https://doi.org/10.56093/ijas.v90i5.104327

Indian Journal of Agricultural Sciences **90** (5): 847–54, May 2020/Review Article

Algorithmic and computational comparison of metagenome assemblers

ANU SHARMA*, DWIJESH CHANDRA MISHRA, NEERAJ BUDHLAKOTI, ANIL RAI, SHASHI BHUSHAN LAL and SANJEEV KUMAR

ICAR-Indian Agricultural Statistics Research Institute, New Delhi 110 012, India

Received: 24 December 2018; Accepted: 31 October 2019

ABSTRACT

Assembly of genome sequences of a microbial community is computationally challenging and complex than its single genome counterparts. Keeping in view the volume, diversity and varied abundance of different microbes, number of metagenome assemblers have been developed addressing specific associated computational issues mainly following De Bruijn Graph (DBG) and Overlap Layout Consensus (OLC) approaches. It is very pertinent to understand different computational approaches and issues of metagenomic assembly to further improve them with respect to time and computational resource requirements. Therefore, the main objective of this article is to discuss various metagenomics assemblers with respect to their development addressing major computational issues. Initially the computational perspective of single genome assemblers based on OLC and DBG graph construction approaches was described. This is followed by review of metagenomic assemblers with respect to the algorithm implemented for addressing issues in metagenome assembly. Further, performance of some of the popular metagenome assemblers were empirically evaluated with respect to their run time and memory requirements by taking diversified benchmark metagenomics data at ICAR-IASRI, New Delhi in 2019. It was concluded that performance of assemblers varied considerably on these datasets and there is further need to make an effort to develop new tools or to modify the existing ones using efficient algorithms and data structures.

Key words: Assembler, Algorithm, Computation, Metagenome

Metagenomics is emerging as a promising technology for structural and functional profiling of microbial communities. With the advent of new sequencing techniques, the sequencing of the genome of many environmental and human microbial communities have been done as these were difficult to culture in the laboratory. This information overload has given rise to the development of many tools and software for analysis and assembly of metagenomics data. Sufficient literature is available on assemblers which are capable in assembling sequencing data of single genome, but in case of assembling mixed genomic sequences, i.e. metagenomics data of microbial community of an environment have number of computational issues and challenges. Attempts have been made to analyze the assembly challenges and accordingly reviews were published (El-Metwally *et al.* 2013, Ghurye *et al*. 2016). Miller *et al*. (2010) have compared genome assemblers for single genome based on DBG and OLC approaches. A classification of existing assemblers has been provided considering the common features of these along with complete description of layout and DBG graph theory. Kleftogiannis *et al*. (2013) have compared the memory efficiency of single genome assemblers in terms of memory usage, execution time and quality metrics.

It may be noted that the above reviews were confined to the genomic assemblers capable of assembling sequence data of single species. A comprehensive review of metagenomics assemblers was done by Vollmers *et al*. (2017) in terms of biological/microbiologist perspective. Recently in a review, methods and tools for metagenome read classification, metageome assembly tools and contents of existing metagenome databases with quality aspects is done (Breitwieser *et al*. 2017). Therefore, hardly there is any review article which compares metagenomics assemblers from computational and algorithmic perspective. Therefore, in this article an attempt has been made to compare metagenomic assemblers from computational and algorithmic perspective, i.e. (i) variants of DBG and OLC graph construction, (ii) algorithmic approaches and (iii) computational requirements with the help of benchmark datasets. This article will be highly useful to the computational biologists/scientists, scholars and students working in the area of metagenomics and related fields.

