Integration of string and de Bruijn graphs for genome assembly

Huang YT, Liao CF. *Bioinformatics*, 32(9); 2016

JC 11.05.2016
Fanny-Dhelia Pajuste

Motivation

- Two graph models mostly used for genome assembly are string and de Brujin graphs
- String graph can use entire reads for better assembly of repetitive regions
- De Brujin graphs use k-mers for better assembly of error-prone regions
- None of the assemblers clearly outperforms the others across all datasets
- The new assembler StriDe uses the best features of both graph models
- The results are comparable to other methods for both long and short read data

Genome assembly

- Although next-generation sequencing and genome assembly from sequencing reads are widely used, the assembled genomes are still fragmented
- Fragmentation of the assembled genomes cause problems in downstream processing (estimation of gene family size, comparative analysis etc)
- The accuracy, contiguity and speed of existing assemblers have been compared in many projects, but none of these clearly outperforms others

OLC graph vs de Brujin graph

OLC (overlap layout consensus):

- Assemblers: MIRA, Newbler, Celera
- Graph: a vertex contains a read, an edge denotes to two overlapping reads
- A feasible layout of reads is found from the graph
- Good for long reads
- Inefficient for great amount of short reads

De Brujin graph:

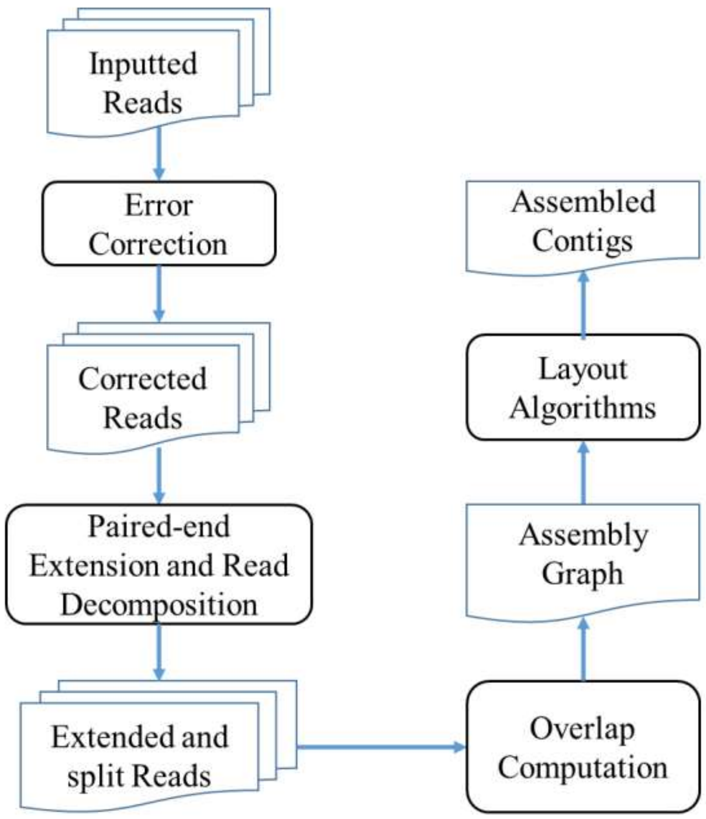
- Assemblers: SOAPdenovo, Velvet, ABySS, ALLPATHS
- Divides reads into k-mers
- Graph: A vertex contains a *k*-mer, an edge denotes to two adjacent *k*-mers overlapping by *k*-1 letters
- More efficient
- Doesn't work well with repetitions longer than *k*
- SPAdes – paired *k*-mers from paired-end reads

String graph

- Similar to OLC graph, but without transitive edges
- Assembler: SGA (String Graph Assembler)
- Can assemble mammalian-size genomes
- Uses many properties of OLC and de Brujin but doesn’t work as well with real data
- The main problem is assembling regions that are error-prone due to sequencing bias

