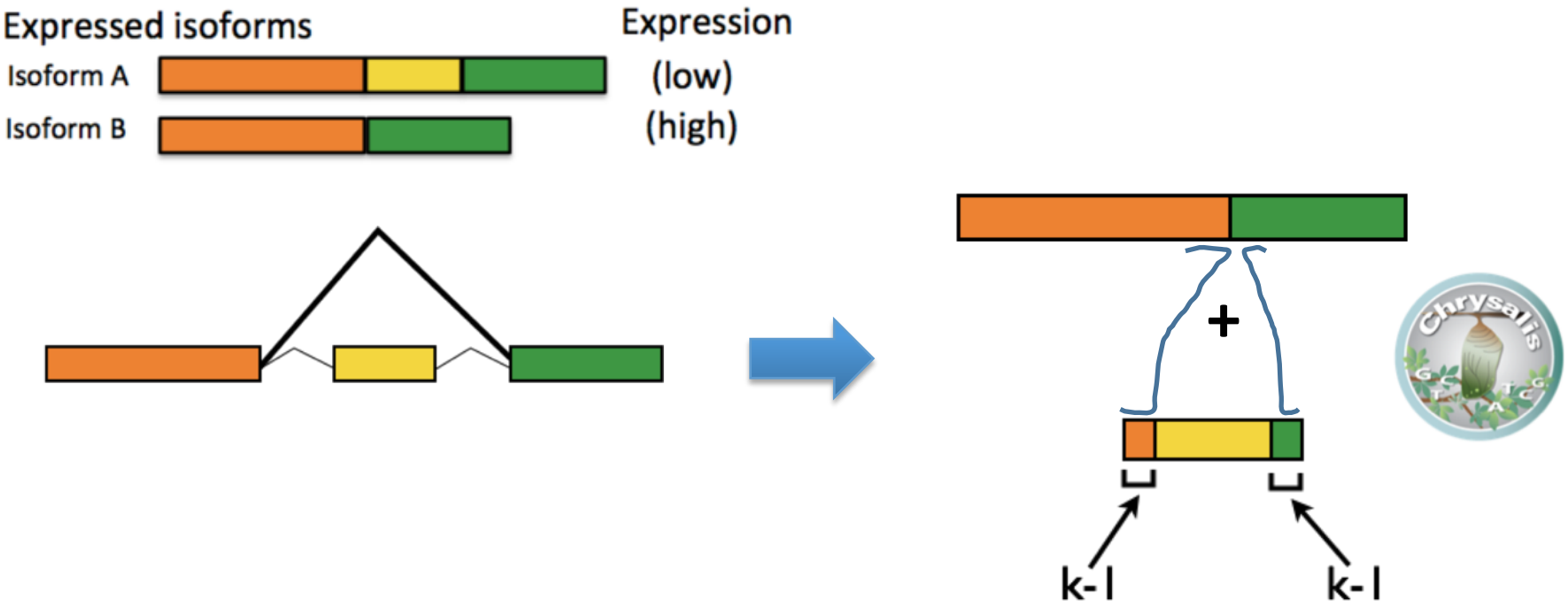
Report contig: ….AAGATTACAGA….

Remove assembled kmers from catalog, then repeat the entire process.

Inchworm Contigs from Alt-Spliced Transcripts



Inchworm Contigs from Alt-Spliced Transcripts



Inchworm can only report contigs derived from unique kmers.

Alternatively spliced transcripts :

- the more highly expressed transcript may be reported as a single contig,
- the parts that are different in the alternative isoform are reported separately.

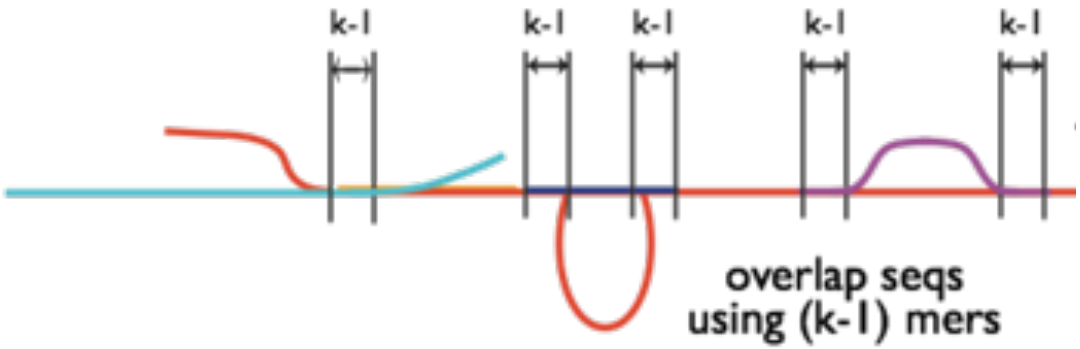
Chrysalis

Integrate (clustering)
Isoforms via k-1 overlaps

Chrysalis

>a121:len=5845
>a122:len=2560
>a123:len=4443
>a124:len=48
>a125:len=8876
>a126:len=66

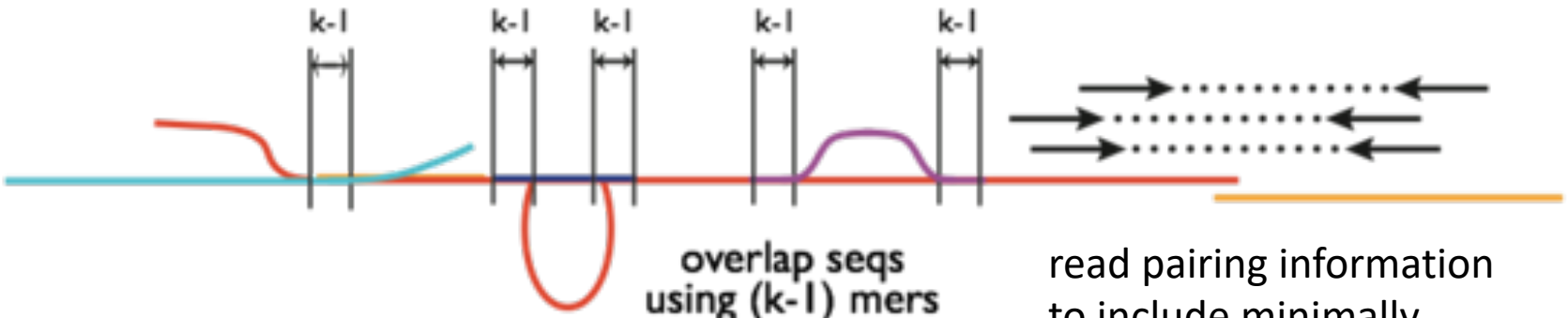
Integrate (clustering)
Isoforms via k-1 overlaps



Chrysalis

>a121:len=5845
>a122:len=2560
>a123:len=4443
>a124:len=48
>a125:len=8876
>a126:len=66

Integrate (clustering)
Isoforms via k-1 overlaps



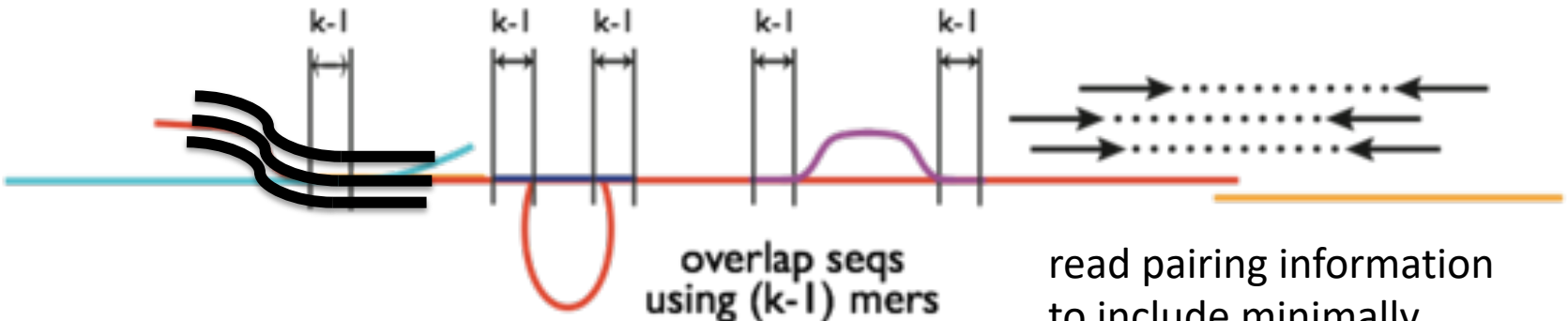
read pairing information
to include minimally
overlapping contigs

Chrysalis

>a121:len=5845
>a122:len=2560
>a123:len=4443
>a124:len=48
>a125:len=8876
>a126:len=66

Integrate (clustering)
Isoforms via k-1 overlaps

Verify via “welds”



read pairing information
to include minimally
overlapping contigs

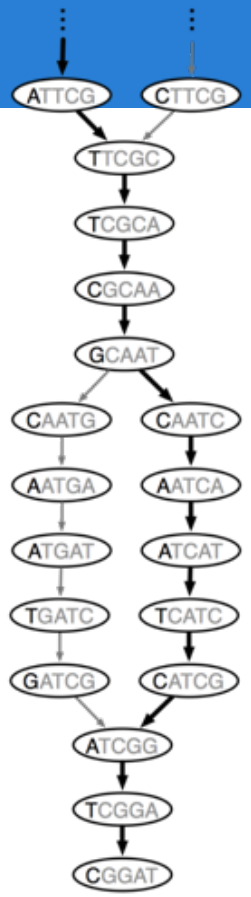
Chrysalis

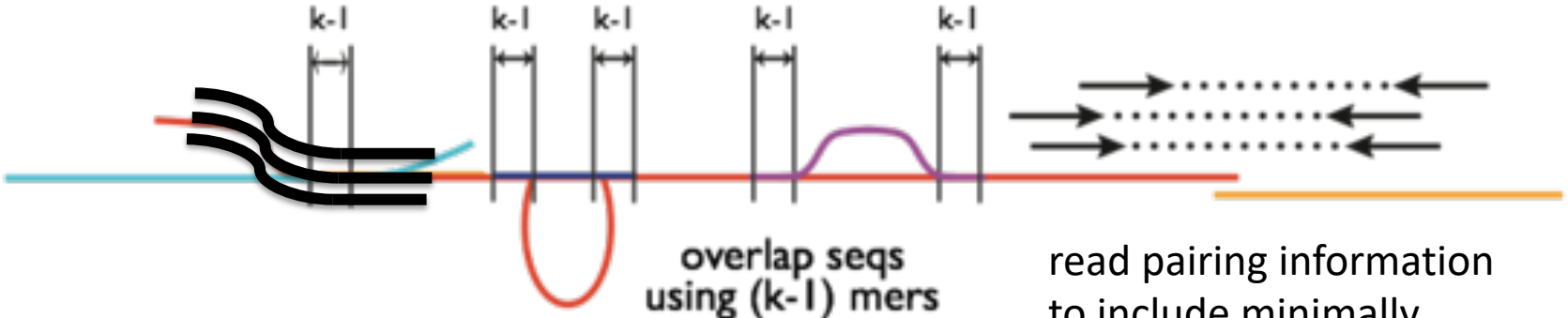
>a121:len=5845
>a122:len=2560
>a123:len=4443
>a124:len=48
>a125:len=8876
>a126:len=66

