

CS4330: Combinatorial Methods in Bioinformatics

# Practical genome assembly based on de Bruijn graphs

Wong Limsoon

Acknowledgement: This set of slides were adapted from Ken Sung's



**NUS**  
National University  
of Singapore

National University of Singapore

# Practical issues in genome assembly based on de Bruijn graph

Read errors

Heterozygosity

Repeats

Incomplete coverage



De Bruijn graph becomes big, complicated, & contains many erroneous edges

De Bruijn graph fragments into many connected components

## Choice of K for constructing the de Bruijn graph

# How successful practical genome assemblers deal with these issues

## Velvet

*Efficient in assembling short-read sequencing data*

Zerbino & Birney, “Velvet: Algorithms for *de novo* short read assembly using de Bruijn graphs”, *Genome Research*, 18(5):821-829, 2008

## SOAPdenovo

*Suitable for large-scale genome assembly*

Li et al., “*De novo* assembly of human genomes with massively parallel short read sequencing”, *Genome research*, 20(2):265-272, 2010

## SPAdes (aka St. Petersburg genome assembler)

*Can handle diverse sequencing data types, including short reads, long reads, and mate-pair reads*

Bakewich et al. “SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing”, *Journal of Computational Biology*, 19(5):455-477, 2012



I will describe Velvet. You read up on the rest

# Velvet

[https://en.wikipedia.org/wiki/Velvet\\_assembler](https://en.wikipedia.org/wiki/Velvet_assembler)

Simplification

Remove tips

Merge bubbles

Remove low-coverage edges

## Step 1: Simplify the graph $DB_K(\mathcal{R})$

Whenever node  $x$  has only one outgoing edge, and it goes to a node  $y$  which has only one incoming edge, these two nodes are merged

*Akin to  $x \oplus^K y$*

Do this until no further merging is possible

# Exercise

Is this step “safe”?

If not, how to make it  
safer?

## Step 1: Simplify the graph $DB_K(\mathcal{R})$

Whenever node  $x$  has only one outgoing edge, and it goes to a node  $y$  which has only one incoming edge, these two nodes are merged

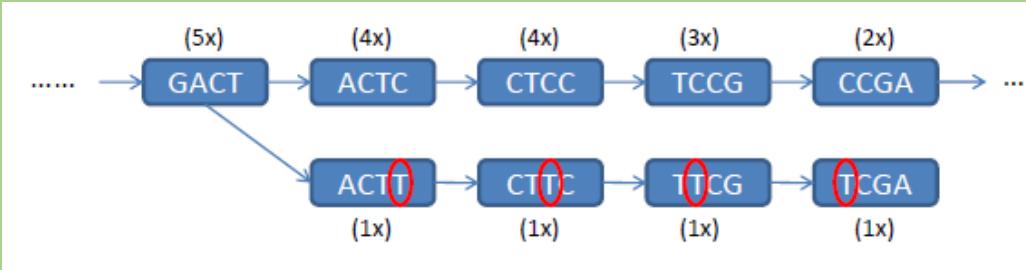
*Akin to  $x \oplus^K y$*

Do this until no further merging is possible

## Step 2: Remove “tips”

“Tips” correspond to edges due to erroneous reads

Do tip removal until no further removal is possible



A “tip” is a chain of nodes such that

*Length is at most  $2K$*

*Disconnected at one end*

*Each node has low # of occurrences in the reads*

# Exercise

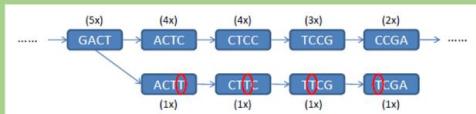
Why is the length of a tip defined to be at most 2K?

What might be other good choices?