StriDe additions

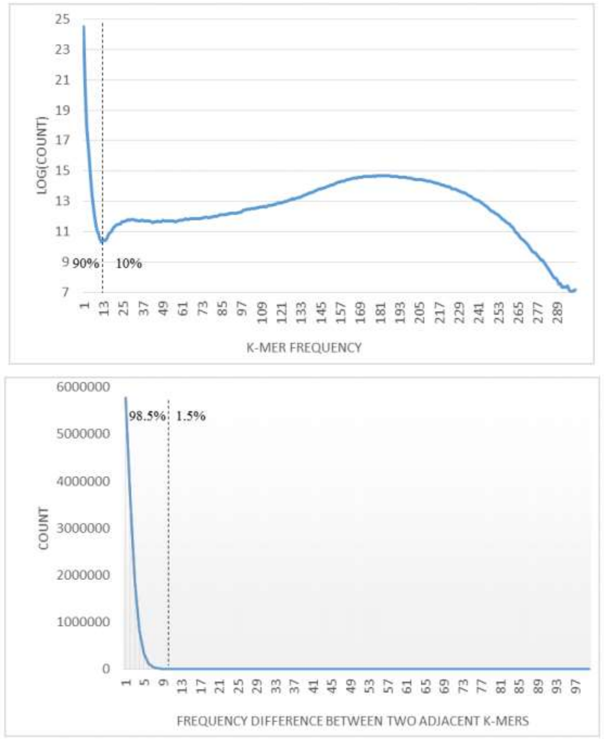
- Decomposes reads in error-prone regions
- Extends paired-end reads to long reads using FM-index
- Improved error correction algorithm
- Improved overlap computation
- Specialized layout algorithms
- Full parallelization



Supplementary Figure S2. Workflow of StriDe assembler The black rectangles represent software components and the remaining are input/output of each component.

Error correction

- Usually a k-mer frequency cut-off is used (depends on sequencing bias and repeats)
- The difference of adjacent k-mer frequencies are small and stable regardless of the repetitions and low coverage
- Large frequency differences indicate sequencing errors
- Error base is identified if the difference for adjacent k-mers is bigger than a given threshold (default: 10)



Supplementary Figure S3. Upper: the *k*-mer frequency spectrum (in $\log_2$ scale) of *S. aureus* from GAGE-b. Most approaches identify the first valley found in the spectrum as the cutoff for error frequency. ~90% of *k*-mers are below the cutoff, implying many of them are due to sequencing bias. Lower: the differences of 18 million adjacent 31-mers sampled from GAGE-b. ~98.5% of adjacent frequency differences are less than 10, regardless of repetitive or low-coverage regions.

- The frequency turbulence is eliminated by replacing the error base letter
- For indel and cluster errors seed-and-extend approach is used to align overlapping reads
- Banded dynamic programming is used to compute exact alignment

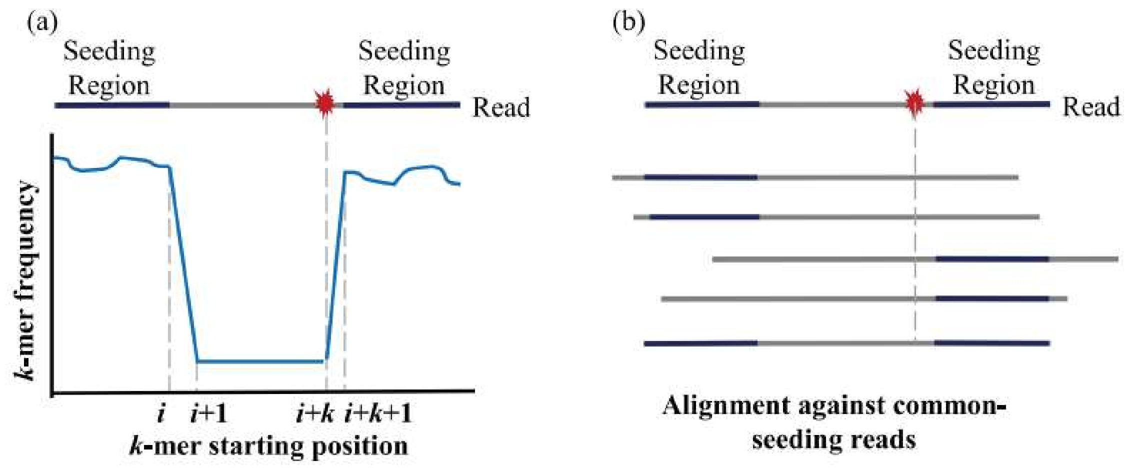


Fig. 3. (**a**) The *k*-mer frequency drops when the rightmost base reaches the error and increases after leaving the error; (**b**) The *k*-mer seeds are selected from the regions flanking the low-frequency region and are used for identifying potentially overlapping reads