Integrate (clustering)
Isoforms via k-1 overlaps

Verify via “welds”

Build de Bruijn Graphs
(ideally, one per gene)

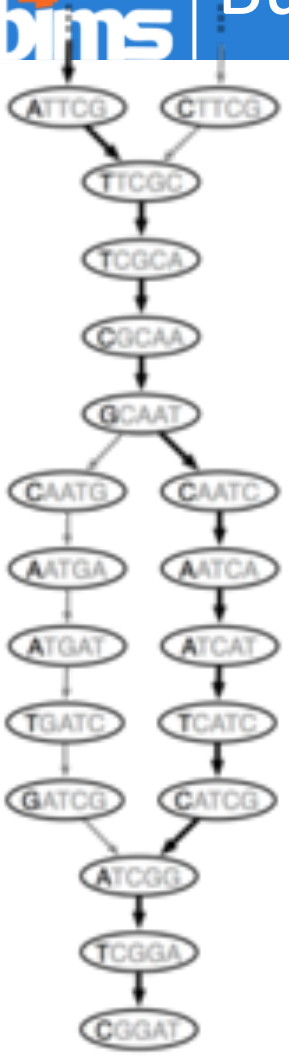




overlap seqs
using (k-1) mers

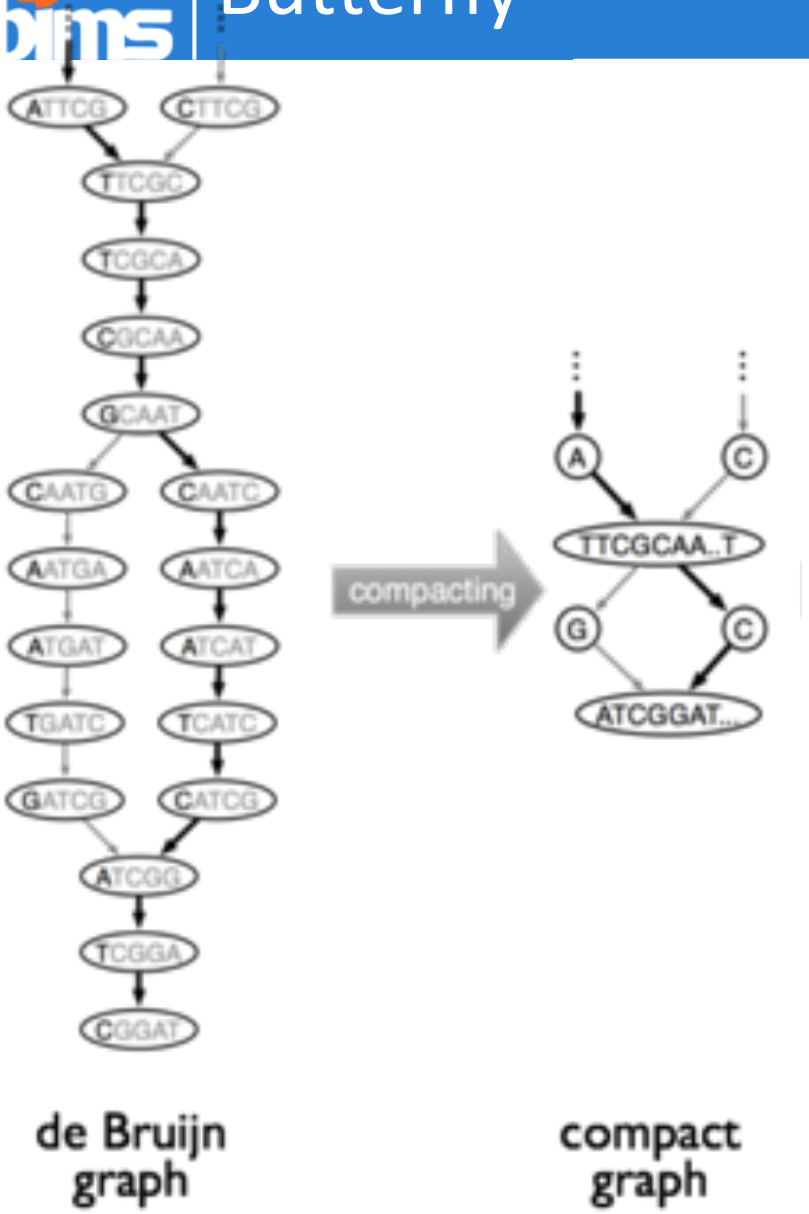
read pairing information
to include minimally
overlapping contigs

Butterfly

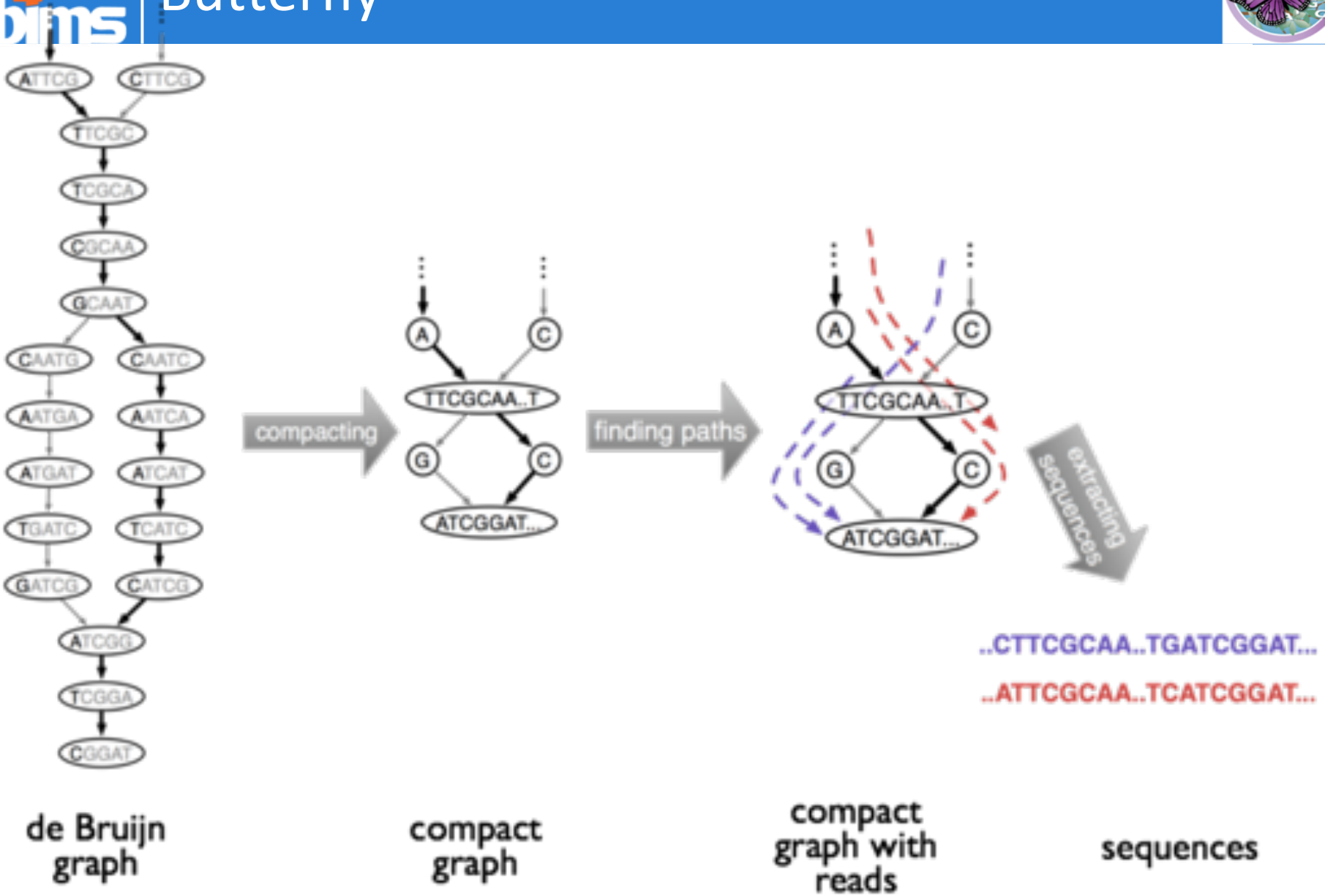


de Bruijn graph

Butterfly


ATTCG
CTTCG
TTCGC
TCGCA
CGCAA
GCAAT
CAATG
CAATC
AATGA
AATCA
ATGAT
ATCAT
TGATC
TCATC
GATCG
CATCG
ATCGG
TCGGA
CGGAT
compacting
A
C
TTCGCAA..T
G
C
ATCGGAT...
de Bruijn graph
compact graph

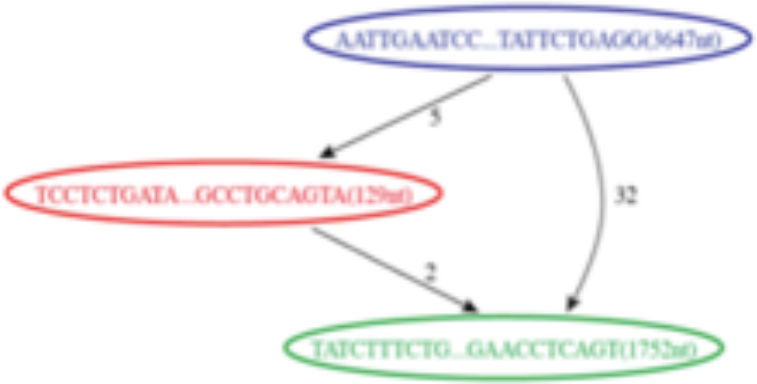
Butterfly


ATTCG
CTTCG
TTCGC
TCGCA
CGCAA
GCAAT
CAATG
CAATC
AATGA
AATCA
ATGAT
ATCAT
TGATC
TCATC
GATCG
CATCG
ATCGG
TCGGA
CGGAT
compacting
A
C
TTCGCAA..T
G
C
ATCGGAT...
finding paths
A
C
TTCGCAA..T
G
C
ATCGGAT...
extracting sequences
..CTTCGCAA..TGATCGGAT...
..ATTCGCAA..TCATCGGAT...
de Bruijn graph
compact graph
compact graph with reads
sequences