## Step 2: Remove “tips”

“Tips” correspond to edges due to erroneous reads

Do tip removal until no further removal is possible



A “tip” is a chain of nodes such that

*Length is at most 2K*

*Disconnected at one end*

*Each node has low # of occurrences in the reads*

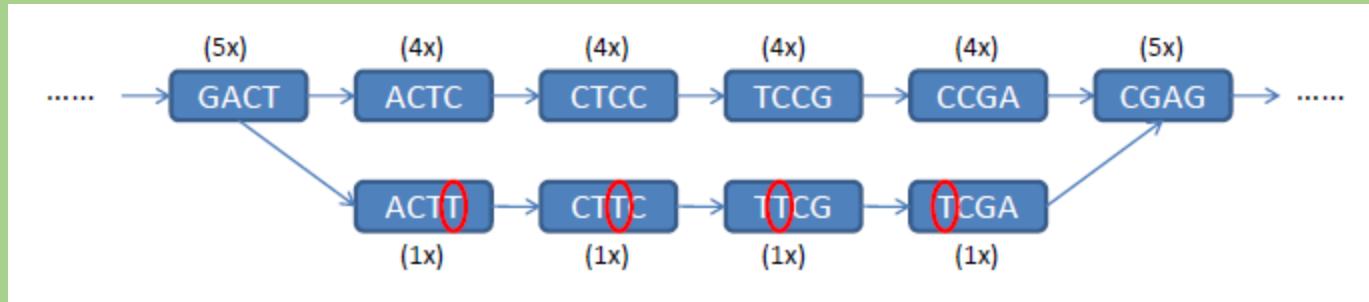


# Step 3: Merge “bubbles”

“Bubbles” correspond to errors or SNPs

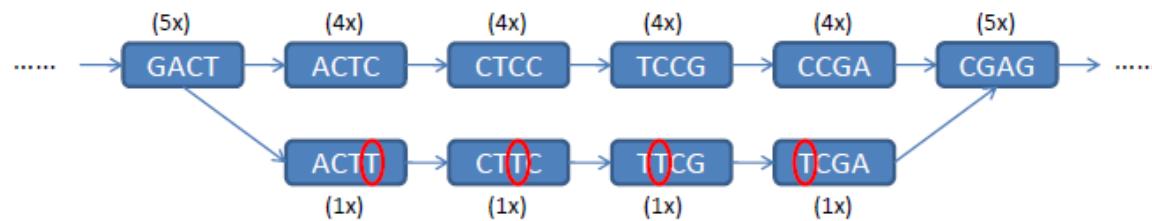
Use “Tour bus” algo to merge bubbles

“Bubbles” are two paths having the same starting node and the same ending node where these two paths represent two strings differing by very few nucleotides (e.g., 1 nucleotide)

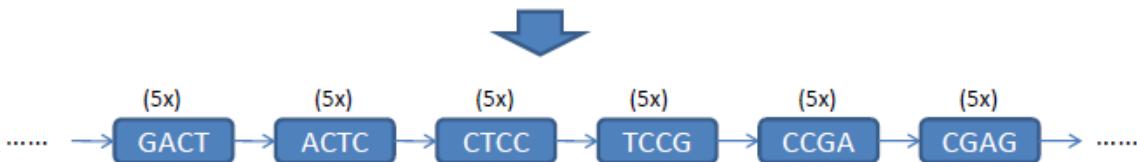


# Merge a bubble

- The top path represents GACTCCGAG.
- The bottom path represents GACTTCGAG.



- There is only one nucleotide different. We merge them.