Extension of paired-end reads

- The paired-end reads are extended inward into long reads
- Finds feasible extensions from the first to the other end (FM-index walk)

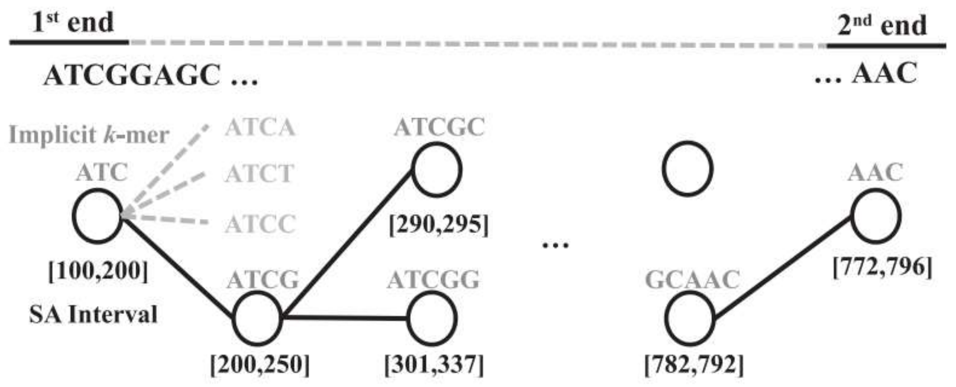
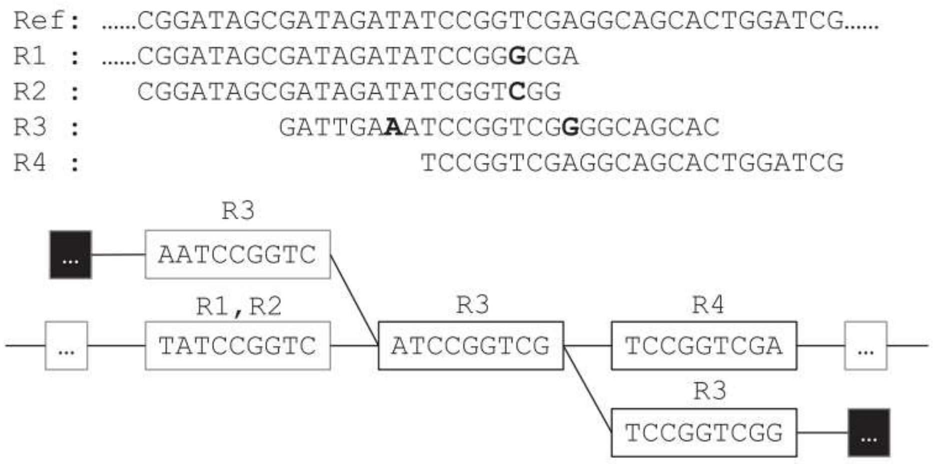


Fig. 4. Illustration of the FM-index walk within a paired-end read. The implicit/initial *k*-mer size is 3 and shown only for illustration purposes. Each tree node internally stores forward and reverse suffix array (SA) intervals. The solid lines represent successful extensions of the existing SA interval and creation of new leaf nodes, where the dashed lines represent infeasible SA intervals: no new nodes are created

Read decomposition in Error-prone regions

- Paired-end reads that can not be extended are often error-prone
- Reads are decomposed into subreads in each error base
- Read is broken into k-mers, the breakpoint is found if one of them has overlap to other k-mer(s)
- K-mer frequency has to be <3
- Subreads can be used later for assembly



Ref: ......CGGATAGCGATAGATATCCGGTCGAGGCAGCACTGGATCG......
R1 : ......CGGATAGCGATAGATATCCGGGCGA
R2 : CGGATAGCGATAGATATCGGTCGG
R3 : GATTGAAATCCGGTCGGGGCAGCAC
R4 : TCCGGTCGAGGCAGCACTGGATCG
R3
AATCCGGTC
R1,R2
TATCCGGTC
R3
ATCCGGTCG
R4
TCCGGTCGA
R3
TCCGGTCGG
...

