MetaVelvet – A soft touch for many little genomes

Stuart Bradley

Toshiaki, N., Hachiya, T., Tanaka, H., Sakakibara, Y. (2012). MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic Acids Research, 40 (20), 155 – 167.

Three Problems with Metagenomic Assembly

1. Short length of sequence reads produced by NGS.
    1. Polymorphism between closely related members of the community.
2. Number of strains is unknown.
3. Relative abundance of strains is unknown.
    1. Highly abundant species are either seen as:
        1. Errors – and removed.
        2. Repeats.

A Refresher Course in de Brujin Graphs

Given two reads:

TGGT GTCA

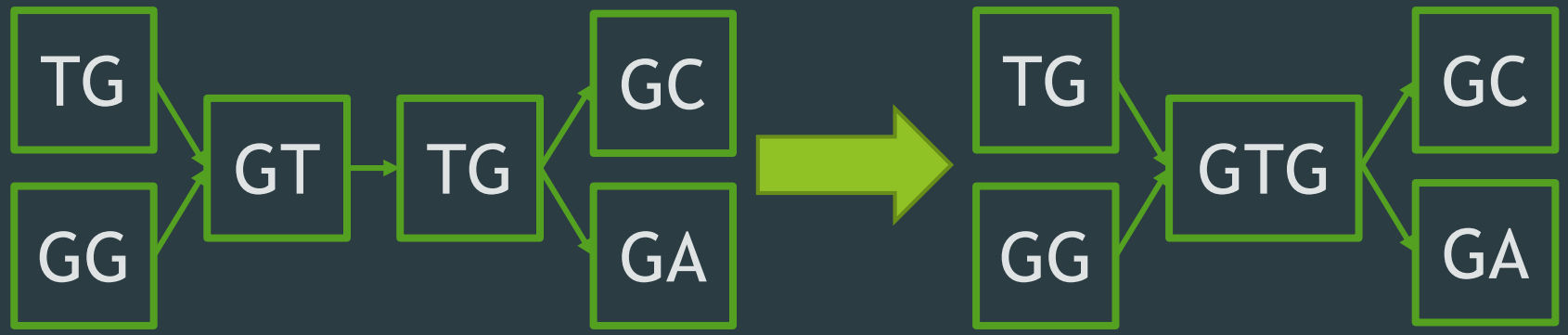
Pick a k-mer length (3) and produce all 3-mers from the reads, then split them into k-1-mers that **overlap**:

TGGT	**GTCA**
TGG -> TG, GG	GTC -> GT, TC
GGT -> GG, GT	TCA -> TC, CA

Create a graph structure of the 2-mers, with each pair joined by an edge.

How Velvet Does it Differently

- **Simplification** – Two nodes with only two edges become one:



- **Tip removal** – Chains of nodes with hang off the end of the graph are removed.
- **Bubble removal** – Two almost identical sets of nodes are merged (or one is removed).

The MetaVelvet Method – Coverage Peaks

Once the primary de Brujin graph has been constructed by the standard Velvet algorithm, it must be decomposed into species graphs.

This is done by looking for peaks in the **k-mer coverage** of the sequences. Different k-mer concentrations suggest distinct species.

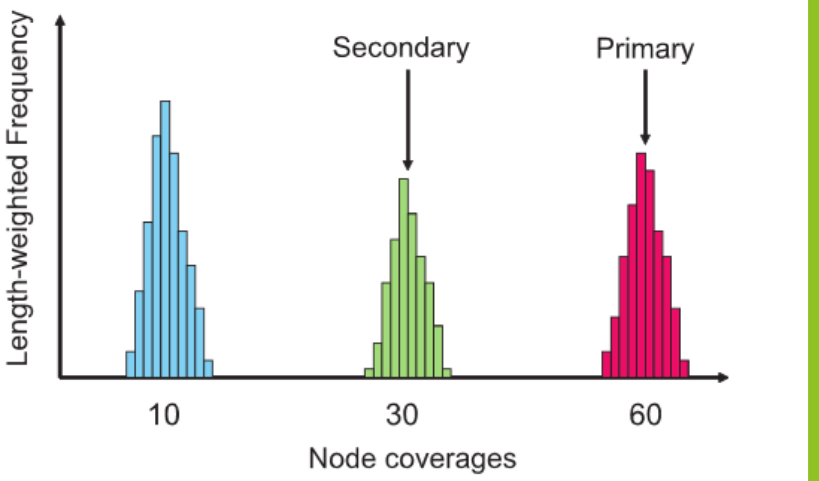


Figure 5. Detection of multiple peaks in the histogram of coverage values of the nodes.

