Omega: an Overlap-graph *de novo* Assembler for Metagenomics

Bahlel Haider, Tae-Hyuk Ahn, Brian Bushnell,
Juanjuan Chai, Alex Copeland, Chongle Pan

Bioinformatics, vol.30 no.19 2014
Pages 2717-2722

Fanny-Dhelia Pajuste
JC 03.12.2014

Motivation

THE AIM:

Reconstructing genomes of microorganisms in a community directly from environmental samples.

Assembling and scaffolding Illumina sequencing data of microbial communities.

Introduction – Isolate Genome Assembly

- There are many isolate genome assemblers
- Using de Brujin or overlap graphs

De Brujin graphs	Overlap graphs
ABySS	SGA
IDBA	PEGASUS
ALLPATH	
Velvet	
SOAPdenovo	

- These assemblers cannot be directly applied to metagenome assembly

Introduction – Isolate vs Metagenome

- Assuming a uniform coverage depth across a genome
  - Identifying repeat regions
  - Estimating the size of a genome
  - In metagenome, genomes have different coverage depths depending on their relative abundance
- Repeat regions in a single genome vs between multiple genomes
- Sequencing errors
  - Introduce false overlaps
  - Disrupt true overlaps
  - Error correction using consesus sequence for isolate genomes

Introduction – Metagenome Assembly

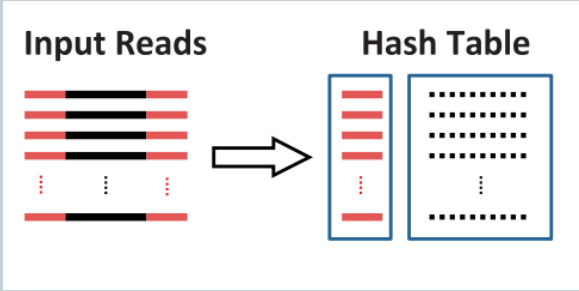
- Some de Brujin graph assemblers have been upgraded for metagenomics
  - MetaVelvet
  - IDBA-UD
- Omega – Overlap-graph MEtaGenome Assembler
- Developed specifically for metagenome assembly
  - Using general overlap graph described in BOA and PEGASUS
- Compared with SOAPdenovo, IDBA-UD, MetaVelvet, Celera

Omega

- Developed in C++
- Developed using object-oriented programming
- Accepts multiple input datasets
  - Different insert sizes
  - Different read lengths
  - Fasta or fastq
- Source code and binaries available from http://omega.omicsbio.org

Algorithm – Hash Table Construction (1)

All unique reads are loaded to the memory and indexed in a hash table



- *K* – userdefined minimum overlap
- Each read is assigned to the hash table with four keys:
  - Prefix of length *K*-1
  - Suffix of length *K*-1
  - Prefix of length *K*-1 of reverse complement
  - Suffix of length *K*-1 of reverse complement
- The value in the hash table is an array of pointers to the reads associated with these keys

Algorithm – Hash Table Construction (2)

- Initialised as eight times of the total read number
- Hash collision is resolved using linear probing
- Nearly constant search time by prefix or suffix

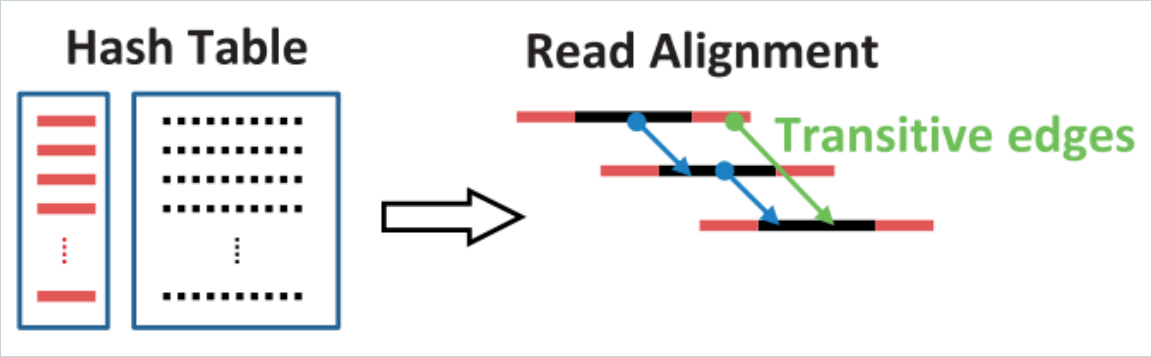
- *Contained read* – read that is a substring of another read
- Used for:
  - Coverage depth calculation
  - Mate-pair linkage analysis