Current challenges in assembly of metagenome

Assembly of metagenome consists of three main steps namely (i) sequence cleaning, (ii) fragment assembly and

*Corresponding author e-mail: anu.sharma@icar.gov.in

(iii) Binning/taxonomic assignment. Sequence cleaning deals with the filtering of duplicated, short, low quality, contaminated reads and reads containing ambiguous bases. Fragment assembly of metagenomic sequences is performed following two approaches namely OLC and DBG. *De novo* /Fragment assembly results in the generation of contigs/ scaffolds. Binning/taxonomic assignment is needed to classify/associate these data with the organisms from which they were originated. This process of association between sequence data to their related species is called binning or classification. Existing binning methods can be classified into two categories, namely taxonomy dependent and taxonomy independent. In case of taxonomy dependent methods, the extent of 'similarity' of the reads with sequences (in reference databases) or pre-computed models helps in this assignment process. Further, based on the strategy used for comparing sequence reads with existing sequences/ pre-computed models, taxonomy-dependent methods can be further classified into three categories, i.e. alignment-based, composition-based and hybrid methods (Mande *et al*. 2012). Alignment based algorithms uses existing alignment and read mapping algorithms to align the reads to known and characterized genomes. This approach is used in MG-RAST, CAMERA, MEGAN, MetaPhyler and MARTA etc. Faster methods based on composition features use GC content, codon usage, oligonucleotide usage patterns for comparing reads to reference databases. Under this method, specific models are built for various genomes using various statistical and data mining approaches. Some binning tools based on this method are Phylophthia, TACOA, Phymm, ClaMS and RAIphy etc. Hybrid approaches combine the alignment and composition based methods for classification of metagenomic data. In case of taxonomy independent methods, simply group/bin of sequence reads are made from given dataset based on their mutual similarity and it does not involve any database comparison. Tools following this approach are TETRA (Teeling *et al.* 2004), CompostBin (Chatterji *et al*. 2008), AbundanceBin (Wu *et al*. 2011) and MetaCluster (Leung *et al*. 2011).

Initially, single genome assemblers were used for assembly of metagenomic data but due to computational and biological issues specific to this field, effective and accurate assembly could not be done. Some of the major challenges with assembly of metagenomes are: (i) uncertainty about population size, (ii) composition of species and (iii) noise model of sequencing technology. Also, coverage across species is uneven and affected by the species abundance in the sample. Therefore, an alternative approach was required which may also construct the contigs even for species having low coverage data. It was further observed that the performance of existing DBG based approaches degrade with increase in number of errors in the reads. This demands additional efforts for detecting, correcting and filtering the reads before assembly. Most of existing techniques in this regard implemented in single genome assemblers assumes the low coverage as a criterion to filter out the reads. This is not desirable as well as applicable in case of metagenomics data. Also, identification of repeat region in case of metagenomic data needs to be extended to multiple genomes. Separation of Single Nucleotide Polymorphism (SNPs) is difficult to separate from sequencing errors in metagenome. Due to the above facts, application of single genome assemblers for metagenome assembly leads to the formation of chimeric and shorter contigs. Therefore, number of metagenomic assemblers are developed in due course of time by addressing above issues. In the next section computational developments in DBG and OLC approaches are described briefly.

Computational developments

Assemblers have been developed for assembling metagenomic data generated through various sequencing platforms. The algorithmic approach of different assemblers varies in: (i) ways of handling type of reads (i.e. long reads to short reads), (ii) type of graph construction, (iii) sequencing error correction methods, and (iv) ability to deal with different length of fragments. Mainly, two types of assembling algorithms were developed following OLC and DBG construction approach according to the size of reads. The DBG is a graph data structure suitable for representing the overlap relationship of short read sequences, whereas, OLC is generally used for assembling long read sequences. It was observed that, the overlap graphs work well if there is small number of reads with significant overlap. However, this method is computationally expensive for large genomes. Complexity of pairwise sequence alignment is quadratic in terms of number of reads. Efforts have been made by researchers for improving the computational efficiency of OLC and DBG based approaches.