The MetaVelvet Method – Chimera Nodes

Nodes that have one incoming edge and one outgoing edge for two different coverages are termed Chimeric nodes.

They differ from repeats by having two pairs of incoming-outgoing coverage.

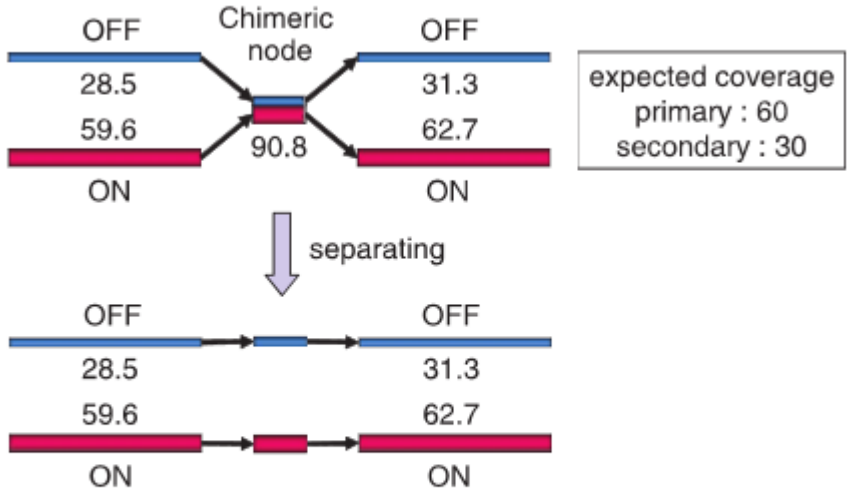


Figure 4. An example of a chimeric node and its resolution by separating the node. The node is chimeric because $[(28.5+59.6)+(31.3+62.7)]/2 \simeq 90.8$.

The MetaVelvet Method – Final Steps

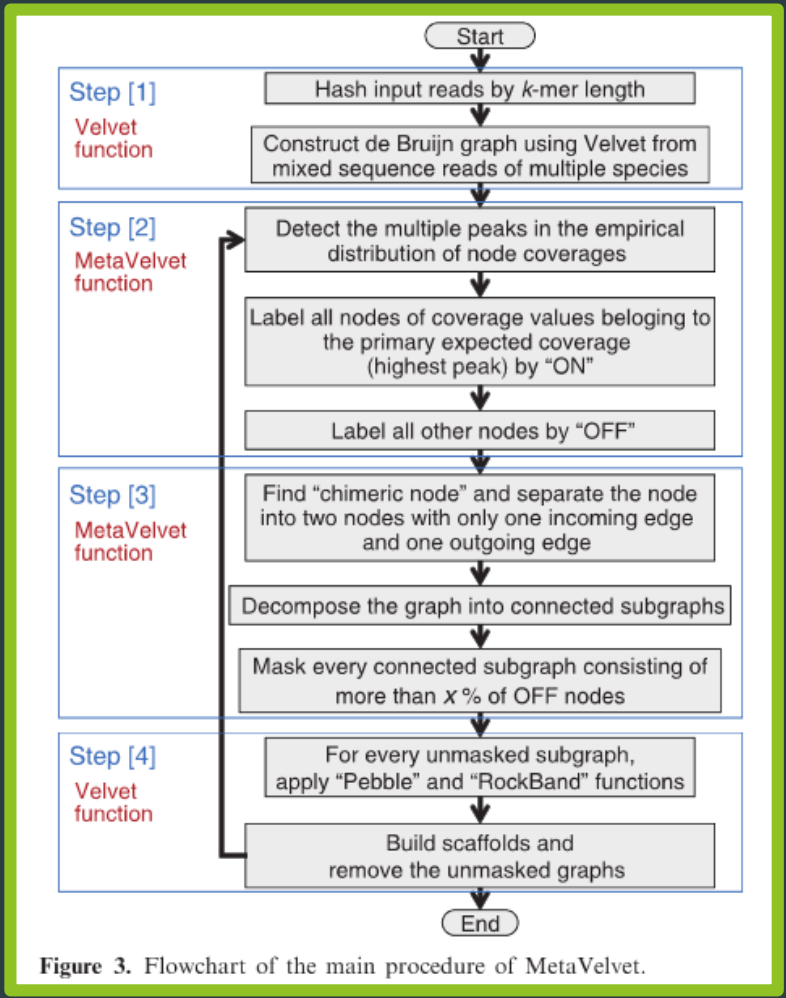


Figure 3. Flowchart of the main procedure of MetaVelvet.