Alternatively spliced transcripts

Butterfly Example 1:
Reconstruction of Alternatively Spliced Transcripts

Butterfly's Compacted Sequence Graph



Results

Result: linear sequences grouped in *components*, *contigs* and sequences

```
>TRINITY_DN1000_c101030_g1_i1 len=1072 path=[1:0-85 87:86-136 4106:137-
268 @4179@!:269-517 519:518-532 @534@!:533-694 4261:695-695 4262:696-702
2227:703-719 @2244@!:720-821 2346:822-825 2350:826-848 @4440@!:849-992
2517:993-1008 2533:1009-1071]
AAAGATTTATGATGACAATGACAACGATGGACAACGGACAAAAACAAAATGAAAAAGTAT
TAAATTCTTATACAGGTAGTCATTTATTGAATAGATACATTCTATTTTAAATGAAAATAA
TATTCAAAACAGTTGTTTCTCTGTTCAGTTACTCCGATTTACCCCATTCCATCAACAATG
TCACACATCATTCGGTCAAGTTGATCACAAGGTTCCGTTTCCATGTCATAACCATTATGA
TGCTGAGATGCTTGCAAGAGTTCAGTTTTGACTAAACTTTTGGAATCAAAACTGAAATCT
TCTGGCAAGCTCTGCAGACTGGTCTGAATTAGTTCTTTCACTTTATTCAAATGAGTTTCA
AAGTTCTTTGATATTTGTGCACGCTCTTTCTTCTGTCTCTCCATCATGACACGAAGAGTC
TCCCGAGCTTGATGCGGCCTGAACTCATTTATCAAATGATGCATGTGTATGAATAACAAG
TTGAGATCTTCCAACTTTTCAGTGCGTTTTGTTGAGTCGGGGGCCTTTATTAAAATGTCT
ATCAAATCCAAAAAATTCACCAGAATGGAATGATTCAACTTCTTCAGCTCTCTTTTATGA
TCATAATTCTGTGGATGCAATCTACGAAATCCCTGGGATTCTAAAGGTCTGATGATGGCA
TCATCTGCATTAAAGTGAGCTCCAAACATAGTGTATGTATCCTTCAAAAGAACATGAAAG
CTTTTAATAAAAAGTTTTATCAAGTTTTGTTATGTGTGTTCCCGAGATTTAAAATCATTC
TGTCTTTACATTTCTTTATTCTGTGCTACACCTTTCAAACTACCAGCACATAAATGGGGA
CCTAACAAATCACTGGAATGCATATTACATGTATATTTTGGTGTTAACAATGATTTTTTA
AGTTTTACAATCCTATAAACCTCAAAGATTATAGGAAAATGCTGCACAATATAAAATCTT
TATTCTTATTAGTAACAGTTTAAGAGTAAATCAAATTTTATCTGTATTTAATTTTATCTG
TATTTAATTTTCTATTGAATCAAGACACTCACCTGAATTGGAGGGGGTGGTA
```

```
> TRINITY_DN1000_c101030_g1_i2 len=836 path=[2844:0-16 87:17-
67 4106:68-199 @4179@!:200-448 519:449-463 @534@!:464-625
4260:626-626 4262:627-633 704:634-755 4358:756-756 4359:757-
802 4457:803-835]
GAACCGCTCTTCCGATCTTGAATAGATACATTCTATTTTAAATGAAAATAATATTCAAAA
CAGTTGTTTCTCTGTTCAGTTACTCCGATTTACCCCATTCCATCAACAATGTCACACATC
ATTCGGTCAAGTTGATCACAAGGTTCCGTTTCCATGTCATAACCATTATGATGCTGAGAT
GCTTGCAAGAGTTCAGTTTTGACTAAACTTTTGGAATCAAAACTGAAATCTTCTGGCAAG
CTCTGCAGACTGGTCTGAATTAGTTCTTTCACTTTATTCAAATGAGTTTCAAAGTTCTTT
GATATTTGTGCACGCTCTTTCTTCTGTCTCTCCATCATGACACGAAGAGTCTCCCGAGCT
TGATGCGGCCTGAACTCATTTATCAAATGATGCATGTGTATGAATAACAAGTTGAGATCT
TCCAACTTTTCAGTGCGTTTTGTTGAGTCGGGGGCCTTTATTAAAATGTCTATCAAATCC
AAAAAATTCACCAGAATGGAATGATTCAACTTCTTCAGCTCTCTTTTATGATCATAATTC
TGTGGATGCAATCTACGAAATCCCTGGGATTCTAAAGGTCTGATGATGGCATCATCTGCA
TTAAAGTGAGCTCCAAACATAGTGTAGGTATCCTGAATTGGAGGGGGTGGTAGAGGGACT
CGGCCTCTCTTGACATTTTCATCAGTGTACATATTGAAATACTTGCTGGGAGGGAGAGGG
AAAGCACTTACACCTTGATTCTCTGCCATGGCTTCTGATTATACGCAGAAACAAACTGTC
TTGTTTGTCTATGGTTTGTGTCTTGTTTGATGATTTTCTTCTTCTAGGCTACTTTT
```

TRINITY_DNW_cX_gY_iZ (until release 2.0 cX_gY_iZ previously **compX_cY_seqZ**

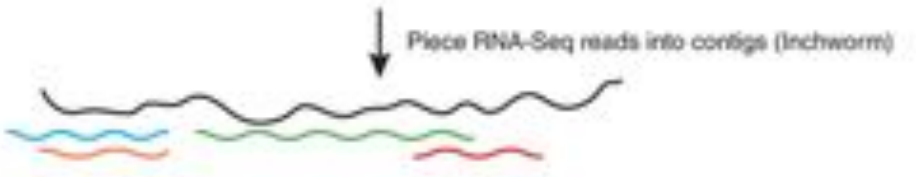
TRINITY_DNW_cX defines the graphical component generated by Chrysalis (from clustering inchworm contigs).

Butterfly might tease subgraphs apart from each other within a single component, based on the read support data .

This gives rise to subgraphs (**gY**).: trinity genes

Each subgraph then gives rise to path sequences (**iZ**). : trinity isoforms

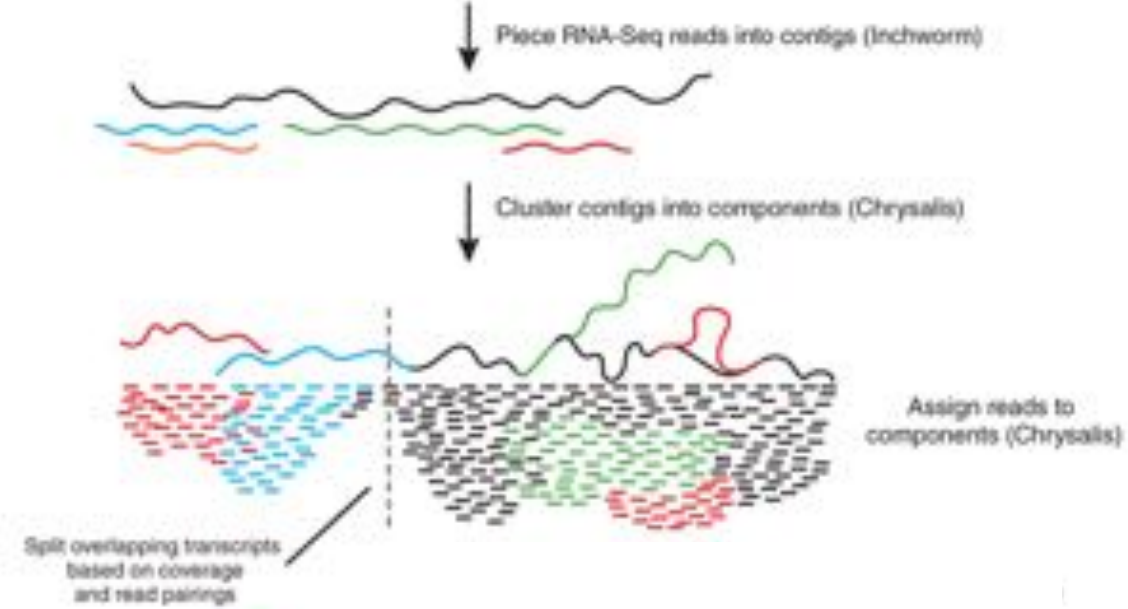
(path) list of vertices in the compacted graph that represent the final transcript sequence and the range within the given assembled sequence that those nodes corresond to.



Summary

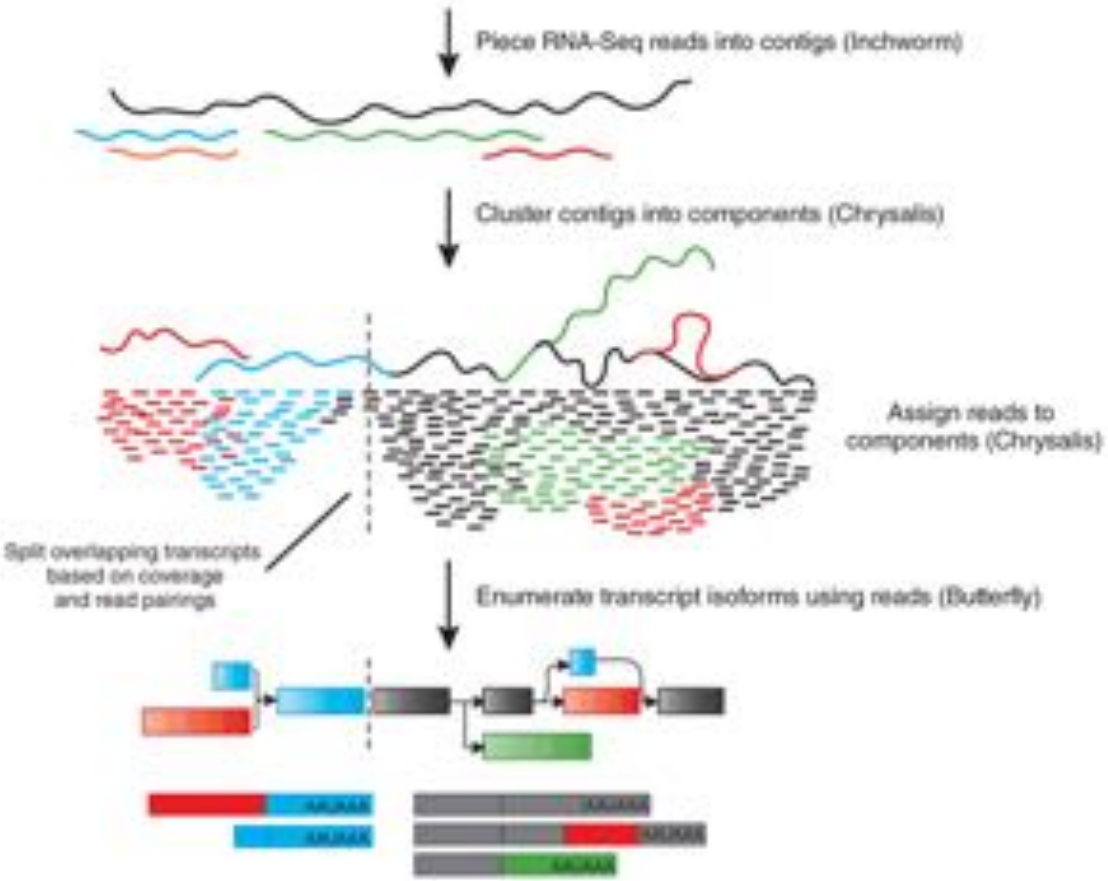
Iyer MK, Chinnaiyan AM (2011)
Nature Biotechnology **29**, 599–600

Summary



Iyer MK, Chinnaiyan AM (2011)
Nature Biotechnology **29**, 599–600

Summary



Iyer MK, Chinnaiyan AM (2011)
Nature Biotechnology **29**, 599–600

Trinity cons.

- Heuristic : re-running the assembly step lead to a different assembly

- Many transcripts (up to 300 000 for 30 000 genes)
  - Add a clustering step (uclust, cap 3)
  - Trinity «REDUCE» option : `Trinity.pl --bfly_opts " --REDUCE "`
  - Exclude the low coverage contigs (FPKM < 1)

- Frequent Version release : 18 versions in 1 ½ year.