Algorithm – Overlap Graph Construction (1)

- Each read is represented by a vertex in a bi-directed overlap graph
- Edge is inserted between vertices if the reads have an exact-match overlap of at least *K* bp
- Bi-directed edges from overlaps:
  - Suffix with prefix (•→——→•)
  - Suffix of the reverse complement with prefix (•←——→•)
  - Suffix with prefix of the reverse complement (•→——←•)
  - Suffix of the reverse complement with prefix of the reverse complement (•←——←•)

Algorithm – Overlap Graph Construction (2)

- Every substring of length *K*-1 of a read is searched
- The reads are compared for their remaining overlap
- Edge is created if there is exact match
- All transitive edges are removed
  - Using linear algorithm as described in (Haider, 2012; Myeers, 2005)



Algorithm – Composite Edge Contraction

- Edges can be traversed in both directions
- Vertices can be traversed if the in-going and out-going arrows have the same direction
  - (→•→—)
  - (—←•←)
- A valid path in the graph represents an assembled DNA sequence
- Simple vertices: exactly one in-arrow and one out-arrow
- Merged into composite edges

Algorithm – Sequence Variation Removal

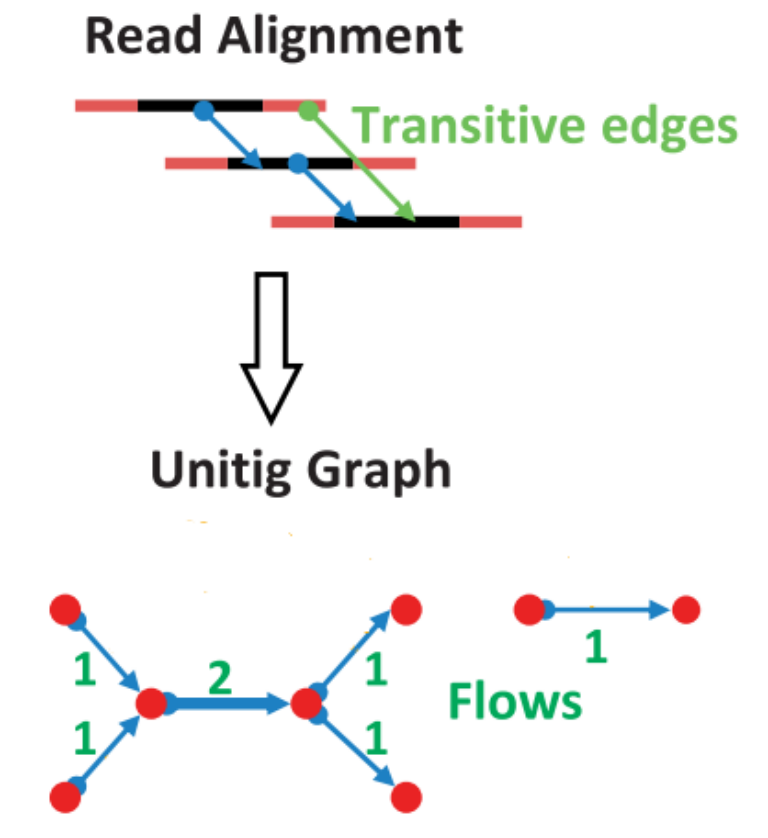
- Sequence variations
  - Uncorrected sequencing errors
  - Natural sequence polymorphisms
- Do not overlap
  - Isolated vertices
- Same sequence variations
  - Small branches (dead-end path with <10 reads)
  - Bubbles (two edges connecting the same two vertices with the same arrow types)
- Removed

Algorithm – Minimum Cost Flow Analysis (1)