# Tour bus algorithm

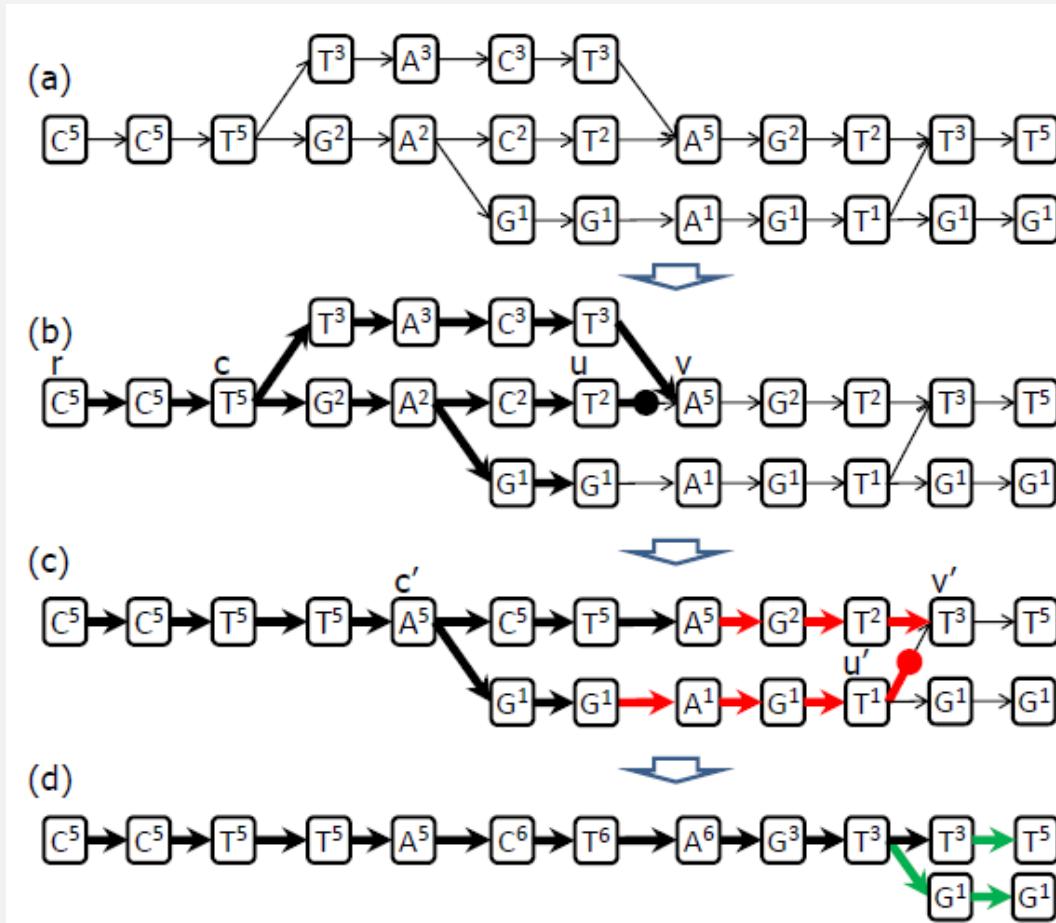
## Algorithm Tour\_Bus( $H, s$ )

**Require:**  $H$  is the de Bruijn graph and  $s$  is an arbitrary node in  $H$

**Ensure:** A graph formed after merging the bubbles

```
1: Set  $Q$  be a queue with one node  $s$ ;  
2: while  $Q \neq \emptyset$  do  
3:    $u = \text{dequeue}(Q)$ ;  
4:   for each child  $v$  of  $u$  do  
5:     if  $\text{visited}[v] = \text{false}$  then  
6:       Set  $\pi(v) = u$ ; /* set  $u$  as  $v$ 's parent in the BFS tree */  
7:       Set  $\text{visited}[v] = \text{true}$ ;  
8:        $\text{enqueue}(Q, v)$ ;  
9:     else  
10:      Find the lowest common ancestor  $c$  of  $u$  and  $v$  by  $\pi()$ ;  
11:      if the paths  $c \rightarrow u$  and  $c \rightarrow v$  are similar enough then  
12:        Merge the two paths and keep the path with the highest path  
           weight;  
13:      end if  
14:    end if  
15:  end for  
16: end while
```

# Example



## Step 4: Remove erroneous connections

Remove edges with low coverage (a cutoff set by users)

# Exercise

How to derive a reasonable threshold for this step?