Trinity pro

- Integration cleaning step (trimmomatic)
- Integration of normalization steps
  - Accept reads when its average kmer coverage does not exceed a defined threshold
  - Removes reads with too much variability in kmer coverage
- Integration of DE step
- Integration of annotation step (trinotate)
- Accept reference genome
- Accept long reads
- Multiple kmer choice (in progress)

Trinity programs

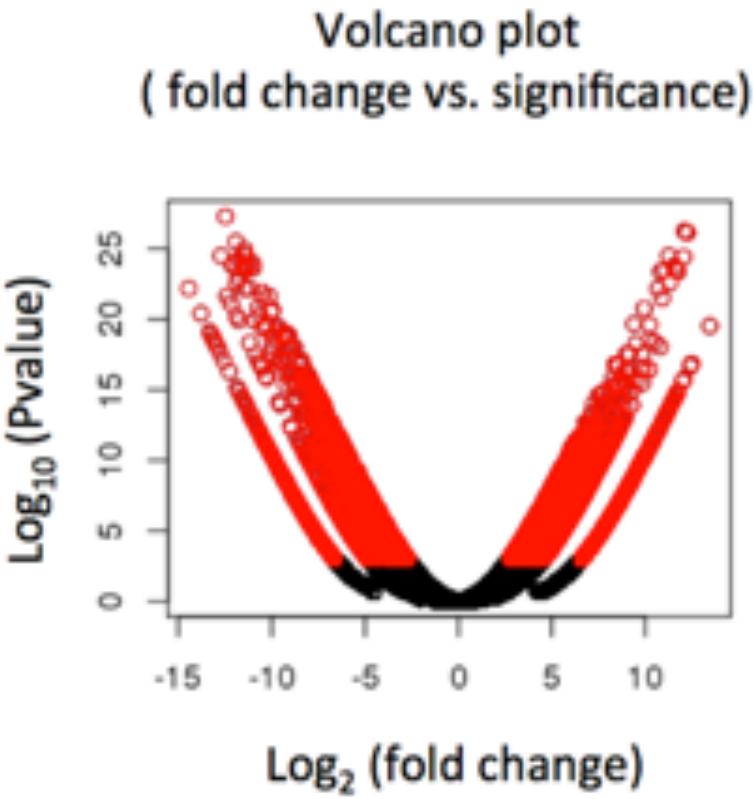
Trinity (perl script to glue it all together)
Inchworm
Chrysalis
Butterfly (Java code – needs Java 1.7)
various utility and analysis scripts (in perl)

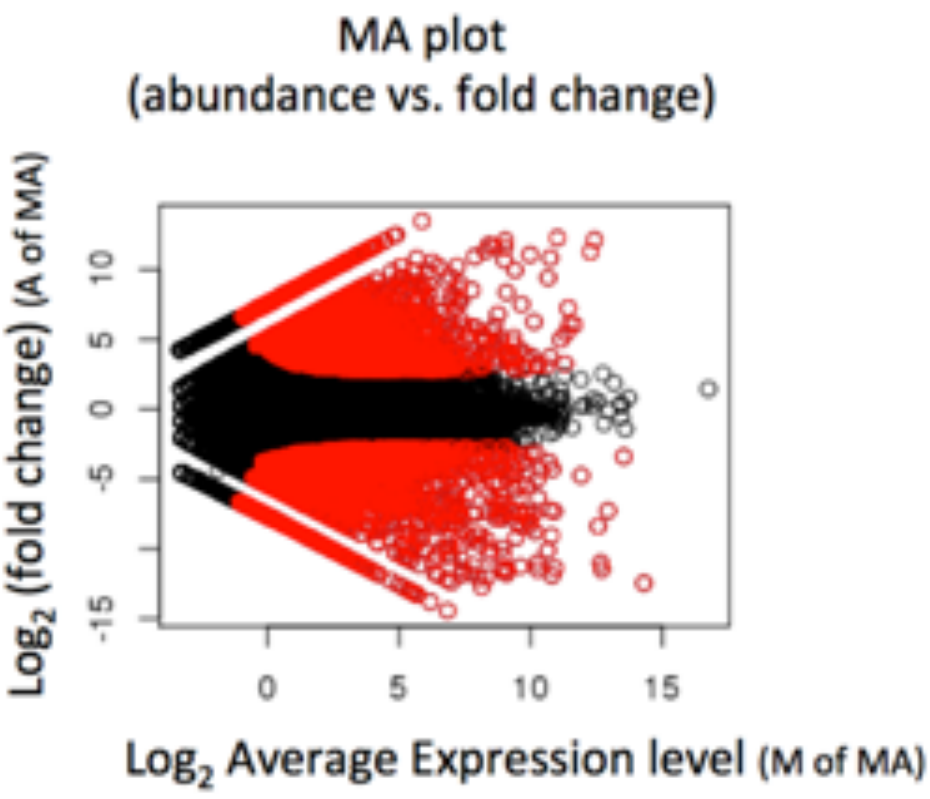
Bundled third-party software

Trimmomatic: clean up reads by trimming and removing adapter remnants (Bolger, A. M., Lohse, M., & Usadel, B)
Jellyfish: k-mer counting software
Fastool: fasta and fastq format reading and conversion (Francesco Strozzi)
ParaFly: parallel driver (Broad Institute)
Slclust: a utility that performs single-linkage clustering with the option of applying a Jaccard similarity coefficient to break weakly bound clusters into distinct clusters (Brian Haas)
Collectl : system performance monitoring (Peter Seger)
Post-assembly analysis helper scripts (in perl)

External software Trinity depends on (needs to be in the search PATH):
samtools
Bowtie
RSEM, eXpress: alignment-based abundance estimation (Bo Li and Colin Dewey)
kallisto, salmon: alignment-free abundance estimation
Transcoder: identify candidate coding regions in within transcripts (Brian Haas - Broad, Alexie Papanicolaou – CSIRO)

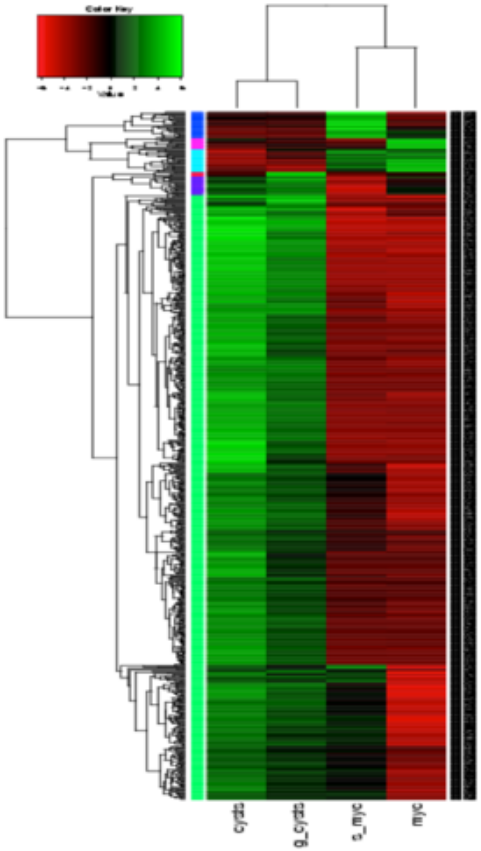
Additional Trinity scripts : Plotting Pairwise Differential Expression Data





Significantly differently expressed transcripts have FDR <= 0.001 (shown in red)

Additional Trinity scripts : Comparing Multiple Samples



Heatmaps provide an effective tool for navigating differential expression across multiple samples.

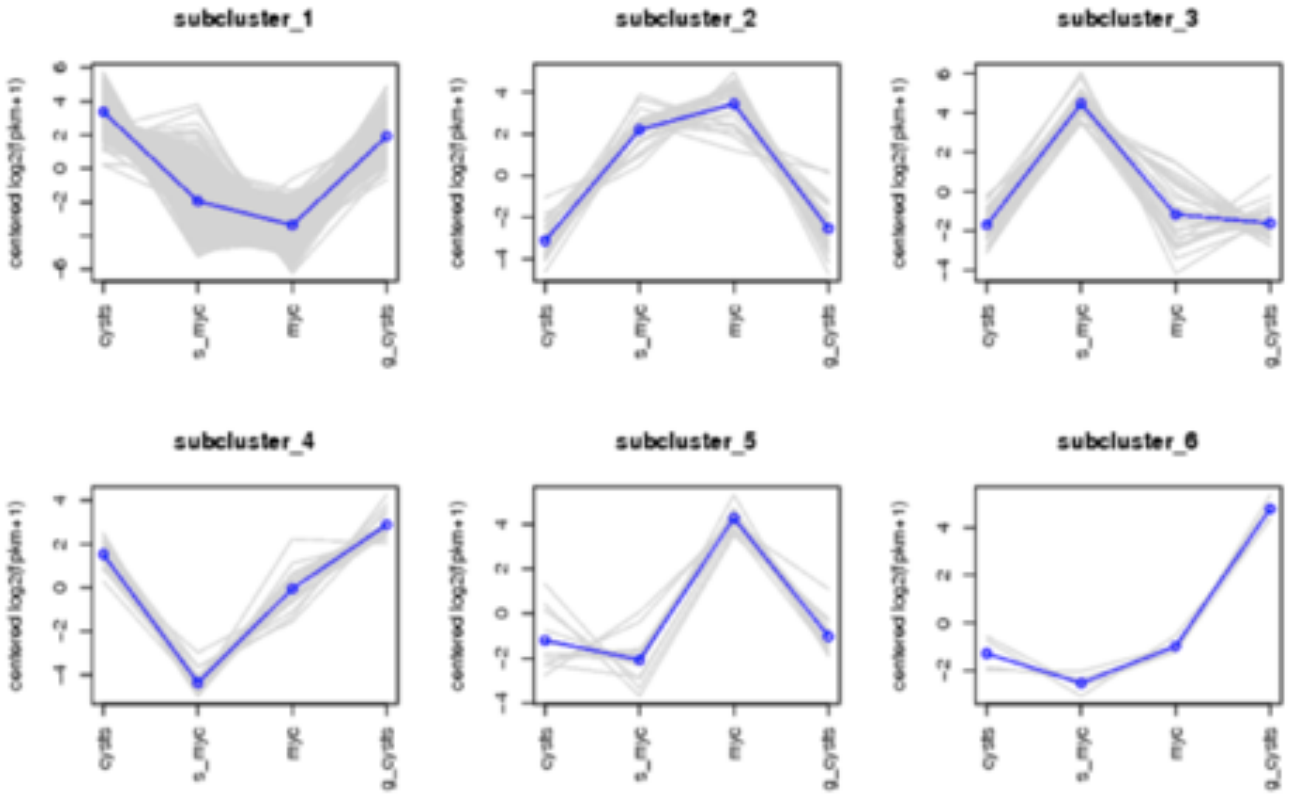
Clustering can be performed across both axes:

-cluster transcripts with similar expression patterns.

-cluster samples according to similar expression values among transcripts.

Examining Patterns of Expression Across Samples

Can extract clusters of transcripts and examine them separately.