Fig. 5. Illustration of decomposition across four reads (R1–R4). The error is boldfaced, and error subreads are shown in black boxes, which are often tips in the graph. Correct subreads are thus overlapped and can be assembled at a later stage

Overlap computation

- Most time-consuming part of the graph creation
- The minimum overlap length has to be *k*-1 (due to the decomposed subreads)
- Short overlap length leads to a huge amount of edges and affects greatly the efficiency and the memory usage
- A pruning procedure removes small overlaps with other reads if longer overlaps are present
- The length difference has to be greater than a given threshold (default: half of the read length)
- The number of retained overlaps equals to the sequencing coverage

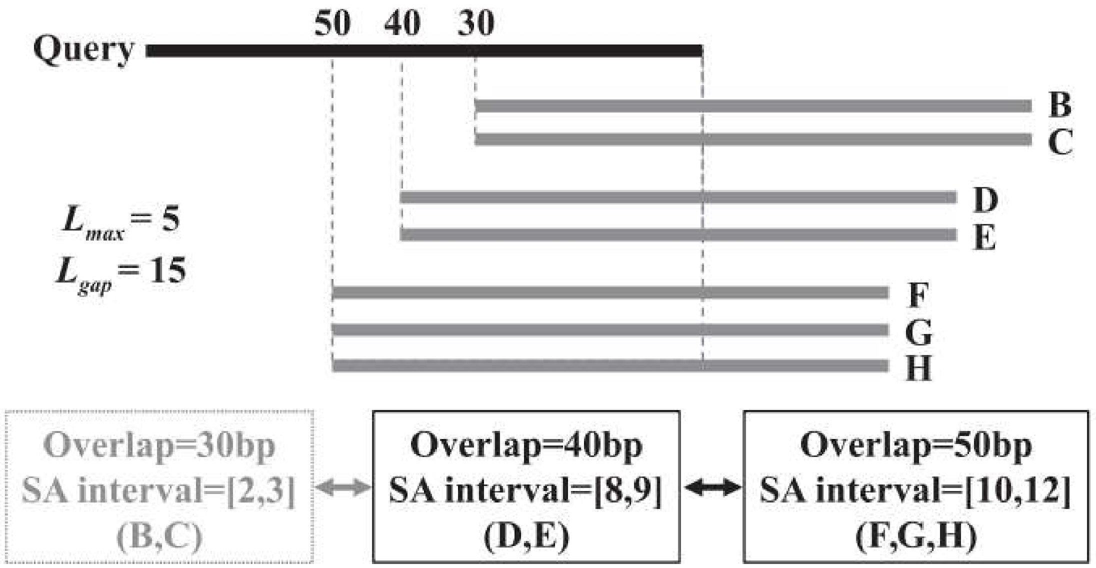
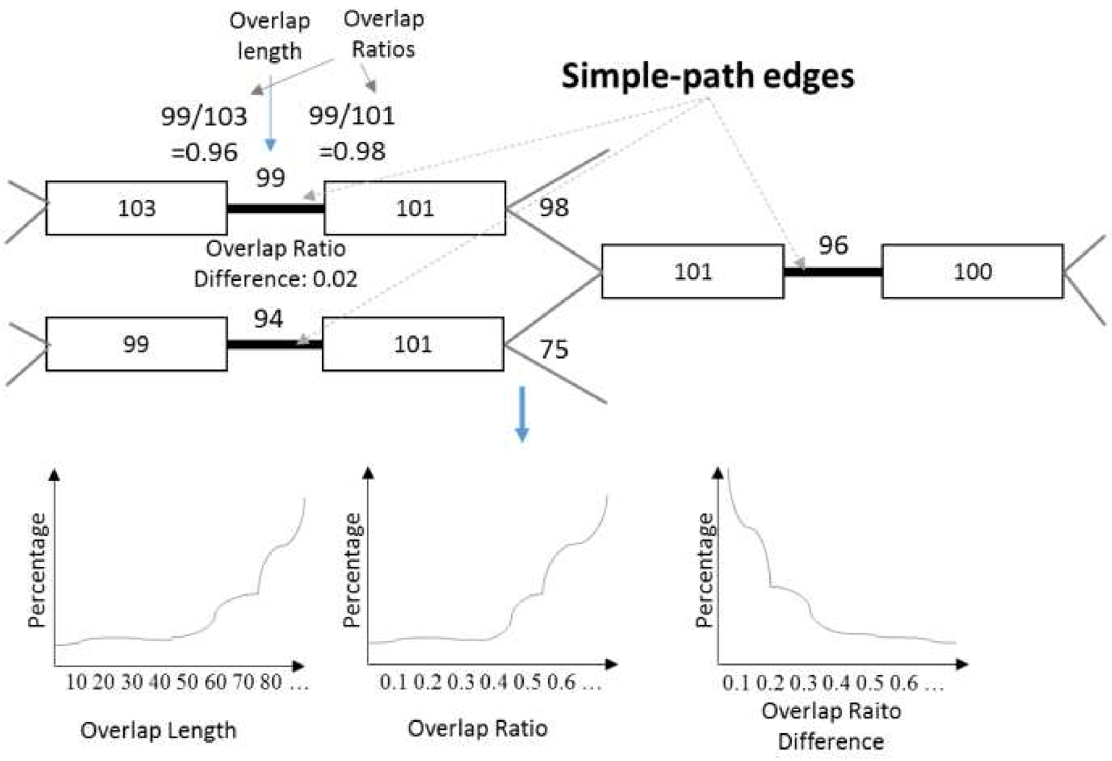