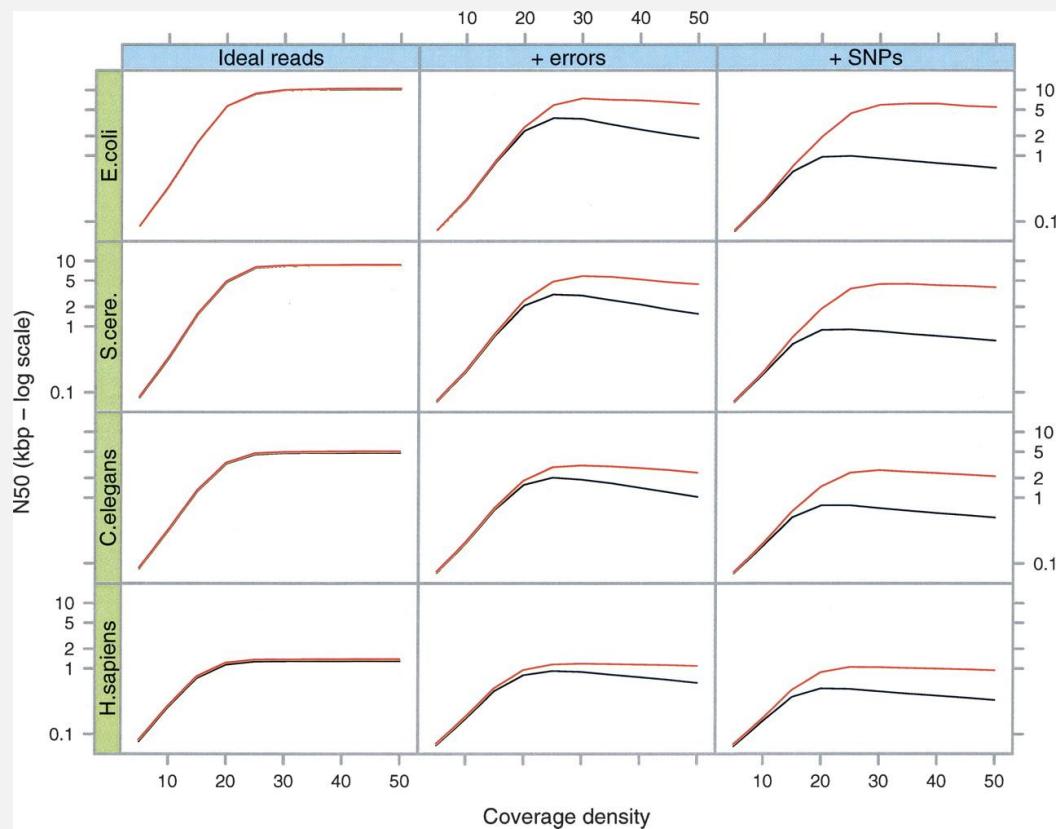
## Step 4: Remove erroneous connections

Remove edges with low coverage (a cutoff set by users)



# Assembly quality

Simulations of Tour Bus. The genome of *E. coli* and 5-Mb samples of DNA from three other species (*S. cerevisiae*, *C. elegans*, and *H. sapiens*, respectively) were used to generate 35-bp read sets of varying read depths (X-axis of each plot). We measured the contig length N50 (Y-axis, log scale) after tip-clipping (black curve) then after the subsequent bubble smoothing (red curve). In the first column are the results for perfect, error-free reads. In the second column, we inserted errors in the reads at a rate of 1%. In the third column, we generated a slightly variant genome from the original by inserting random SNPs at a rate of 1 in 500. The reads were then generated with errors from both variants, thus simulating a diploid assembly.



Zerbino & Birney, *Genome Research*, 18(5):821-829, 2008

# Efficiency

**Table 1.** Efficiency of the Velvet error-correction pipeline on the BAC data set

Step	No. of nodes	N50 (bp)	Maximum length (bp)	Coverage (percent >50 bp)	Coverage (percent >100 bp)
Initial	1,353,791	5	7	0	0
Simplified	945,377	5	80	4.3	0.2
Tips clipped	4898	714	5037	93.5	78.7
Tour Bus	1147	1784	7038	93.4	90.1
Coverage cutoff	685	1958	7038	92.0	90.0
Ideal	620	2130	9045	93.7	91.9

Each line in this table represents a different stage in Velvet. The initial graph was built directly from the BAC reads. The second was the result of node concatenation. The next three graphs were the result of the three consecutive steps of error correction: tip clipping, Tour Bus, and coverage cutoff. The last graph was obtained by building the graph of the reference sequence then submitting it to Tour Bus, to simulate an error-free and gap-free assembly.