Additional Trinity scripts : magic (not so) hidden folders

TrinityStats.pl

filter_low_expr_transcripts.pl

insilico_read_normalization.pl

retrieve_sequences_from_fasta.pl

abundance_estimates_to_matrix.pl
align_and_estimate_abundance.pl
run_DE_analysis_from_samples_file.pl
run_expr_quantification_from_samples_file.pl

analyze_blastPlus_topHit_coverage.pl

Additional Trinity scripts : magic (a little bit more) hidden folder


```
acc_list_to_fasta_entries.pl
alexie_analyze_blast.pl
align_reads_launch_igv.pl
…
fasta_seq_length.pl
fasta_filter_by_min_length.pl
fasta_remove_duplicates.pl
…
map_gtf_transcripts_to_genome_annots.pl
merge_blast_n_rsem_results.pl
merge_rsem_n_express_for_compare.pl
merge_RSEM_output_to_matrix.pl
…
run_HISAT2_via_samples_file.pl
run_jellyfish.pl
run_read_simulator_per_fasta_entry.pl
run_read_simulator_per_gene.pl
```

160 scripts !!!

Trinity usage and options

Typical Trinity command

```
Trinity --seqType fq --max_memory 50G
\--left A_rep1_left.fq  --right A_rep1_right.fq --CPU 4
```

```
Trinity --seqType fq --max_memory 50G --single single.fq  --
CPU 4
```

Running a typical Trinity job requires ~1 hour and ~1G RAM per ~1 million PE reads.

The assembled transcripts will be found at 'trinity_out_dir/Trinity.fasta'.

Trinity statistics

```
TRINITY_HOME/util/TrinityStats.pl Trinity.fasta
```

```
################################
## Counts of transcripts, etc.
################################
Total trinity 'genes': 7648
Total trinity transcripts: 7719
Percent GC: 38.88
########################################
Stats based on ALL transcript contigs:
########################################
Contig N10: 4318
Contig N20: 3395
Contig N30: 2863
Contig N40: 2466
Contig N50: 2065
Median contig length: 1038
Average contig: 1354.26
Total assembled bases: 10453524
```

```
#################################################
## Stats based on ONLY LONGEST ISOFORM per 'GENE':
#################################################
Contig N10: 4317
Contig N20: 3375
Contig N30: 2850
Contig N40: 2458
Contig N50: 2060
Median contig length: 1044
Average contig: 1354.49
Total assembled bases: 10359175
```

Trinity usage and options

Typical Trinity command with multiple samples

```
Trinity --seqType fq --max_memory 50G --CPU 4
\--left A_rep1_left.fq,A_rep2_left.fq
\--right A_rep1_right.fq,A_rep2_right.fq
```

sample.txt

cond_A	cond_A_rep1	A_rep1_left.fq	A_rep1_right.fq
cond_A	cond_A_rep2	A_rep2_left.fq	A_rep2_right.fq
cond_A	cond_A_rep3	A_rep3_left.fq	A_rep3_right.fq
cond_B	cond_B_rep1	B_rep1_left.fq	B_rep1_right.fq
cond_B	cond_B_rep2	B_rep2_left.fq	B_rep2_right.fq
cond_B	cond_B_rep3	B_rep3_left.fq	B_rep3_right.fq

```
Trinity --seqType fq --max_memory 50G --CPU 4
\--samples_file sample.txt
```

Trinity « genome guided »

If your RNA-Seq **sample differs sufficiently** from your reference genome and you'd like **to capture variations** within your assembled transcripts

De novo assembly is restricted to only those reads that map to the genome.

The advantage is that **reads that share sequence in common but map to distinct parts of the genome** will be targeted separately for assembly.

The disadvantage is that reads that do not map to the genome will not be incorporated into the assembly.
-> Unmapped reads can, however, be targeted for a separate genome-free de novo assembly.

Genome guided Trinity command

```
Trinity --genome_guided_bam rnaseq_alignments.csorted.bam --
max_memory 50G --genome_guided_max_intron 10000 --CPU 6
```

The assembled transcripts will be found at 'trinity_out_dir/Trinity-GG.fasta'.

Trinity « longreads »

```
Trinity --seqType fq --max_memory 50G --CPU 4
\--samples_file sample.txt --long_reads contigs.fasta
```

`contigs.fasta:`
fasta file containing error-corrected or circular consensus (CCS) PacBio reads

In short, the Trinity v2.4.0 version uses the pacbio reads mostly for path tracing in a graph that's built based on the illumina reads (not build using illumina and pacbio) .

Trinity including trimming and normalisation

- Trimming

```
 Trinity --seqType fq --max_memory 50G --CPU 4
--samples_file sample.txt --trimmomatic
--quality_trimming_params "ILLUMINACLIP:illumina.fa:2:30:10
SLIDINGWINDOW:4:5 LEADING:5 TRAILING:5 MINLEN:25"
```

- Trimming and normalisation

```
 Trinity --seqType fq --max_memory 50G --CPU 4
--samples_file sample.txt --trimmomatic
--quality_trimming_params "ILLUMINACLIP:illumina.fa:2:30:10
SLIDINGWINDOW:4:5 LEADING:5 TRAILING:5 MINLEN:25
--normalize_by_read_set
```

Trinity usage and options

Typical Trinity command

```
Trinity --seqType fq --max_memory 100G --left reads_1.fq  --
right reads_2.fq --CPU 6
```

Minimum count for K-mers to be assembled by Inchworm

--min_kmer_cov <int> : (default: 1)

➔increase coverage will reduce the contigs size assembled by Inchworm ,

Maximum number of reads to anchor within a single graph (Chrysalis)

--max_reads_per_graph <int> : (default: 200000)

➔decrease time /memory for analysis if reduced

➔decrease sensibility

Maximum length expected between fragment pairs (Butterfly)

--group_pairs_distance <int> : (default: 500)

➔reads outside this distance are treated as single-end.

Trinity usage and options

Graph compaction parameters (Butterfly): `--edge-thr` + `--flow-thr` :

inscrease threshold -> increase graph pruning

➔segmented transcript reconstruction at low coverage regions,

➔lower sensitivity for detection of variant transcripts

Transcript path extension read (pair) overlap requirements (**Butterfly**).

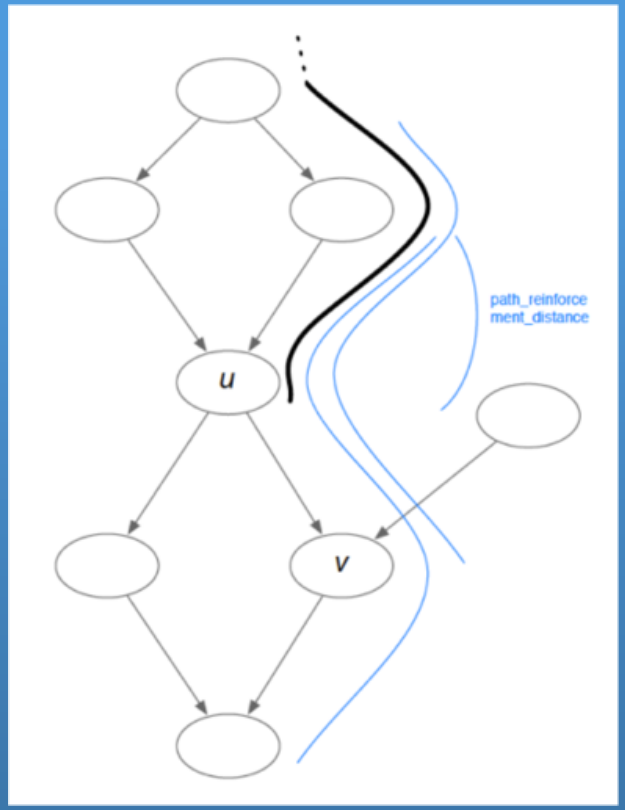
`--path_reinforcement_distance`

(default 75pb)

`-R` (defaults 2 reads support)

➔Using strict parameters (high minimal read support and long reinforcement distance) might prevent the extensions of paths resulting in a partially reconstructed transcript, with breaks at insufficient coverage regions.

➔However, using permissive parameters might fuse several distinct transcripts that have assembled together, possible due to overlapping un-translated regions (UTRs).



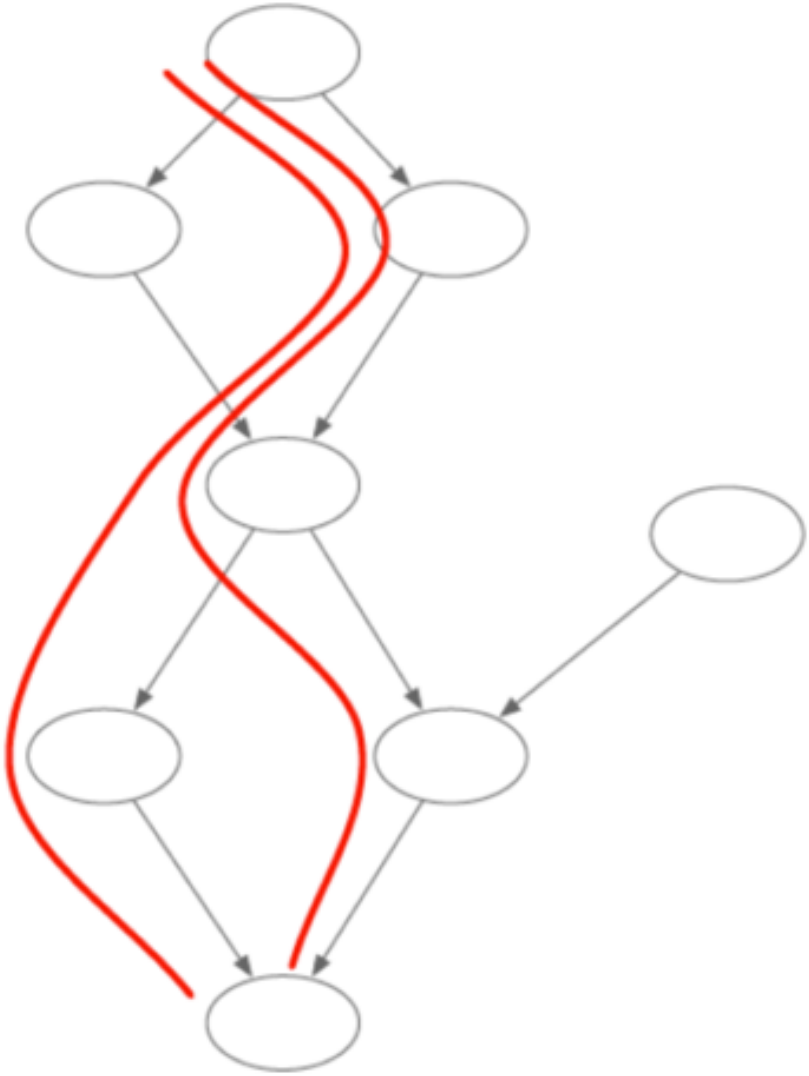
Trinity usage and options

Merging insufficiently different path sequences during reconstruction (Butterfly).

–min_per_id_same_path (default 95%)
–max_diffs_same_path (default 2)
–max_internal_gap_same_path (default 10)

➔Using strict parameter assignments will result in a minimal non-redundant set of output assembled sequences,

➔A more permissive assignment is recommended in order to discover slight variants.