- String copy number to each edge
- Represents how many times the sequence is present in the metagenome
- Uses minimum cost flow analysis (Haider, 2012; Myers, 2005)
- Composite edges with sequence >1000 bp – min flow 1
- Shorter edges – min flow 0

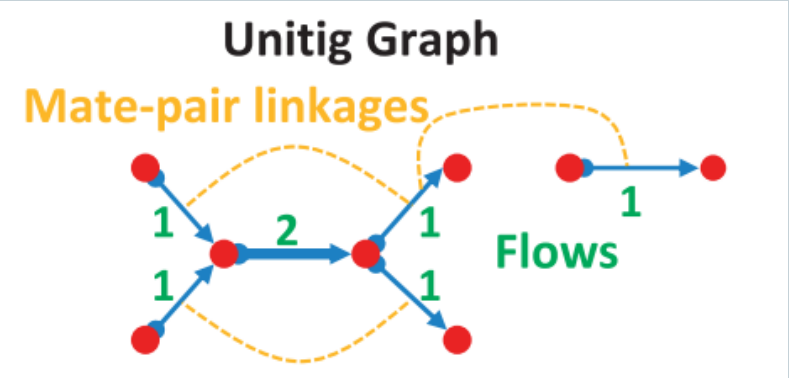
Algorithm – Minimum Cost Flow Analysis (2)

- CS2 algorithm for optimizing the total cost of the flow network in the overlap graph
- Flow > 1 – repeat regions
- Flow = 0 – short sequences (ignored)
- Simplifying tree structures



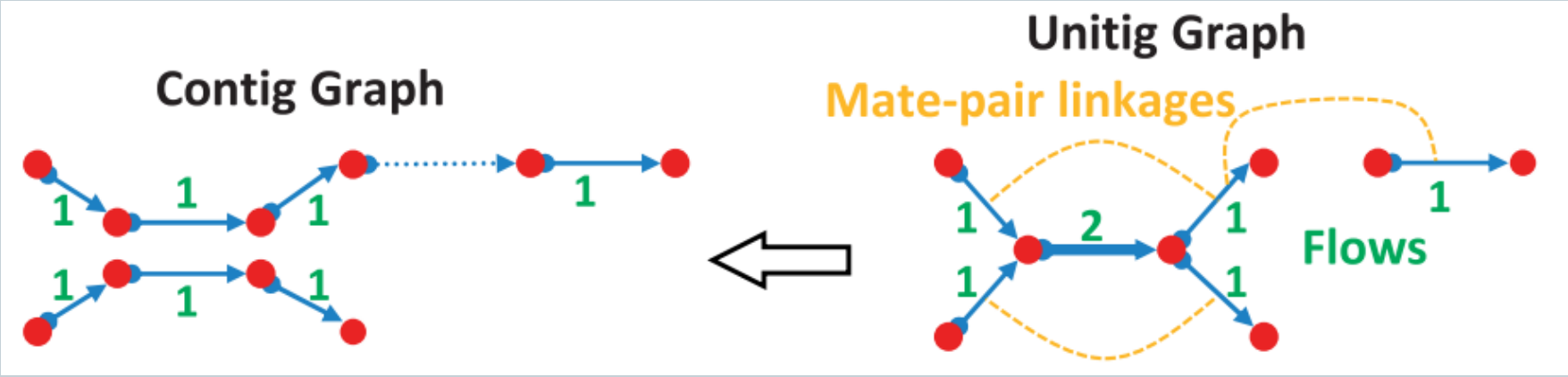
Algorithm – Merging Adjacent Edges

- Estimating insert size of each paired-end dataset
- Mate-pairs in long edges are used to estimate mean and standard deviation of insert size
- Mate-pairs from different edges are used to merge them
  - Finding paths within range ($\mu - 3\sigma$, $\mu + 3\sigma$)
  - All paths of more than three mate-pairs have to travel through the same two edges



Algorithm – Scaffolding Long Edges

- Mate-pairs that have no valid path between reads
- Attempted for every pair of non-adjacent edges with >1000 bp
- Uniquely mapped to two edges with appropritate distance
- More than three mate-pairs have to support the edges