Developments in computational framework of OLC approach

The first graphical representation of sequence assembly as an overlap graph was introduced by Kececioglu and Myers in 1995. In an overlap graph, a vertex represents a read and an edge represents an overlap. Optimal path through each vertex was used to find the layout of reads to covert to larger fragments of reads. Transitive reduction approach was proposed to reduce the redundancy of the overlaps. In 2000, Celera assembler was developed for best assembly of short-gun reads (Myers *et al*. 2000). A more accurate and efficient assembly program for better handling of overlap reads using multiprocessors was developed in 2003 (Huang *et al*. 2003). Further, an efficient and better tool for detection and classification of single nucleotide polymorphisms for expressed sequence tags using OLC approach was developed by Chevreux *et al*. (2004). Also, Myers in 2005 has proposed a string graph OLC based assembler. In order to apply this approach for short read sequences, FM-index was applied to find overlaps in OLC graph (Simpson and Durbin 2010). This was followed by two faster and efficient overlap finder proposed by Dinh and Rajasekaran (2011) and Gonnella and Kurtz (2012).

Developments in computational framework of DBG approach

DBG based approach is more computationally efficient as compared to OLC approach. The DBG approach also resolves repeat region problem of OLC based approach with polynomial time solution (Pevzner *et al*. 2001, Pevzner and Tang 2001). Further, the problem of repeat region is addressed efficiently through double-barrel data or by additional PCR experiments following this approach. A parallel version of DBG decomposition algorithm was given by Bokhari and Sauer (2005). This algorithm made an attempt to resolve some important issues of genome assembly like reads length, orientation, complementarity and errors in a DNA sequence. Further, the computational efficiency was improved through parallelized DBG implementation in C language. Alternative DBG representation has provided better error elimination and sequence resolution by using more efficient algorithms which was implemented in Velvet assembler (Zerbino and Birney 2008). Velvet was implemented in C and tested on a 64-bit Linux machine. Issue of scaling up the available memory for storing DBG graphs especially for large genomes was addressed by Simpson *et al*. (2009). For reducing the memory requirements, a new distributed DBG structure using hash tables instead of pointers was proposed which was implemented in an assembler for short sequences named as ABySS. In this, sequence data is partitioned and distributed over a number of nodes in Linux Cluster to run it parallel. Message Passing Interface (MPI) protocol was used for communication between the nodes.

In order to further reduce memory requirements, Li *et al*. (2010) proposed an approach of recording minimum-information in DBG instead of read locations and pair-end information. Another effort in this direction of reducing large memory space requirements for DBG construction was undertaken by Ye *et al*. (2011). They proposed sparse DBG structure which requires less memory and space through skipping some intermediate k-mers. This proposed structure also takes care of substitution errors introduced during the sequencing process.

Storing of nodes in a DBG using hash tables and pointers was further modified by Conway and Bromage (2011). They proposed a new storage configuration having more efficient data structure using information theory concepts. In this case, DBG was represented as entropy compressed succinct DBG. This configuration is ten times more memory efficient as compared to the existing methods. Further, Chikhi and Risk (2013) introduced a new, space-efficient representation of the DBG by implicitly encoding this as a Bloom filter. In this case, separate data structure is used to store each of the false and true positive nodes. Besides this, a new data structure was also proposed which stores the information of visited nodes during traversal. Therefore, this structure stores only a subset of k-mers leading to more efficient space utilization. An efficient tool for *de novo* assembly namely Minia was developed by combining the Bloom filter, critical false positives structure and the above marking structure. The memory requirements of the data structure proposed by Chikhi and Rizk (2013) was further reduced up to 40% through using cascading Bloom filters and storing k-mers in less number of bits (Salikhov *et al*. 2014). Many popular assemblies of genome and metagenome requires to construct the DBG graph with multiple values of k. This hampers their performance with the increase in genome size. In order to resolve this problem, generalization to DBG has been proposed by Lin and Pevzner (2014) with the introduction of the manifold de Bruijn (M-Bruijn) graph theory. In this, multiple k-mers sizes are considered in single iteration for selection of optimal k-mer size instead of varying k-mer size in each iteration as in Iterative de bruijn Graph Assembly (IDBA) approach.