**Table 2.** Efficiency of the Velvet error-correction pipeline on the *Streptococcus* data set

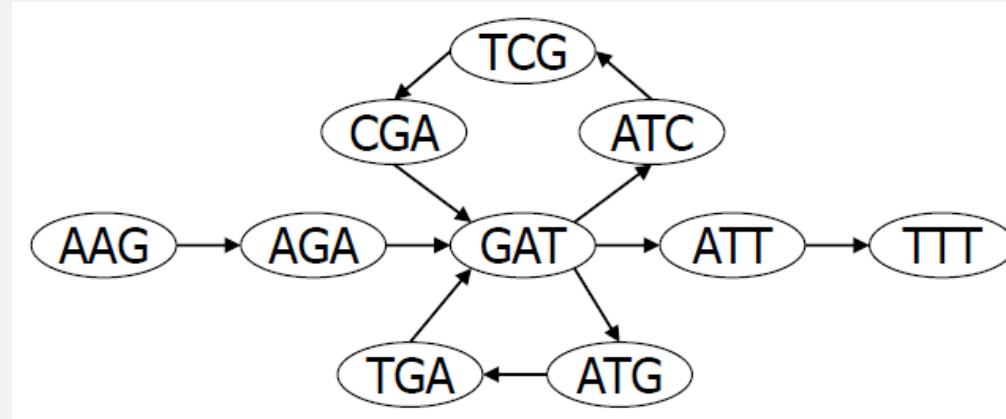
Step	No. of nodes	N50 (bp)	Maximum length (bp)	Coverage (percent >50 bp)	Coverage (percent >100 bp)
Initial	3,621,167	16	16	0	0
Simplified	2,222,845	16	44	0.1	0
Tips clipped	15,267	2195	7949	96.2	95.4
Tour Bus	3303	4334	17,811	96.8	96.4
Coverage cutoff	1496	8564	29,856	96.9	96.5
Ideal	1305	9609	29,856	97.0	96.8

# Recall this example, where $K = 3$

Genome = AAGATCGATGATT

$\mathcal{R} = \{ \text{AAGATC, GATCGAT, CGATGA, ATGATT, GATT} \}$

$\text{DB}_3(\mathcal{R})$ :



Two possible Eulerian paths but can't tell which is real

AAGATCGATGATT

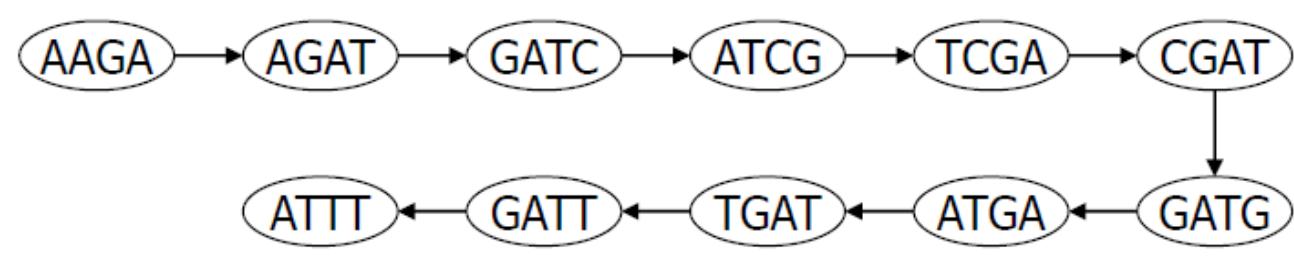
AAGATGATCGATTT

# Consider K = 4

Genome = AAGATCGATGATTT

$\mathcal{R} = \{ \text{AAGATC, GATCGAT, CGATGA, ATGATTT, GATTT} \}$

$\text{DB}_4(\mathcal{R})$ :



A unique Eulerian path:

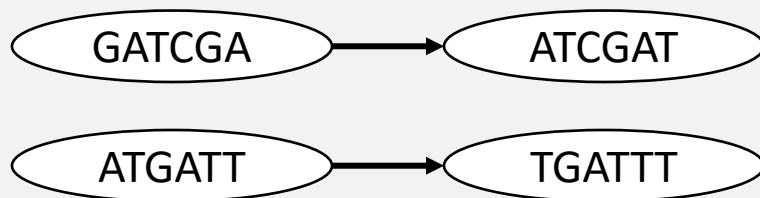
AAGATCGATGATTT

# Consider $K = 6$

Genome = AAGATCGATGATT

$\mathcal{R} = \{ \text{AAGATC, GATCGAT, CGATGA, ATGATT, GATT} \}$

$\text{DB}_6(\mathcal{R})$ :



No Eulerian paths; fragmented graphs give these strings:

GATCGAT, ATGATT

# How to choose suitable K

## Large K

*K-mers more likely to have errors*

*# of correct K-mers is reduced*

*More genome coverage gaps*

## Small K

*More likely to be repeated*

*Short repeats create loops and branches in de Bruijn graph*

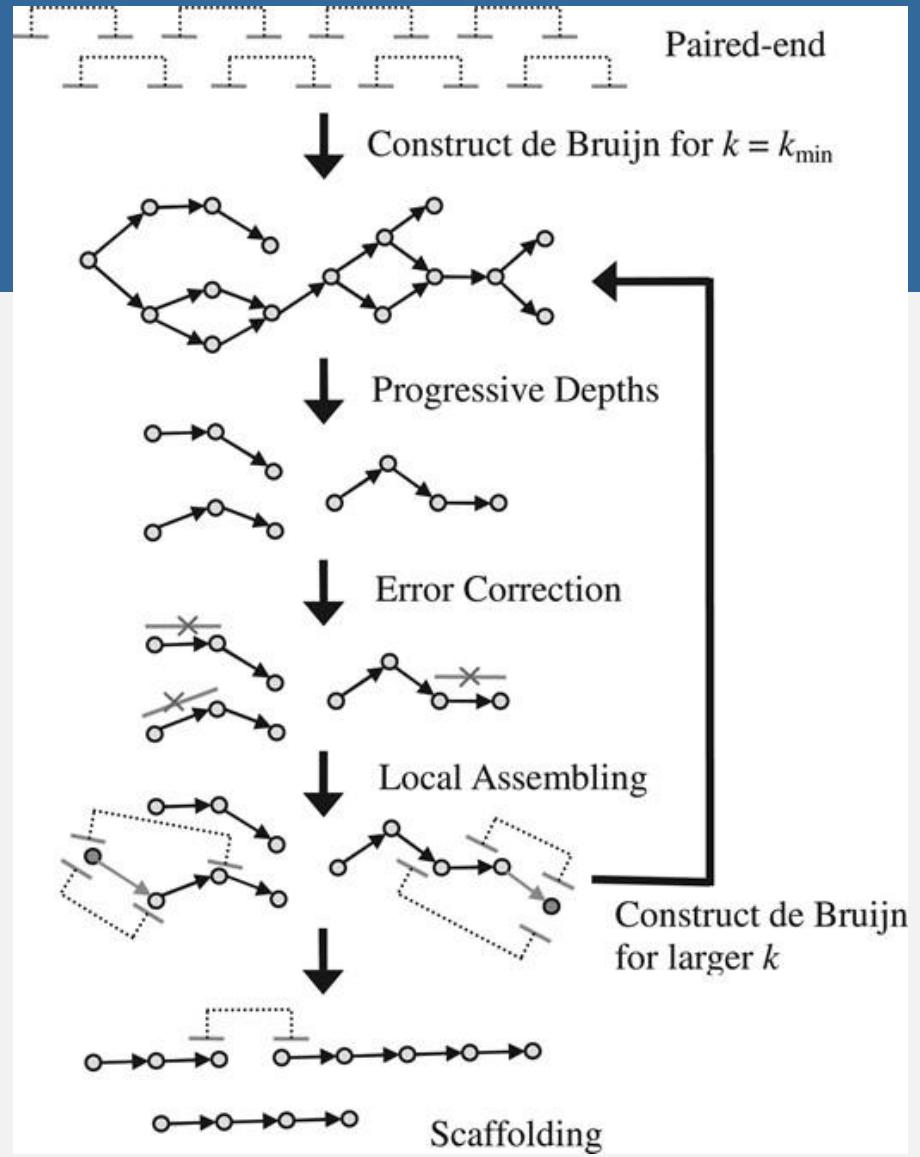
# IDBA

Instead of a fixed  $K$ , try different values

*When  $K$  is small, we get short but high-quality contigs*

*Use them to correct errors in reads*

*Then, increment  $K$  and try again*



Peng, et al., "IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth", Bioinformatics, 28(11): 1420-1428, 2010