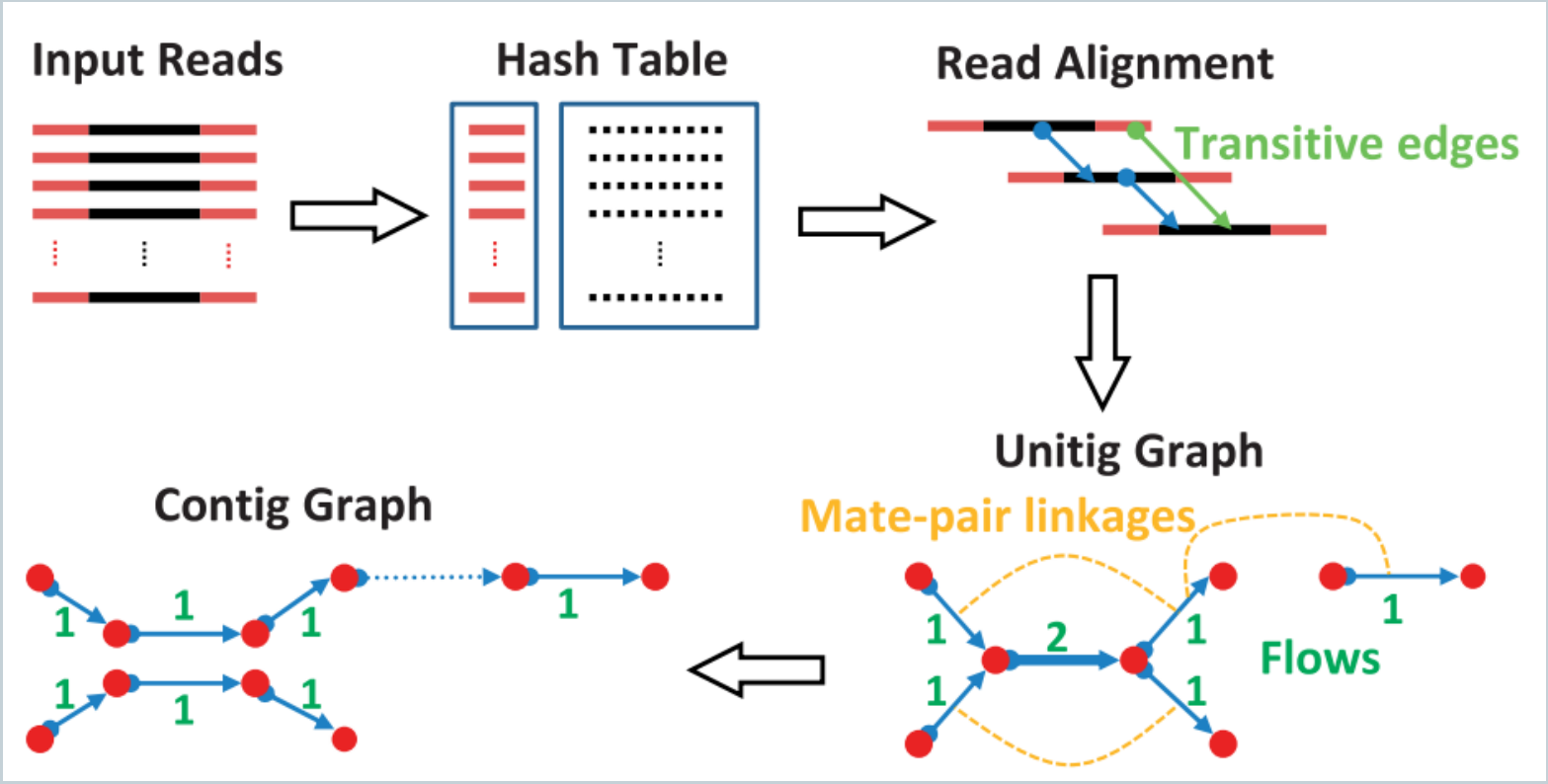


Algorithm – Resolving ambiguity

- Vertices with two in-coming and out-going edges
- Short repeat region between two genomes
- Coverage depth is calculated using unique reads
- Mean δ and SD θ of coverage depth are estimated
- Adjacent edges are merged if $| \delta_1 - \delta_2 | < \theta_1 + \theta_2$