Boucher *et al*. (2015) described a method for efficiently representing multiple DBG of different orders in a single succinct data structure. The proposed data structure allows the order of the DBG to be changed on the fly. This technique considerably improves the memory and space requirements of the existing assemblers. Recent technology such as Single Molecule Real Time (SMRT) sequencing platform generates longer but error-prone reads which is being assembled using OLC based approach. Lin *et al*. (2016) have applied generalized DBG approach to assemble these reads to generate more accurate genome reconstruction using A-Bruijn graph. A-Bruijn graph is developed by exploiting the benefits from both DBG and OLC approaches.

A computational perspective of metagenome assemblers

Assembly of metagenomic data poses many challenges which cannot be addressed by existing single genome assemblers. Some of the existing metagenomic assemblers are Genovo, Xgenovo, Meta-IDBA, Ray Meta, IDBA-UD, MetaVelvet, MetaVelvet-SL, MegaHit, MegaHit1.0, PRICE, Omega etc. There is a need to understand the algorithmic approaches of existing assemblers to develop more efficient algorithm to address the computational challenges. Therefore, in this section the important algorithms used by above assemblers are given in brief.

OLC based approaches for metagenome assembly

Genovo is a *de novo* assembler developed by Laserson *et al*. (2011) for assembling metagenome. In order to discover the population structure, sample is initially subjected to Chinese restaurant process to generate prior probabilities to classify reads. This process accounts for the unknown number of genomes in the sample prior to their assembly. Likely assemblies are made by applying a series of hill-climbing algorithms steps iteratively until convergence. This Bayesian based approach offers better sensitivity for assembly in highly diverse environment. Genovo's reconstruction procedures are comparatively more efficient than their competitive procedure which consequently yields a higher assembly score. Genovo was further improved as Xgenovo by incorporating paired-end information to get higher quality assemblies (Afiahayati *et al.* 2013).

Haider *et al*. (2014) proposed an assembler named Omega (Overlap Graph Metagenome Assembler) for

assembling and scaffolding metagenomics data obtained from Illumina sequencing platform at a deeper coverage. In this, hash table is used as data structure to find overlaps between reads. Also, simplification to the overlap graph was done by deleting transitive edges and trimming short branches. Further, mate-pair information is used to get scaffolds after merging contigs. It is reported that Omega outperforms its competitors in terms of computing time with respect to existing OLC based Celera assembler on the MiSeq dataset. Also, it was found that the performance of Omega is at par on a HiSeq 100-bp dataset and superior on a MiSeq 300-bp dataset when compared to DBG based popular assemblers such as SOAPdenovo, IDBA-UD and MetaVelvet.

DBG based approaches for metagenome assembly

The Meta-IDBA algorithm was proposed by Peng *et al*. (2011) for assembling reads of metagenomic data for handling the issue of polymorphism. This assembler is based on the biological observation that the genome of sub species of same species share more common regions than the genome of sub-species from different species. Therefore, the major task is partitioning of the DBG graph is into different components such that each component represents a consensus contig of a species. The process of DBG partition is based on the concept that if, path of a given length from one k-mer can be traversed to another k-mer in DBG starting from each of its out-going edges then two k-mers are assumed to be originated from the same species. Otherwise, the given two k-mers may belong to genomes of different species/sub-species and should be separated. Further, larger components are broken down into the smaller components and each component represents a consensus contig of sub species from a single species. Paired-end information was used by Meta-IDBA to combine the contigs into scaffolds. This was followed by performing multiple alignment of each component with small variants of the genomes of sub-species. Finally, consensus sequence was used to represent genome of one species. Meta-IDBA was shown by Peng *et al*. (2011) to construct longer contigs with accuracy similar to Velvet and Abyss for different metagenomic datasets. One of the major limitation of Meta-IDBA is that it cannot reconstruct the contigs of each single sub-species due to presence of common regions in the genomes of sub-species. Another, drawback is that the accurate separation of components in some case for some species is not possible. Also, in some situations many small components may be obtained after graph partitioning like presence of false positive edges, more number of species (>1000). It may be noted that Meta-IDBA does not depend upon the coverage for the construction of DBG graph, so the change in abundance ratio does not affect its performance. Subsequently, Peng *et al*. (2012) developed an algorithm (IDBA-UD) based on DBG approach considering uneven sequencing depths for metagenomic assembly. In this assembler, instead of using single threshold value for k-mer removal multiple values are used depending on the sequencing depth. Also, the branching problem of low-depth short repeat regions is solved with the help of local assembly using paired-end information. In order to speed up the process, an error correction step is incorporated to correct reads of high-depth regions which can be aligned to high confident contigs.