Velvet functions are applied to the primary nodes, and this (first species) assembly is then removed from the graph.

The whole process is repeated until there are no more nodes left in the parent graph.

Performance Comparison - General

Method	Num. scaffolds	Cover Rate	N50	Gene Prediction	CPU	RAM
MetaVelvet						
Velvet						
SOAPdenovo						
Meta-IDBA	**Num. Contigs**					

Reads were generated using the DNAA package[1], with 80bp read length and a log-normal distribution of species abundance.

[1](http://sourceforge.net/projects/dnaa)

Performance Comparison – Chimeric Algorithm

To test the optimality of the MetaVelvet algorithm, the genus-level dataset was analyzed. This showed that 750/770 chimeric sites identified were correct.

The main stated reason for misidentification was statistical variance in k-mer coverage.

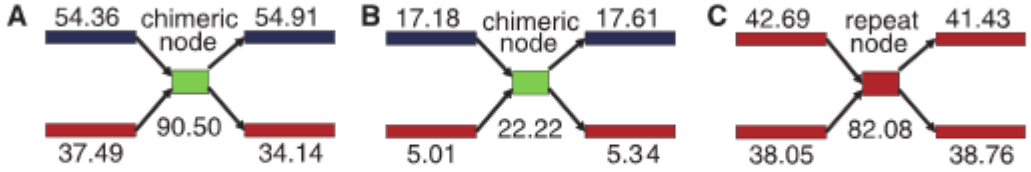


Figure 9. (**A** and **B**) Two examples of chimeric nodes correctly identified by MetaVelvet. (**C**) Example of a repeat node misidentified by MetaVelvet as chimeric.

Performance Comparison – Species Level

At the species taxa level, MetaVelvet underperforms when compared to Meta-IDBA.

This is because Meta-IDBA was designed to deal with species polymorphisms, something that MetaVelvet was not.

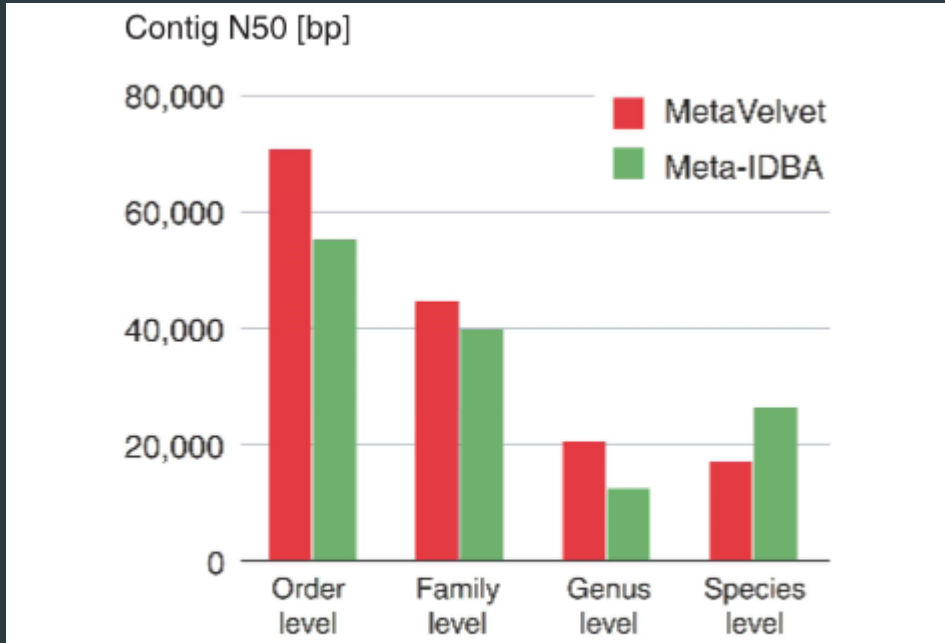


Figure 13. N50 scores for contigs generated by MetaVelvet and Meta-IDBA.

Conclusion

MetaVelvet – in general – is a fairly good *de novo* metagenome assembler.

However, 2.5% chimeric mis-identification could possibly be reduced by a more complex method of identification. Which would result in a more robust system.

Topological structure partioning could also be integrated into the program, so it could better compete with Meta-IDBA at the species level.