Algorithm

- Reporting contigs and scaffolds based on the edges of the overlap graph



Data

- Illumina HiSeq 100-bp reads
- Real-world sequencing dataset
- Genomic DNA mixture of 64 diverse bacterial and archaeal microorganisms
- 108.7 million paired-end + 0.4 million single-end reads

- Illumina MiSeq 300-bp reads
- Simulated dataset
- Nine-genome synthetic community
- 10 million paired-end reads (avg insert size 900 bp)

Methods – 100 bp

- SOAPdenovo
  - Best *k*-mer length 51
  - Configuration described (Pop, 2011)
- IDBA-UD
  - *K*-mer length range 30-60, step size 10
- MetaVelvet
  - Best *k*-mer length 51
- Omega
  - Best minimum overlap length 50

Methods – 300 bp

- SOAPdenovo
  - Best *k*-mer length 121
- IDBA-UD
  - *K*-mer length range 40-120, step size 20
- MetaVelvet
  - Best *k*-mer length 151
  - (Maximum *k*-mer length changed to 171)
- Omega
  - Best minimum overlap length 150
- Celera
  - Default setting

Methods – Comparison

- Contigs and scaffolds >200 bp were aligned to reference
- List of correct contigs (<5% substitutions and indels)
- Scaffolding was correct if:
  - Contigs were in correct orientation
  - Apart by less than mean + 1 SD of the mate inner distances
- Performance of the methods was compared by:
  - Genome sequence coverage for each reference
  - Largest contig length
  - N80, N50, N20

Results – 100 bp, Data

- Trimming, filtering $\rightarrow$ 101 million paired-end reads
- Aligned to references using Bowtie2
  - Up to three mismatches per read
- Sequencing errors – mismatches supported by less than three reads
- Before error correction:
  93.8 million reads aligned, 0.12 sequencing error per read
- After error correction:
  97.5 million reads aligned, 0.02 sequencing error per read

Results – 100 bp, CPU and Memory (1)

Method	CPU usage	Peak memory usage
SOAPdenovo	13 h	29 GB
IDBA-UD	49 h	112 GB
MetaVelvet	8 h	21 GB
Omega	15 h	105 GB

Results – 100 bp, CPU & Memory (2)

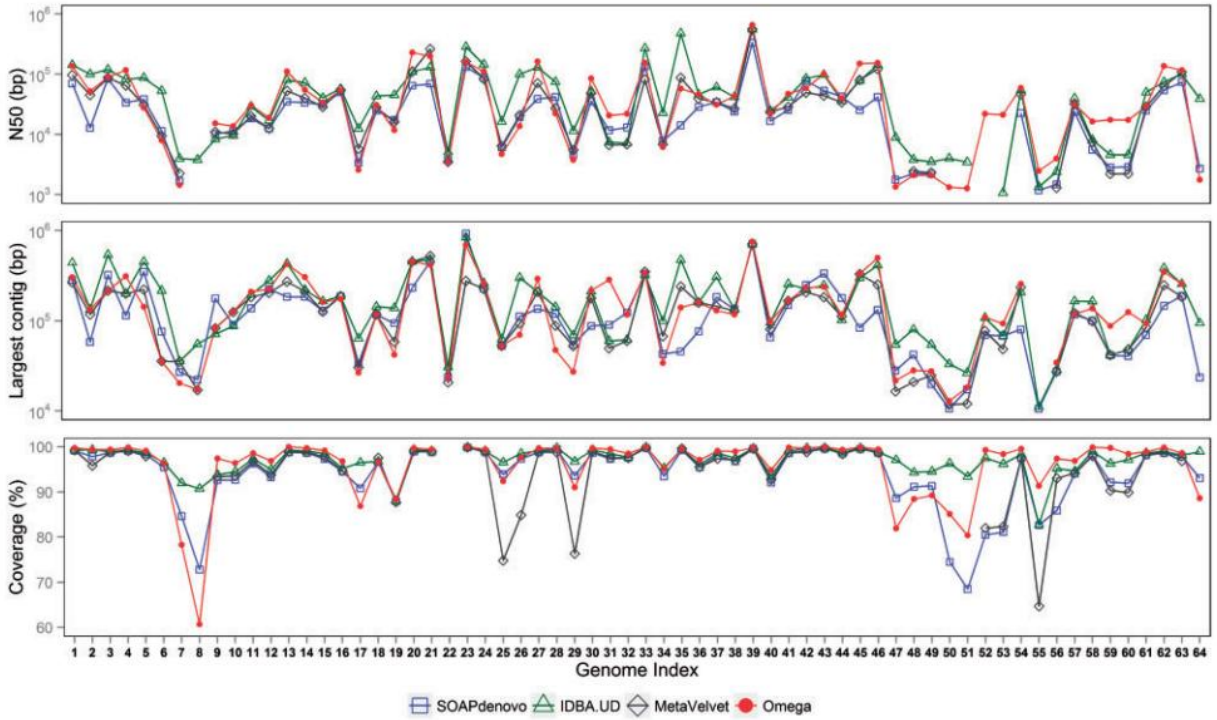
- Omega:
  - 1.5 h – building hash table
  - 2.7 h – indentifying contained reads
  - 7.1 h – constructing the overlap graph
  - 3.5 h - simplifying the graph
- Peak memory usage:
  - At the end of overlap graph construction
  - Reads (50 GB) + hash table (5 GB) + overlap graph (50 GB)