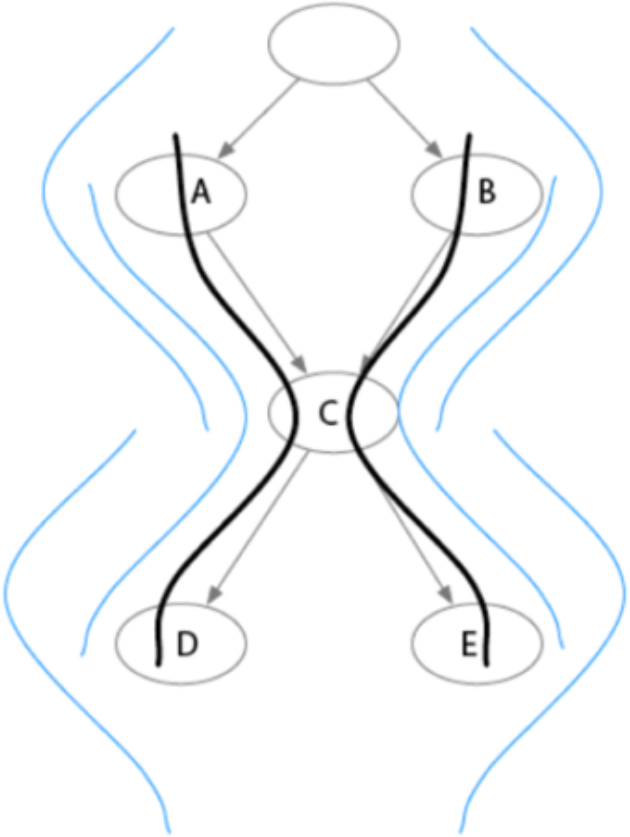


Trinity usage and options

Reducing combinatorial path construction via triplet-locking (Butterfly):

`–triplet-lock` (Butterfly) or
`–no_triplet_lock` (Trinity.pl)

➔Using the *no_triplet_lock* flag will result in a larger number of transcripts reported, of which some might be chimeric (lower specificity), but will ensure that all possible transcripts will be reported (higher sensitivity).

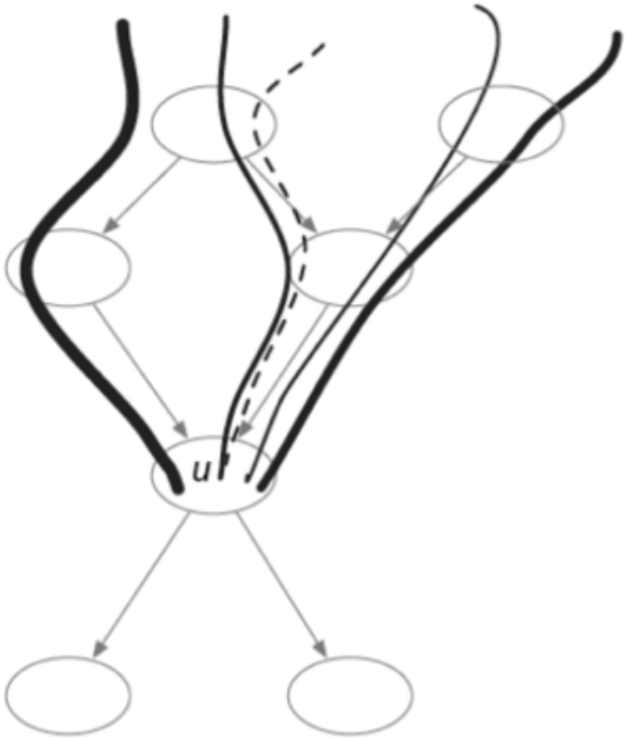


Trinity usage and options

Reducing combinatorial path construction by path restriction (Butterfly):

`–max_number_of_paths_per_node` (default: 10)

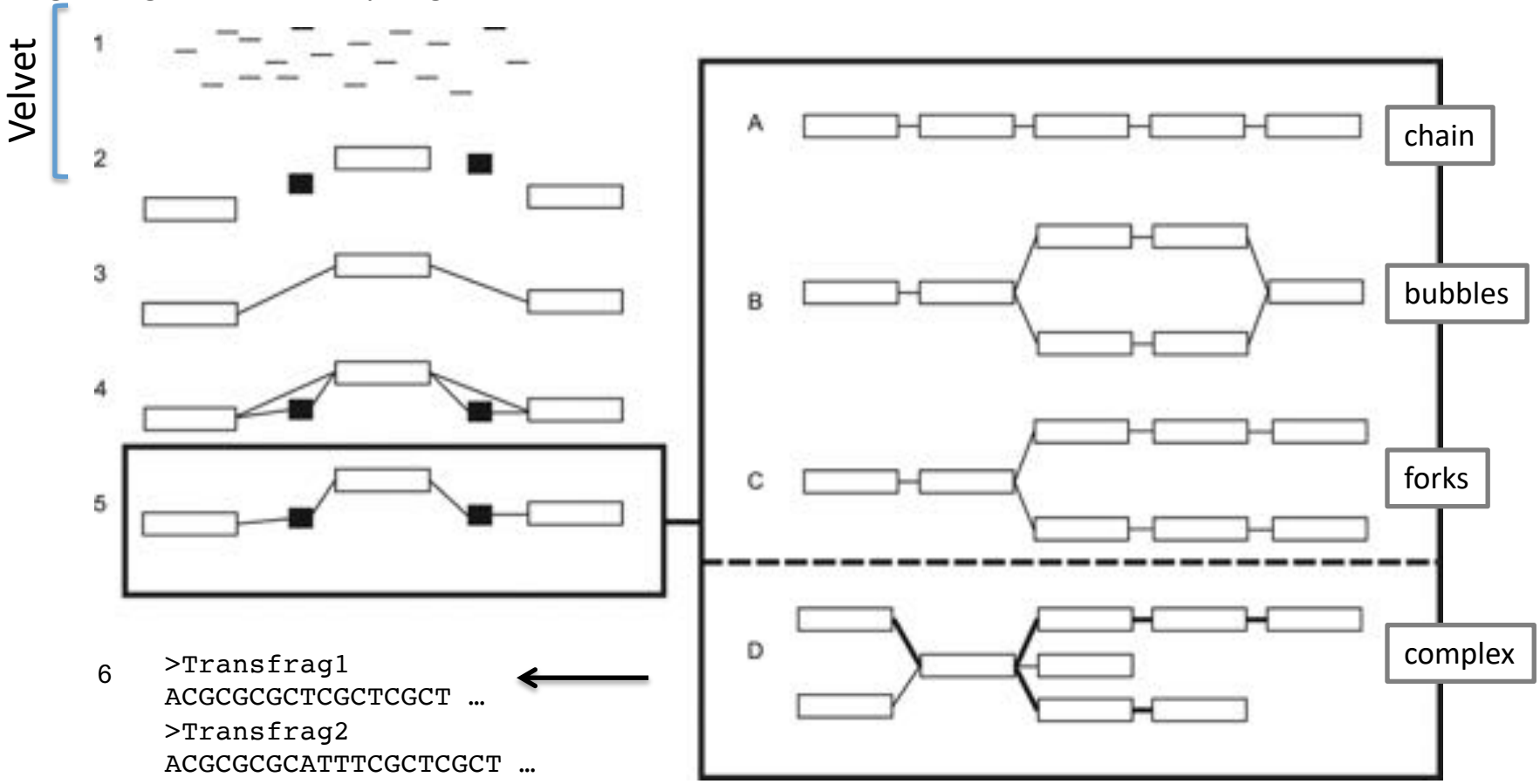
➔Using a low *max_number_of_paths_per_node* parameter will result in a much faster run of butterfly, reporting a minimal non-redundant set of transcripts.



Trinity …. and friends

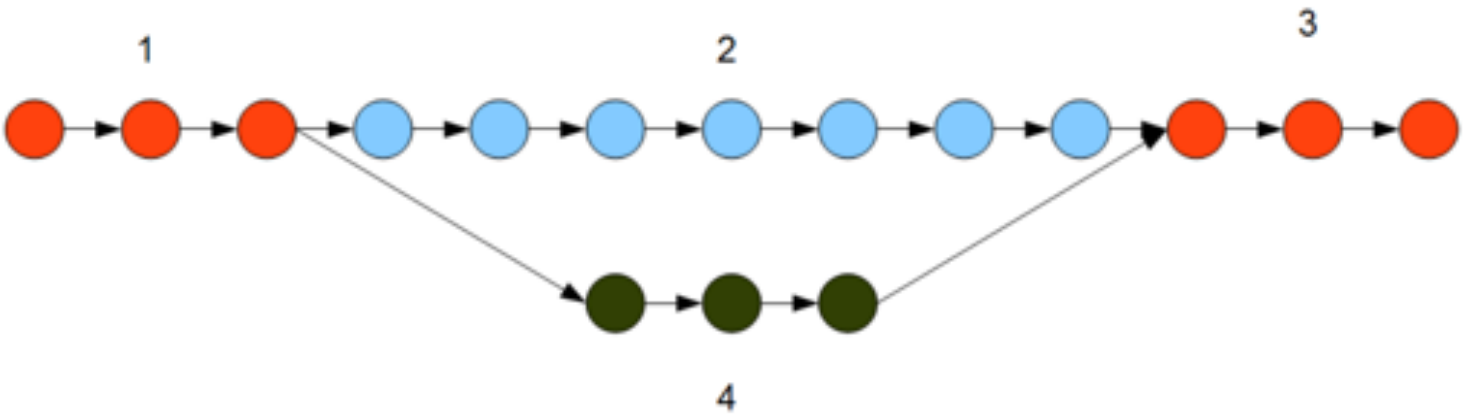
Velvet/Oases

Schematic overview of the Oases pipeline:
(1) Individual reads are sequenced from an RNA sample;
(2) Contigs are built from those reads, some of them are labeled as long (clear), others short (dark);
(3) Long contigs, connected by single reads or read-pairs are grouped into connected components called loci;
(4) Short contigs are attached to the loci; and
(5) The loci are transitively reduced.
(6) Tranfrags are then extracted from the loci



Schulz M H et al. Bioinformatics 2012;28:1086-1092

Velvet/Oases vs Trinity



Velvet searches for connectivity in a de Bruijn graph using a depth search module. The search for a contig stops, as soon as a junction is reached in the de Bruijn graph. So, in the example graph presented above, **Velvet will identify the branches 1, 2, 3 and 4 as separate contigs. The role of Oases is to connect all those separate contigs** to build the gene structures.