Metavelvet assembler is the extension of single genome assembler named Velvet for the assembly of genome of various species in microbial communities. It was developed by Namiki *et al*. (2012) for mixed short read sequences of multiple species. The process of assembly in MetaVelvet is done in four phases. In the first phase, it constructs the DBG from the input reads. In second phase, it detects the multiple peaks based on empirical distribution of node coverage assuming that the expected frequencies of k-mer occurrences follow a Poisson distribution in a single-genome assembly and the expected k-mer frequencies in metagenome assembly were shown to follow a mixture of Poisson distributions. MetaVelvet distinguishes a sub-graph composed of nodes belonging to a same peak from the other sub-graphs in the main DBG. In phase three, it identifies shared nodes (chimeric nodes) between two sub-graphs and disconnects two sub-graphs by separating the shared nodes with one in and out degree. MetaVelvet uses the number of incoming and outgoing nodes to classify a chimeric node constructed due to repeats in the same species or due to similar sequence in different species. Lastly in phase four, it builds contigs and scaffolds based on the decomposed subgraphs using Velvet functions.

The major issue in MetaVelvet is the correct identification of chimeric nodes as this prevents generation of longer contigs and scaffolds due to their misclassification among species. In order to address this problem a new tool called MetaVelvet-SL was developed (Afiahayati *et al*. 2015). The performance of this tool is improved by applying supervised machine learning for classification of chimeric node. The first step of MetaVelvet-SL is the construction of DBG followed by identification of chimeric candidate nodes along with their features as a second step. Further, a Support Vector Machine is used for learning the classification model based on features extracted from candidate nodes. It also prepares training sample using MetaPhlAn based on prior knowledge about the taxonomic profile of the target microbial community. This training samples are generated using various profiling methods such as: (i) collection of reference genome of closely related species, (ii) collection of genomes to simulate sequence reads, and (iii) alignment of nodes with reference genome. Finally, species for each node is identified. The major drawback of low accuracy and sensitivity in detecting the chimeric nodes in the DBG generated by MetaVelvet is overcome by this approach. Further, DBG is decomposed into sub-graphs for identification of unique nodes in each sub-graph. Finally, MetaVelvet is used for generation of contigs and scaffolds.

The problem of assembling metagenome of species having large genomes was addressed by Li *et al*. (2015) by developing MEGAHIT based on succinct DBG (SdBG), i.e.

compressed representation of DBG. This is the first attempt of efficient metagenome assembly from NGS data of large genomes. In this parallelized algorithm was implemented for construction of SdBG graph which encodes m edges on O(m) bits, and supports O(1) time traversal from a vertex to its neighbors. Although, it offers many advantages over DBG but its construction is a difficult task. One of major bottleneck involved in constructing of SdBG is the sorting a set of (k+1)-mers representing edges of an SdDB in reverse lexicographical order of their length-k prefixes. MEGAHIT tackles this problem by (i) parallelizing this with the help of BWT-construction algorithm CX1(Liu *et al*. 2014), and (ii) taking advantage of a GPU to sort suffices of a set of reads very efficiently. In order to address the problem of limited on-board memory of GPU, block wise strategy that partition the k-mers according to their length-*l* prefix (where *l*=8) was adopted. Accordingly, k-mers in consecutive partitions are sorted together in memory. The issue of erroneous k-mer singletons, which increases the memory consumption significantly was resolved by notion of mercy k-mers for strengthening the contiguity of low-depth regions, as, the removal of such reads may lead to the removal of low-abundant species from data. MEGAHIT is available in both CPU only and GPU accelerated versions. An enhancement to MEGAHIT has been proposed by Li *et al*. (2016) with improved algorithms for assembly and merging of long bubbles. Construction of SdBG is reduced by developing a more refined algorithm along with modifications in existing data structure for reducing the memory usage.