Results – 100 bp (1)

Table 1. Average genome assembly statistics across all genomes in the HiSeq 100-bp dataset

Assembler	N80 (10^3bp)	N50 (10^3bp)	N20 (10^3bp)	Largest Contig (10^3bp)	Coverage (%)
SOAPdenovo	11	33	73	144	92.81
MetaVelvet	17	46	92	147	82.10
Omega	25	61	111	174	94.50
IDBA_UD	35	70	136	203	95.65

Results – 100 bp (2)



1: *Acidobacterium capsulatum* ATCC 51196
2: *Aciduliprofundum boonei* T469
3: *Akkermansia muciniphila* ATCC BAA 835
4: *Archaeoglobus fulgidus* DSM 4304
5: *Bacteroides thetaiotaomicron* VPI 5482
6: *Bacteroides vulgatus* ATCC 8482
7: *Bordetella bronchiseptica* RB50
8: *Burkholderia xenovorans* LB400
9: *Caldicellulosiruptor bescii* DSM 6725
10: *Caldicellulosiruptor saccharolyticus* DSM 8903
11: *Chlorobium limicola* DSM 245
12: *Chlorobium phaeobacteroides* DSM 266
13: *Chlorobium phaeovibrioides* DSM 265
14: *Chlorobium tepidum* TLS
15: *Chloroflexus aurantiacus* J 10 fl
16: *Clostridium thermocellum* ATCC 27405
17: *Deinococcus radiodurans* R1
18: *Desulfovibrio piger* ATCC 29098
19: *Desulfovibrio vulgaris* DP4
20: *Dictyoglomus turgidum* DSM 6724
21: *Enterococcus faecalis* OG1RF
22: *Fusobacterium nucleatum* ATCC 25586
23: *Gemmatimonas aurantiaca* T 27
24: *Geobacter sulfurreducens* PCA
25: *Haloferax volcanii* DS2
26: *Herpetosiphon aurantiacus* DSM 785
27: *Hydrogenobaculum* Y04AAS1
28: *Ignicoccus hospitalis* KIN4 I
29: *Leptothrix cholodnii* SP 6
30: *Methanocaldococcus jannaschii* DSM 2661
31: *Methanococcus maripaludis* C5
32: *Methanococcus maripaludis* S2
33: *Methanopyrus kandleri* AV19
34: *Methanosarcina acetivorans* C2A
35: *Nanoarchaeum equitans* Kin4 M
36: *Nitrosomonas europaea* ATCC 19718
37: *Nostoc* PCC 7120
38: *Pelodictyon phaeoclathratiforme* BU 1
39: *Persephonella marina* EX H1
40: *Porphyromonas gingivalis* ATCC 33277
41: *Pyrobaculum aerophilum* IM2
42: *Pyrobaculum arsenaticum* DSM 13514
43: *Pyrobaculum calidifontis* JCM 11548
44: *Pyrococcus furiosus* DSM 3638
45: *Pyrococcus horikoshii* OT3
46: *Rhodopirellula baltica* SH 1
47: *Ruegeria pomeroyi* DSS 3
48: *Salinispora arenicola* CNS 205
49: *Salinispora tropica* CNB 440
50: *Shewanella baltica* OS185
51: *Shewanella baltica* OS223
52: *Sulfitobacter* EE 36
53: *Sulfitobacter* NAS 14 1
54: *Sulfolobus tokodaii* 7
55: *Sulfurihydrogenibium yellowstonense* SS 5
56: *Sulfurihydrogenibium* YO3AOP1
57: *Thermoanaerobacter pseudethanolicus* ATCC 33223
58: *Thermotoga neapolitana* DSM 4359
59: *Thermotoga petrophila* RKU 1
60: *Thermotoga* RQ2
61: *Thermus thermophilus* HB8
62: *Treponema denticola* ATCC 35405
63: *Wolinella succinogenes* DSM 1740
64: *Zymomonas mobilis* ZM4