# IDBA algorithm

Step 7 uses techniques similar to reference based genome assembly

## Algorithm IDBA( $\mathcal{R}, k_{min}, k_{max}$ )

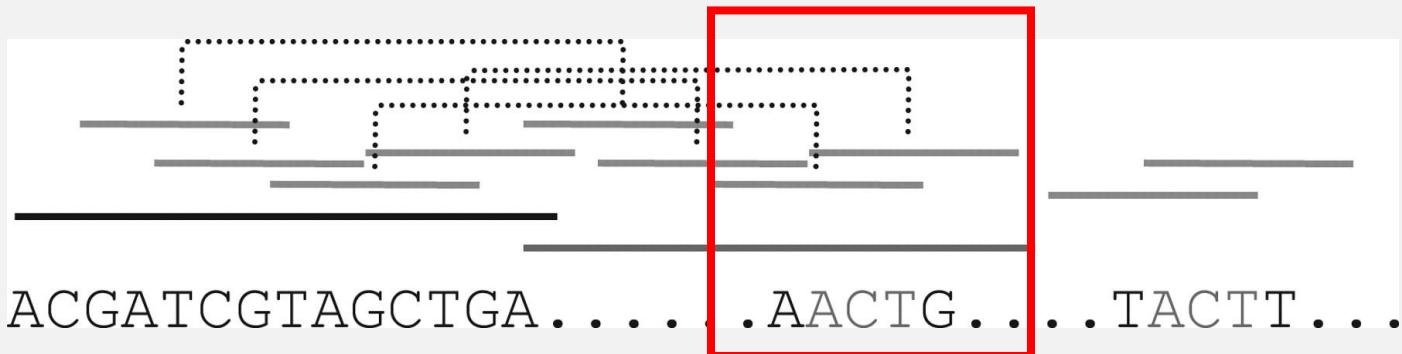
**Require:**  $\mathcal{R}$  is a set of reads and  $k_{min}$  and  $k_{max}$  are de Bruijn graph parameter

**Ensure:** A set of contigs

- 1: **for**  $k = k_{min}$  to  $k_{max}$  **do**
- 2:   Generate the de Bruijn graph  $H_k$  for  $\mathcal{R}$ ;
- 3:   Remove tips;
- 4:   Merge bubbles;
- 5:   Remove nodes with multiplicity  $\leq m$ ;
- 6:   Extract all maximal simple paths in  $H_k$  as contigs;
- 7:   All reads in  $\mathcal{R}$  are aligned to the computed contigs;
- 8:   The mismatch in the read is corrected if 80% of reads aligned to the same position has the correct base;
- 9: **end for**
- 10: Extract all maximal simple paths in  $H_{k_{max}}$  as contigs;

Velvet

# Low-coverage regions remain problematic



A region is covered by two reads ...AACT and ACTG...

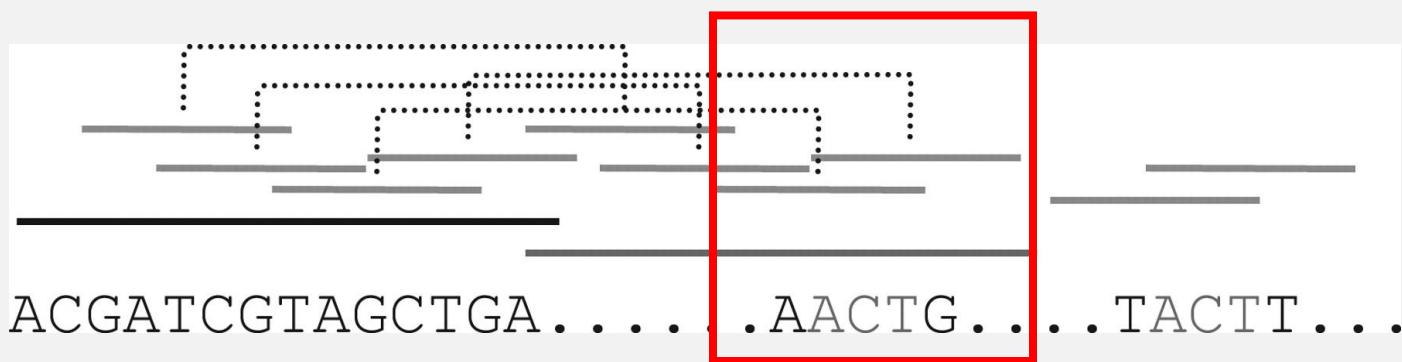
At  $K = 3$ ,  $DB_K(\mathcal{R})$  shows a path  $AAC \rightarrow ACT \rightarrow CTG$  but cannot output the contig ... AACTG ...