Paired-read Iterative Contig Extension (PRICE) is a software developed by Ruby *et al*. (2013) for complex metagenomics data having highly uneven coverage across an entity along with tiny portion of a massive largely irrelevant dataset. This is quiet useful for metageomic assembly of viral genomes. This software operates in cycles, with each cycle in the assembly includes various steps like (i) mapping of reads to existing contigs, assembly with the help of pair end read information to create larger contigs, (ii) construction of scaffolds linking multiple seed contigs that can be assembled together into a single sequence, (iii) avoidance of spurious assemblies that can be created by multi-copy genetic elements, (iv) evaluation of the output sequence to determine its relevancy to the original target of the assembly and, (v) removal of redundant output sequence. This cycle of steps is repeated iteratively till desired results are obtained. Also, Ray Meta assembler associated with Ray Communities has been proposed by Boisvert *et al*. (2012) which profiles microbiomes based on uniquely-colored k-mers and have the capability of assembling three billion reads from a metagenomic experiment representing 1000 bacterial genomes of uneven proportions.

Recently, an assembler named as metaSPAdes has been developed which is useful in case of single cell and highly polymorphic diploid genomes (Nurk *et al.* 2017). This tool is useful in efficient assembling of metagenomic data from application of single cell (Kashtan *et al.* 2014) and TruSeq Synthetic Long Reads, i.e. TSLRs (Sharon *et al.* 2015) technologies. Therefore, it is capable of assembling vast micro diversity strains which shares genomic sequences having variations due to mutations, insertions of mobile elements, genome re-arrangements, and gene transfer. metaSPAdes initially constructs DBG using SPAdes and simplifies using various graph simplification algorithms. The main objective of metaSPAdes is to generate consensus assembly of strain mixture. Genomic differences between related strains and other sequencing errors leads to formation of large buldges and longer tips which are detected and removed following the algorithmic approaches suggested by Pevzner *et al.* (2004), Zerbino and Birney (2008). In order to preserve the information about rare strains, this assembler first detects the filigree edges formed due to rare strain variants and then disconnect them from their predominating vertices in the assembly graph instead of removing. Further, metaSPAdes algorithm also resolves repeats through application of exSPAnder module of SPAdes after modification in its decision rules by making it more conservative (Prjibelski *et al.* 2014, Vasilinetc *et al.* 2015, Antipov *et al.* 2016). Subsequently, undesirable path extensions are filtered out using coverage estimate of region being constructed. The various computational features of metagenomic assemblers are provided in Table 1.

Empirical evaluation and discussion

Computational comparison of recent assemblers namely MEGAHIT, Meta-IDBA, MetaVelvet and metaSPAdes has been done for assessing the memory and time complexity on a sever machine with four processors (each with 8 cores), 256 GB RAM and hard disk of 3 TB approximately. For the purpose of performance evaluation of these assemblers, two benchmark datasets (Vollmers *et al.* 2017) representing bacterial community of varying complexity are selected and downloaded from NCBI. The dataset samples are chosen from Kelp BioFilm (KBF) and Marburg Forest Soil (MFS) samples. KBF dataset comprises low community diversity but high read redundancy, whereas, MFS dataset is highly community diversified but is with low read redundancy. Raw sequences from chosen dataset are pre-processed for adapter clipping, quality and length trimming of sequences using