Fig. 6. Collection and pruning of a suffix array (SA) interval list during overlap computation. In this example, only the top five SA indices (set by L_{max}) are retained, which are from the top two SA intervals ([10,12] and [8,9]). That is, only the overlap to reads D, E, F, G and H are retained

Graph layout

- The layout algorithm has to find unique paths of vertices in the graph to maximize contiguity and accuracy of the assembly
- The major challenge is distinguishing true overlapping edges from chimeric/random edges for vertices with two or more ambiguous edges
- Features used: overlap lengths, overlap ratio (overlap to read length), overlap ratio differences on the same edge
- Statistics from simple-path edges is used to get the 95% confidence levels for these features



Supplementary Figure S7. Collection of overlap lengths, overlap ratios, and overlap ratio differences from the initial simple paths. The simple path (or unique path) refers to edges of two vertices with unique overlap.

- Removal of chimeric vertices:
  - Chimeric vertices have shorter overlaps
  - The lengths of the vertices are smaller (don’t merge into larger sequences)
  - K-mer frequency is lower

  Removed: small (≤read length) and low frequency (≤3) vertices with short (>95% confidence) overlaps to neighbours
- Removal of edges with low overlap ratio
  - Due to extended and decomposed reads the overlap length isn’t sufficient for distinguishing true overlapping reads
- Removal of edges with large overlap differences
  - Two neighbouring vertices may have very different sizes
- Tip/Bubble removal after every step

Results

- 12 datasets of genome sequencing of different bacteria using Illumina HiSeq and MiSeq platforms from GAGE-B
- Compared to ABySS, CABOG, MaSurCa, SOAPdenovo, SGA, SPAdes and Velvet
- K=31 was used
- The percentage of extender reads can be > 70% for high quality reads
- For low quality reads the decomposed read percentage can go to 90%
- Medians: 52%, 13%, 25%