$AACTG$  is not in any read

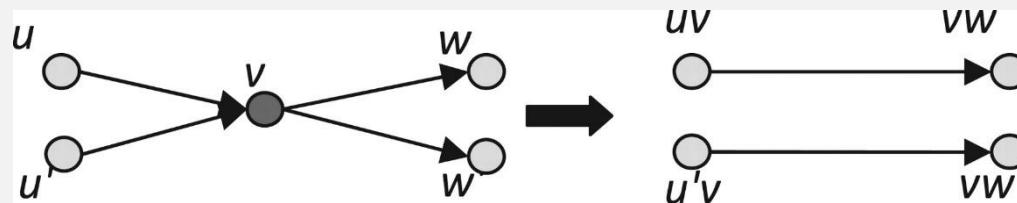
$ACT$  has 2 incoming & 2 outgoing edges due to TACTT

Similar problems exist at  $K = 4, 5, \dots$

# IDBA-UD $\approx$ IDBA + local assembly



Using information from paired-end read mapping, the two copies of ACT can be separated



Then, ...AACTG ... can be produced

# Performance

**Table 2.**

The assembly results on simulated 10× lenght-100 reads of *L.plantarum* (~3.3 Mb) with 1% error rate

<i>k</i>	Contigs							Scaffolds							Time (s)	Mem (M)
	No.		N50	Max	Cov	Sub.err	Err no.	No.		N50	Max	Cov	Sub.err	Err no.		
			len	(%)	(%)	(len)			len	(%)	(%)	(len)				
IDBA-UD	20–100	210	36 513	201 860	99.56	0.0225	104 437	83	194 322	406 269	99.55	0.0218	53 784	63	432	
SOAPdenovo	31	3346	1584	8691	98.36	0.0572	1 079 112k	147	121 214	246 514	92.50	0.0483	1 087 283k	31	852	
Velvet	21	473	13 761	48 489	98.09	0.0323	515k	111	111 871	225 438	96.81	0.0291	667k	43	526	
IDBA	20–40	672	8350	37 391	98.52	0.0164	33 301	60	119 931	308 798	97.55	0.0161	39 420	24	414	

Peng, et al., “IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth”, *Bioinformatics*, 28(11): 1420-1428, 2010

# Limitations

De Bruijn graph is big

*Need lots of memory to run*

Cannot use connectivity of paired-end reads before scaffolding

*SPAdes uses paired de Bruijn graph to capture this info*

If there are long repeats, this approach may fail to get long contigs

# Exercise

IDBA tries increasingly bigger values for K

*I.e., it must assemble the genome multiple times*

Would it be possible to guess a good value for K by just taking a quick look at the read set?

How?

## Hypothesis a la KmerGenie

Empirical evidence from KmerGenie

*When a value  $K^*$  of K maximizes the set of **genomic K-mer species** observed in a read set  $\mathcal{R}$  for a genome G, then  $K^*$  is the best value for producing the genome assembly from  $\mathcal{R}$  for G*

A **genomic K-mer species** is a K-mer species that indeed appears in the genome G

# Good to read

[Velvet] Zerbino & Birney, “Velvet: Algorithms for de novo short read assembly using de Bruijn graphs”, *Genome Research*, 18(5):821-829, 2008, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2336801/>

[IDBA-UD] Peng, et al., “IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth”, *Bioinformatics*, 28(11): 1420-1428, 2010,  
<https://pubmed.ncbi.nlm.nih.gov/22495754/>

# Good to read

[SOAPdenovo] Li et al., “De novo assembly of human genomes with massively parallel short read sequencing”, *Genome research*, 20(2):265-272, 2010, <https://pubmed.ncbi.nlm.nih.gov/20019144/>

[SPAdes] Bakevich et al. “SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing”, *Journal of Computational Biology*, 19(5):455-477, 2012, <https://pubmed.ncbi.nlm.nih.gov/22506599/>

[KmerGenie] Chikni & Medvedev, “Informed and automated k-mer size selection for genome assembly”, *Bioinformatics*, 30(1):31-37, 2014, <https://pubmed.ncbi.nlm.nih.gov/23732276/>