Appendix 1 – Performance Comparison

Table 1. Performance comparison of assembly software packages

Metagenome dataset	Separate assembly	MetaGenomic assembly			
	Velvet	MetaVelvet	Velvet	SOAPdenovo	Meta-IDBA
Order level (total genome size = 71 929 175 bp; 93 423 332 reads)					
Num. scaffolds	685	813	924	5 998	2 678
Total scaffold length	71 009 045	71 053 228	48 450 203	70 296 665	70 312 381
N50 size (bp)	288 838	268 350	142 471	43 796	55 575
Chimeric scaffold length (%)	0.00	0.00	0.46	0.00	0.00
Cover rate (%)	98.38	98.25	67.48	95.67	96.98
Number of predicted genes	66 268	66 241	43 729	60 319	62 833
Required CPU time (s)	4 994	8 685	7 076	11 564	7 375
Required memory (GB)	7.04	56.61	54.07	62.42	15.15
Family level (total genome size = 84 552 832 bp; 113 680 114 reads)					
Num. scaffolds	784	1 019	2 889	9 039	4 421
Total scaffold length	83 275 357	83 322 440	65 789 192	81 739 588	81 990 799
N50 size (bp)	313 454	257 853	76 239	27 510	39 961
Chimeric scaffold length (%)	0.00	0.45	0.02	0.03	0.00
Cover rate (%)	98.12	97.81	77.60	94.09	96.03
Number of predicted genes	77 634	77 655	58 744	68 832	72 746
Required CPU time (s)	9585	11 409	9 813	14 803	12 664
Required memory (GB)	13.15	72.06	68.98	62.48	23.11
Genus level (total genome size = 88 595 850 bp; 103 990 387 reads)					
Num. scaffolds	1288	2 325	3 633	10 282	10 643
Total scaffold length	86 489 808	84 342 495	53 450 902	79 334 848	74 808 521
N50 size (bp)	279 359	239 061	74 182	16 194	12 773
Chimeric scaffold length (%)	0.00	1.56	0.00	0.08	0.00
Cover rate (%)	98.17	97.13	73.31	91.73	90.93
Number of predicted genes	80 812	79 301	46 688	67 267	61 135
Required CPU time (s)	7275	10 395	8 712	12 889	15 071
Required memory (GB)	11.12	63.22	60.43	62.45	16.75
Species level (total genome size = 85 450 435 bp; 98 817 303 reads)					
Num. scaffolds	818	3 447	2 403	9 317	6 657
Total scaffold length	83 865 679	80 628 784	40 619 181	70 762 160	64 880 992
N50 size (bp)	339 109	152 531	100 819	14 471	26 571
Chimeric scaffold length (%)	0.00	0.93	0.00	0.01	0.00
Cover rate (%)	97.79	94.56	60.29	84.62	82.50
Number of predicted genes	83 952	81 842	38 445	65 176	58 367
Required CPU time (s)	7618	12 001	8 775	12 858	20 755
Required memory (GB)	7.68	64.06	61.23	62.46	17.32

All computations were executed with Intel(R) Xeon(R) E5540 processors (2.53 GHz), with 48 GB physical memory, except for a few cases. The figures in 'separate assembly' show the results of single-genome assembly from pure sequence reads of each single-isolate genome, which were not available in real-data analysis. MetaVelvet, Velvet and SOAPdenovo were run with default parameters, except for setting *k*-mer size at 51. Meta-IDBA was run with default parameters, except for setting the maximum *k*-mer size at 50.

Appendix 2 – Gene Prediction

Table 2. Assembly and gene prediction statistics for human gut microbial metagenomic datasets

	MH0006		MH0012		MH0047	
	MetaVelvet	Velvet	MetaVelvet	Velvet	MetaVelvet	Velvet
Scaffolds						
Num. scaffolds	293 805	174 794	368 879	125 387	69 380	21 833
Total scaffold length (bp)	176534 240	141 464 165	239 717742	166 824 609	39 488 884	26 190 998
AUC of N-len(x)	1 740 532	764 099	5 049 858	1 897 179	280 526	181 960
AUC_{min} of N-len(x)	1 732 914	764 099	5 027 177	1 897 179	277 057	181 960
Protein-coding genes						
Num. genes	421 448	284 552	531 824	257 502	92 411	40 267
Num. complete genes	90 617	84 811	129 670	117 792	18 032	14 680

AUC of N-len(x) denotes the area under the curve of the generalized score N-len(x), which is defined by Eq. (2), for $0 < x \leq L$. AUC_{min} of N-len(x) denotes the area under the curve of N-len(x) for $0 < x \leq \min\{L, L'\}$.