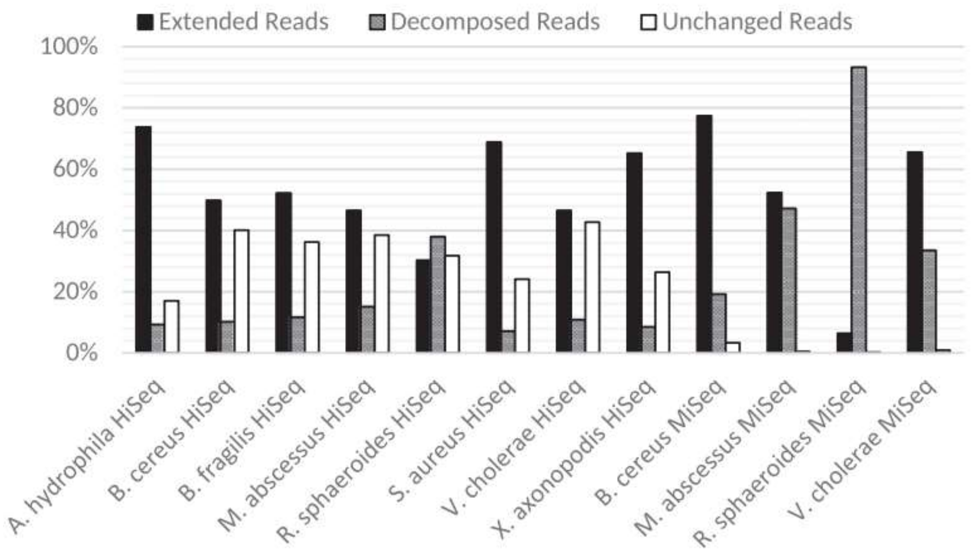
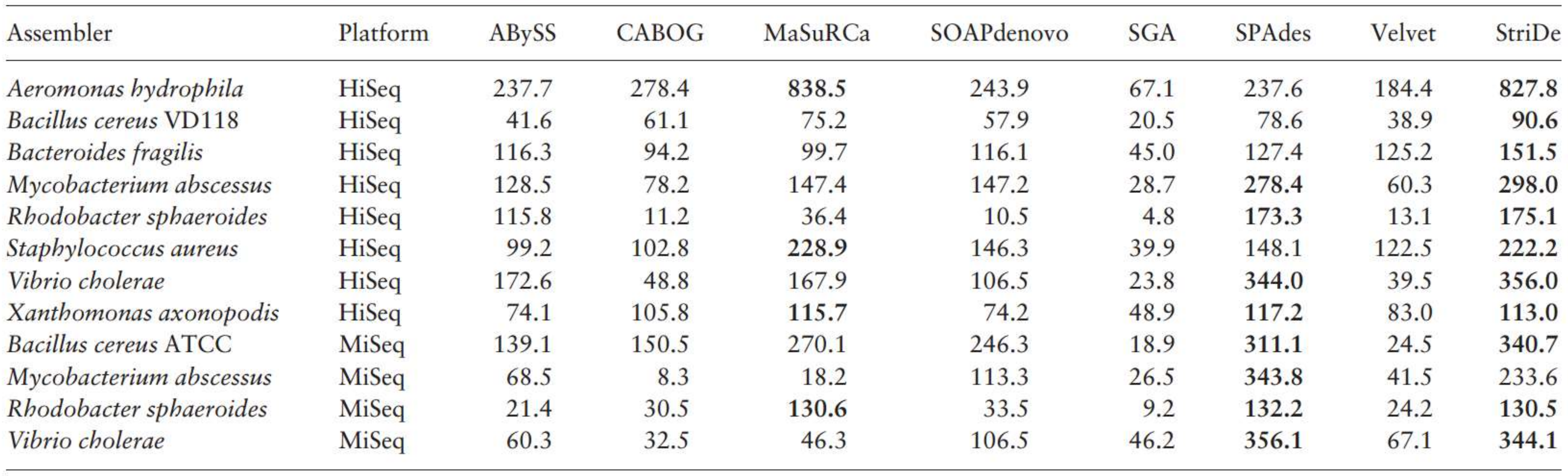


Fig. 10. The percentages of extended, decomposed and unchanged reads across the 12 datasets. The percentages of decomposed reads can be as high as 90% and as low as 10% owing to various sequencing quality

- StriDe assembler outperforms most assemblers in terms of contiguity

Table 1. Assembly contiguity (N50 in kb) of eight short-read (HiSeq) and four long-read (MiSeq) datasets from GAGE-b