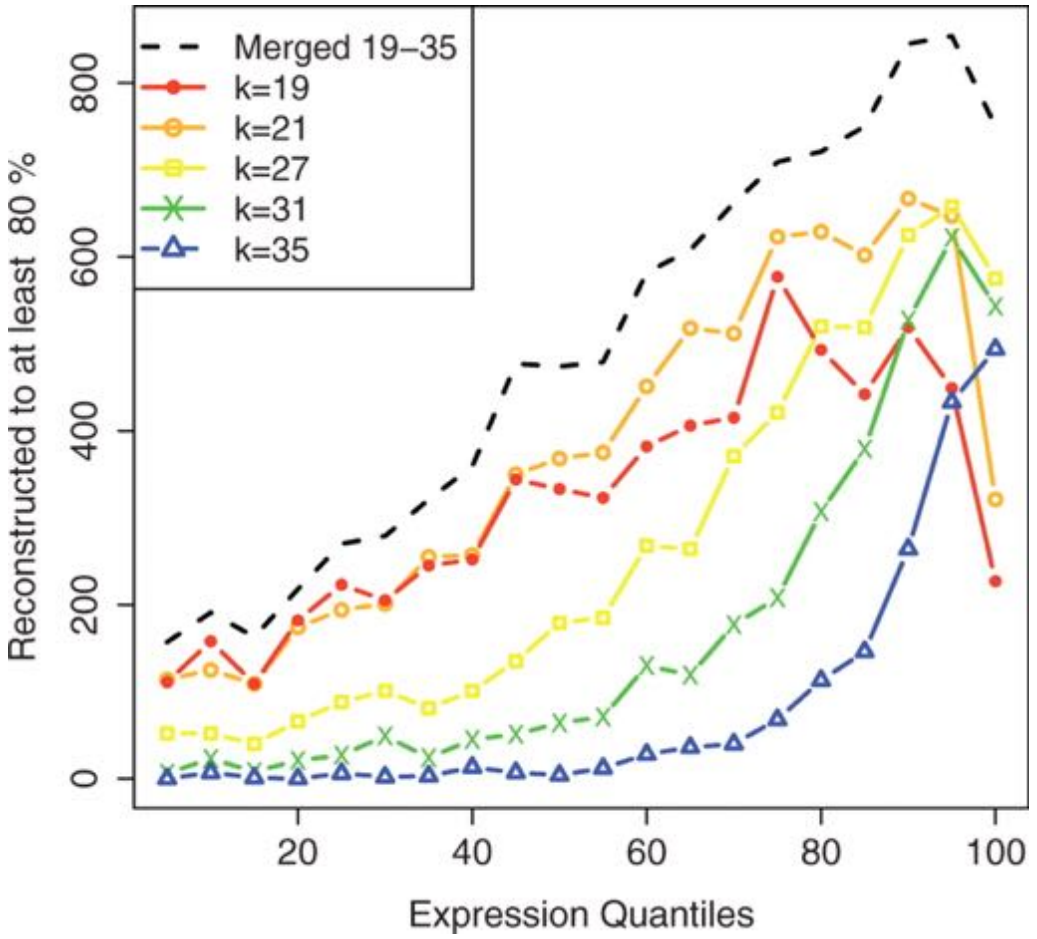
Inchworm is different, because it runs a greedy algorithm that connects as many k-mers as possible without placing any k-mer into two separate contigs. Inchworm does not stop at junctions, but continues forward with assembling contigs. For the graph presented in the above picture, **Inchworm will identify 1+2+3 as one contig and 4 as a different contig**.

Velvet/Oases : single kmer vs merged

Comparison of single k-mer Oases assemblies and the merged assembly from kMIN=19 to kMAX=35 by Oases-M, on the human dataset.

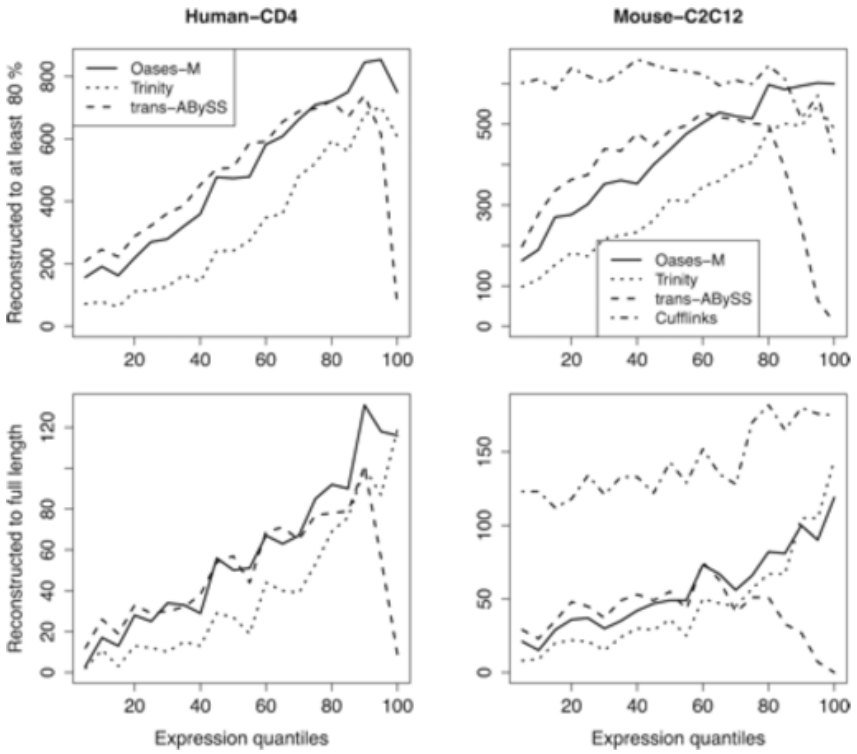
The total number of Ensembl transcripts assembled to 80 of their length is provided by RPKM gene expression quantiles of 1464 genes each.

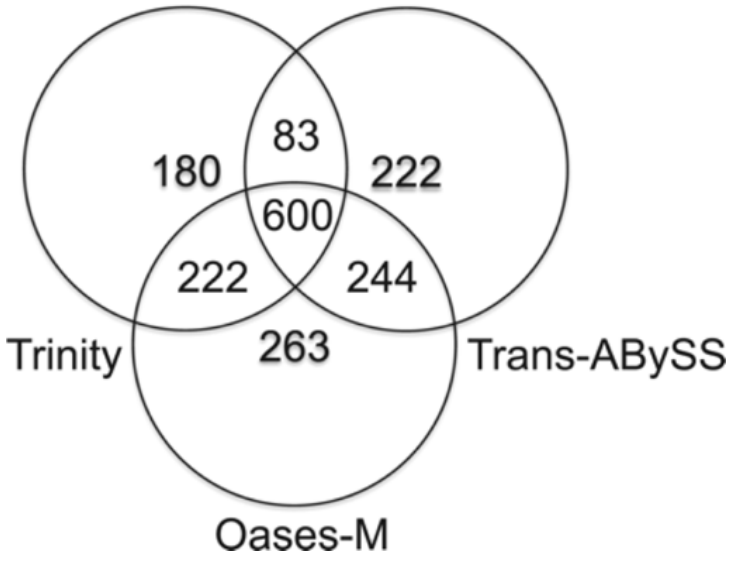
As expected, the assemblies with longer k-values perform best on high expression genes, but poorly on low expression genes. However, short k-mer assemblies have the disadvantage of introducing misassemblies



Schulz M H et al. Bioinformatics 2012;28:1086-1092

Assemblers comparison



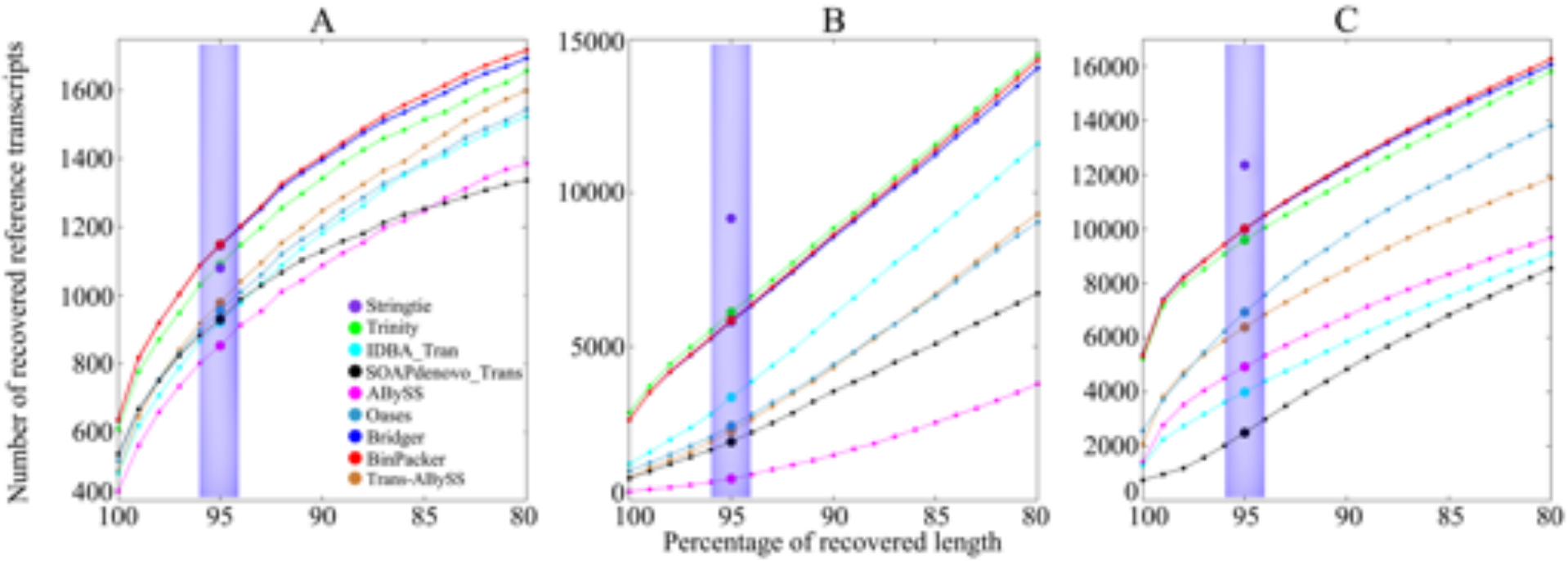


Schulz M H et al. Bioinformatics 2012;28:1086-1092

« In summary, no assembler had consistent good performance in all the statistics.
- For transcriptome assembly of prokaryotic cells that have simple gene structure, Trinity would be recommended.
- For eukaryotic genome, both Oases and Trinity gave acceptable performance. »

Clarke, K., Yang, Y., Marsh, R., Xie, L., & Zhang, K. K. (2013). Comparative analysis of de novo transcriptome assembly. Science China Life Sciences, 56(2), 156–162.