Fig. 2. Scaffold assembly statistics for individual genomes from the HiSeq 100-bp dataset. The *x*-axis lists the 64 genomes in alphabetic order, and the indices and organism names of the genomes are shown in the legend. The assemblies were provided by SOAPdenovo (blue squares), IDBA-UD (green triangles), MetaVelvet (black diamonds) and Omega (solid red circles). Outliers below the minimum threshold of a performance statistics are not shown

Results – 300 bp

Table 2. Comparison of overall assembly statistics on the MiSeq 300-bp dataset*

Assembly	Statistics	Assembler	Total contigs	N50 contigs	N80 (10^3bp)	N50 (10^3bp)	N20 (10^3bp)	Largest contig (10^3bp)	Sum (10^6bp)	Coverage (%)
Contigs	Raw	SOAPdenovo	7815	361	8	20	51	358	29	–
		IDBA_UD	1683	72	38	102	217	965	29	–
		MetaVelvet	1097	52	48	136	340	1389	29	–
		Celera	**435**	48	53	151	483	1406	29	–
		Omega	537	**40**	**64**	**159**	**490**	**2572**	29	–
	Verified	SOAPdenovo	7817	361	8	20	51	358	29	97.95
		IDBA_UD	1775	79	36	95	199	547	29	97.73
		MetaVelvet	1104	54	46	135	313	1389	29	98.06
		Celera	**448**	48	52	151	483	1406	29	**99.16**
		Omega	578	**43**	**55**	**158**	**486**	**2091**	29	99.05
Scaffolds	Raw	SOAPdenovo	5586	50	45	138	379	1404	30	–
		IDBA_UD	1570	63	40	116	302	965	29	–
		MetaVelvet	996	44	51	156	489	1389	29	–
		Celera	**429**	48	53	151	483	1406	29	–
		Omega	434	**36**	**67**	**188**	**556**	**2572**	29	–
	Verified	SOAPdenovo	6926	160	11	38	149	593	29	95.11
		IDBA_UD	1733	75	37	100	213	547	29	97.84
		MetaVelvet	1065	50	48	138	479	1389	29	98.03
		Celera	**450**	48	52	151	483	1406	29	**99.16**
		Omega	562	**42**	**55**	**160**	**486**	**2091**	29	99.02

*Best assembly statistics in each category is highlighted in bold.

Discussion

- Computationally simulated data cannot reproduce many complications of Illumina sequencing
- Omega was comparable on 100 bp dataset
- Omega was superior on 300 bp dataset
- Suitable for different data:
  - Omega – long reads, high coverage
  - MetaVelvet, SOAPdenovo – efficient in memory and CPU usage
  - IDBA-UD – better assembly for more genomes and shorter reads
  - Celera – good for long reads

Summary

- Overlap graph is effective
- More useful for future Illumina technologies with longer reads and higher throughput

Thank you for your attention!