Assembler	Platform	ABySS	CABOG	MaSuRCa	SOAPdenovo	SGA	SPAdes	Velvet	StriDe
Aeromonas hydrophila	HiSeq	237.7	278.4	**838.5**	243.9	67.1	237.6	184.4	**827.8**
Bacillus cereus VD118	HiSeq	41.6	61.1	75.2	57.9	20.5	78.6	38.9	**90.6**
Bacteroides fragilis	HiSeq	116.3	94.2	99.7	116.1	45.0	127.4	125.2	**151.5**
Mycobacterium abscessus	HiSeq	128.5	78.2	147.4	147.2	28.7	**278.4**	60.3	**298.0**
Rhodobacter sphaeroides	HiSeq	115.8	11.2	36.4	10.5	4.8	**173.3**	13.1	**175.1**
Staphylococcus aureus	HiSeq	99.2	102.8	**228.9**	146.3	39.9	148.1	122.5	**222.2**
Vibrio cholerae	HiSeq	172.6	48.8	167.9	106.5	23.8	**344.0**	39.5	**356.0**
Xanthomonas axonopodis	HiSeq	74.1	105.8	**115.7**	74.2	48.9	**117.2**	83.0	**113.0**
Bacillus cereus ATCC	MiSeq	139.1	150.5	270.1	246.3	18.9	**311.1**	24.5	**340.7**
Mycobacterium abscessus	MiSeq	68.5	8.3	18.2	113.3	26.5	**343.8**	41.5	233.6
Rhodobacter sphaeroides	MiSeq	21.4	30.5	**130.6**	33.5	9.2	**132.2**	24.2	**130.5**
Vibrio cholerae	MiSeq	60.3	32.5	46.3	106.5	46.2	**356.1**	67.1	**344.1**

Table 2. The number of misassemblies by each assembler on short-read (HiSeq) or long-read (MiSeq) datasets

Assembler	Platform	ABySS	CABOG	MaSuRCa	SGA	SOAPdenovo	SPAdes	Velvet	StriDe
B.cereus VD118*	HiSeq	1 (140.14)	0 (281.24)	3 (235.23)	0 (97.50)	1 (127.95)	1 (115.88)	0 (108.65)	1 (115.44)
M.abscessus	HiSeq	3 (0.98)	7 (5.81)	7 (3.58)	1 (0.43)	9 (0.67)	5 (0.48)	4 (0.74)	5 (0.43)
R.sphaeroides	HiSeq	9 (5.37)	4 (1.55)	5 (2.13)	1 (0.30)	2 (1.94)	2 (1.19)	2 (0.31)	3 (0.70)
V.cholerae	HiSeq	3 (3.98)	20 (6.84)	16 (6.00)	3 (2.60)	21 (3.67)	7 (3.03)	5 (3.37)	3 (2.85)
B.cereus ATCC	MiSeq	6 (4.60)	4 (2.37)	8 (2.13)	5 (2.60)	2 (2.13)	0 (2.72)	3 (2.64)	1 (2.16)
M.abscessus	MiSeq	2 (0.44)	122 (0.74)	70 (0.47)	7 (0.33)	5 (0.69)	6 (0.40)	72 (0.56)	5 (0.49)
R.sphaeroides	MiSeq	12 (4.71)	6 (0.46)	12 (1.56)	2 (0.40)	1 (0.40)	3 (1.20)	2 (0.62)	3 (0.39)
V.cholerae	MiSeq	2 (2.64)	17 (3.35)	24 (3.47)	3 (2.75)	16 (3.25)	8 (2.83)	14 (2.80)	5 (2.69)

The misassembled-indel rate (per 100kb) is shown in parentheses

Time and memory usage

- 20-40 minutes for 12 GAGE-B datasets
- 1-2 GB RAM
- FM-indices: 27-237 MB
- Graph: 6-40 MB
- Run in a server with 48 cores and 256 GB of RAM

Conclusion

- A new assembler StriDe uses best features of string graph and de Brujin graphs
- A set of improvements are introduced
- Main properties are decomposition of reads to subreads in error-prone regions and extension of paired-end reads to longer reads for repetitive regions
- The results are comparable to other methods for both long and short read data

Thank you!