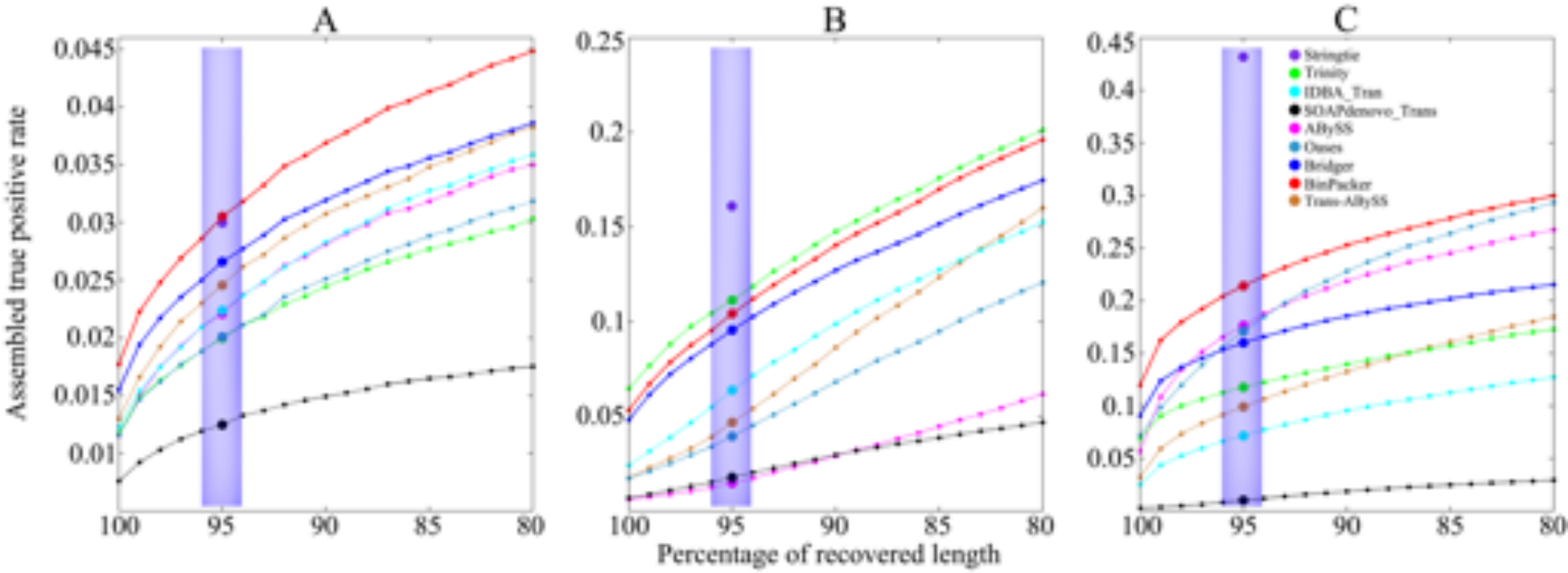
Assemblers comparison



Comparison of recovered reference sensitivity and its distribution against recovered sequence length rates (sequence identity) ranging from 80% to 100% on (A) dog, (B) human and (C) mouse datasets.

Liu J, Li G, Chang Z, Yu T, Liu B, et al. (2016) BinPacker: Packing-Based De Novo Transcriptome Assembly from RNA-seq Data. PLOS Computational Biology 12(2): e1004772. https://doi.org/10.1371/journal.pcbi.1004772
http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004772

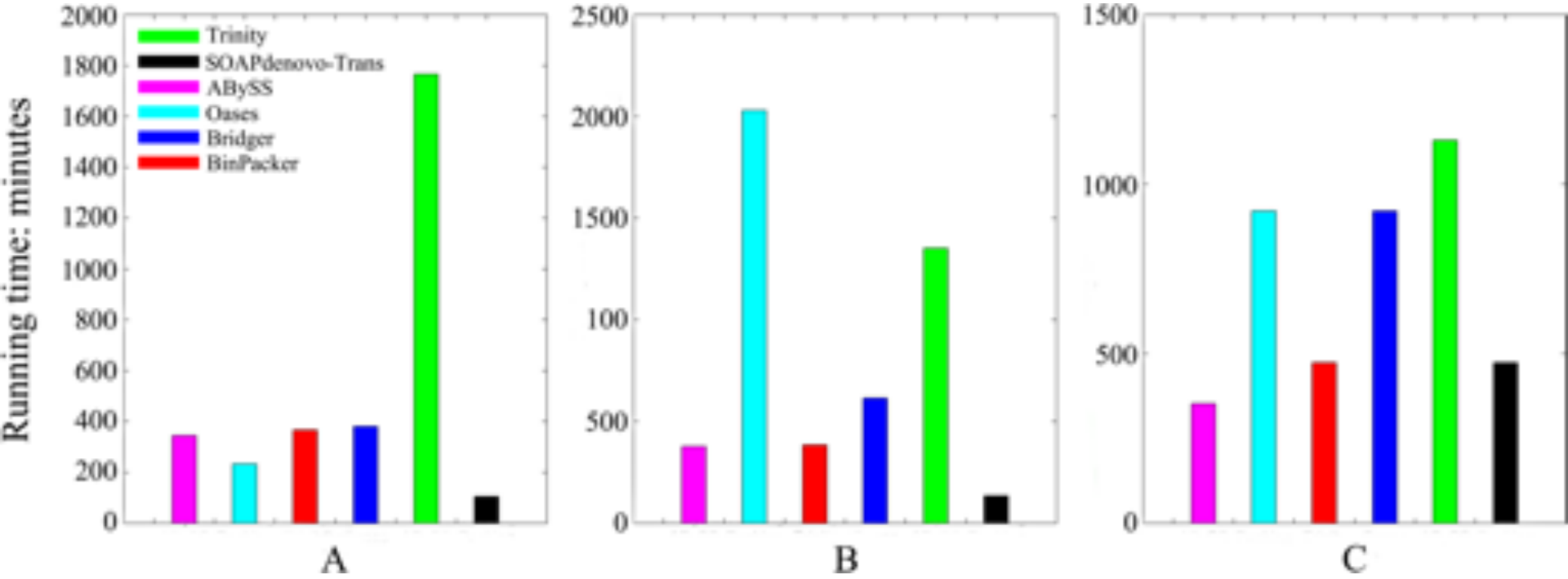
Assemblers comparison



Comparison of assembled true positive rate and its distribution against recovered sequence length rates (sequence identity) ranging from 80% to 100% on (A) dog, (B) human and (C) mouse datasets

Liu J, Li G, Chang Z, Yu T, Liu B, et al. (2016) BinPacker: Packing-Based De Novo Transcriptome Assembly from RNA-seq Data. PLOS Computational Biology 12(2): e1004772. https://doi.org/10.1371/journal.pcbi.1004772
http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004772

Assemblers comparison



Running time for each assembler on (A) dog, (B) human and (C) mouse datasets

Trinity (version 2012-10-05)

Liu J, Li G, Chang Z, Yu T, Liu B, et al. (2016) BinPacker: Packing-Based De Novo Transcriptome Assembly from RNA-seq Data. PLOS Computational Biology 12(2): e1004772. https://doi.org/10.1371/journal.pcbi.1004772
http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004772

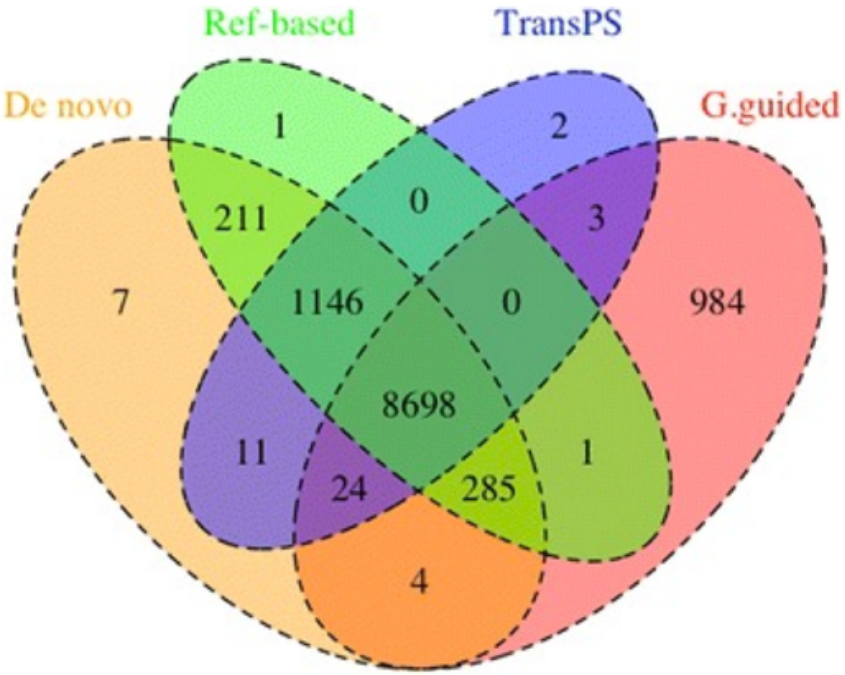
Assemblers comparison

Huang X, Chen XG, Armbruster PA/Comparative performance of transcriptome assembly methods for non-model organisms. BMC Genomics. 2016 Jul 27;17:523. doi: 10.1186/s12864-016-2923-8.

This study compared four transcriptome assembly methods,
- a de novo assembler (Trinity)
- two transcriptome re-assembly strategies utilizing proteomic and genomic resources from closely related species (reference-based re-assembly and TransPS)
- a genome-guided assembler (Cufflinks)

« However, our results emphasize the efficacy of de novo assembly, which can be as effective as genome-guided assembly when the reference genome assembly is fragmented.

If a genome assembly and sufficient computational resources are available, it can be beneficial to combine de novo and genome-guided assemblies »



Assemblers comparison

- Qiong-Yi Zhao et al., Optimizing de novo transcriptome assembly from short-read RNA-Seq data: a comparative study. BMC Bioinformatics 2011, 12(Suppl 14):S2

- Clarke, K., Yang, Y., Marsh, R., Xie, L., & Zhang, K. K. (2013). Comparative analysis of de novo transcriptome assembly. Science China Life Sciences, 56(2), 156–162. doi:10.1007/s11427-013-4444-x

- (Vijay et al., 2013) Challenges and strategies in transcriptome assembly and differential gene expression quantification. A comprehensive in silico assessment of RNA-seq experiments. Molecular ecology. PMID: 22998089

- (Haas et al., 2013) De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. Nature protocols. PMID: 23845962

- (Lu et al., 2013) Comparative study of de novo assembly and genome-guided assembly strategies for transcriptome reconstruction based on RNA-Seq. Sci China Life Sci.

- Chen, G., Yin, K., Wang, C., & Shi, T. (n.d.). De novo transcriptome assembly of RNA-Seq reads with different strategies. Science China Life Sciences, 54(12), 1129–1133. doi:10.1007/s11427-011-4256-9

- (He et al., 2015) Optimal assembly strategies of transcriptome related to ploidies of eukaryotic organisms. BMC genomics. DOI: 10.1186/s12864-014-1192-7

- S. B. Rana, F. J. Zadlock IV, Z. Zhang, W. R. Murphy, and C. S. Bentivegna, “Comparison of De Novo Transcriptome Assemblers and k-mer Strategies Using the Killifish, Fundulus heteroclitus,” *PLoS ONE*, vol. 11, no. 4, p. e0153104, Apr. 2016.

- (Wang and Gribskov, 2016) Comprehensive evaluation of de novo transcriptome assembly programs and their effects on differential gene expression analysis. Bioinformatics. PMID: 27694201

New de novo transcriptome assemblers

- IDBA-Tran (Peng et al., Bioinf., 2014)
- IDBA-MTP (Peng et al., RECOMB 2014)
- SOAPdenovo-Trans (Xie et al., Bioinf., 2014)
- Fu et al., ICCABS, 2014
- StringTie (Pertea et al., Nat. Biotech., 2015)
- Bermuda (Tang et al., ACM, 2015)
- Bridger (Chang et al., Gen. Biol. 2015)
- BinPacker (Liu et al. PLOS Comp Biol, 2016)
- FRAMA (Bens M et al., BMC Genomics 2016)
- rnaSPAdes (Bushmanova et al., bioRxiv, 2018)

Practice

2

Aller sur la practice 2 Assembling transcriptome from RNA-seq *du github*