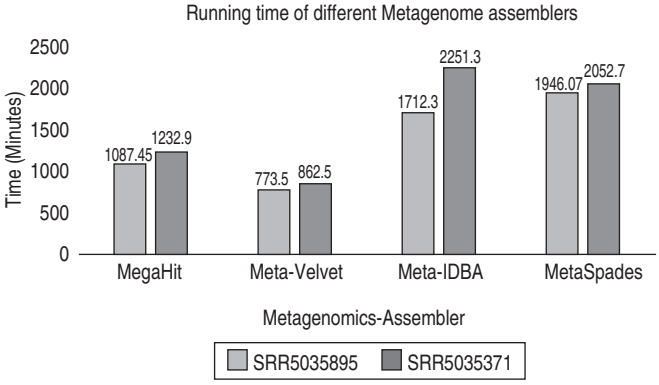

Fig 1 Comparison of time of assembly for selected assemblers for KBF and MFS datasets.

Table 1 Computational features of metagenome assemblers

Name of assembler	Data structure	Novelty in approach	Parallelism	Limitation	Language developed	Read size/ platform
Genovo .	OLC	Extracts more information from datasets and thus is suitable for low abundance sequences.	Single Server	Assessment with Illumima dataset	C++	454 sequencing
Meta-IDBA	DBG	-Handled the polymorphism in similar species. -Change in abundance ratio does not affect the performance	Single Server	-Quality of assembly depends on the quality of DBG. -Many species leads to many false positive edges which leads to many small components.	C++	NGS
MetaVelvet	DBG	Decomposition of DGB graph constructed from the mixed short reads into sub-graphs	Single Server	Accurate identification of chimeric reads	C++	Short Read
PRICE	DBG	Contig length extension using paired read	Multi-threading on multicore single CPU.	-	C++	Short Read Shotgun metagenomics data
XGenovo	OLC	-Additional parameter for solving chimera contig case -Use of modified sampling process for the location of read in a contig	Single Server	-Implementation of probabilistic model with DBG graph -high computational cost	C++	454 paired end reads
Omega	OLC	Uses less computing time as compared to existing OLC based assemblers at higher coverage	Multi-threaded and multiprocess distributed memory	Less efficient as compared to DBG based assemblers	C++	Illumina Long Reads
MetaVelvet-SL	DBG	Better detection of chimeric nodes by supervised learning.	Single Server	Classification model for chimeric reads may be improved.	C++	Short Read
MEGAHIT	Succinct DBG	-No need for normalization and partitioning -Efficient dynamic removal of edges	GPU based and 3-5 times fast as compared to its CPU counterpart.	Parallelization is restricted by small size of on-board GPU memory.	C++	NGS
MEGAHIT v1.0	Refined Succinct DBG	-Long bubble merging -Faster and uses less memory than MEGAHIT	Runs on Single Sever	-	C++	NGS
metaSPAdes	DBG	Detection and removal of large bulges and longer tips	Single Server	Incorporation of recent high quality jumping mate-pair libraries	C++	Single cell and TSLRs

Trimmomatic tool. Running time and peak memory usage of selected assemblers for complete assembly process has been observed and shown in Fig 1 and Fig 2, respectively.

Following points can be observed from this empirical analysis

The assembly of KBF dataset takes less time as compared to MFS irrespective of assemblers. However peak memory usage for KBF is more in case of MEGAHIT and Meta-IDBA, whereas it is less in case of metaSPAdes and Meta-Velvet.

The performance of MetaVelvet with respect to running time is best which is followed by MEGAHIT, whereas the performance of Meta-IDBA is comparable with metaSPAdes specially for KBF dataset. However, for MFS dataset, Meta-IDBA performs poorly in terms of running time than other assemblers.

The peak memory requirement for MEGAHIT is the least followed by metaSPAdes, MetaVelvet and Meta-IDBA.

The runtime requirement for KBF dataset is less in

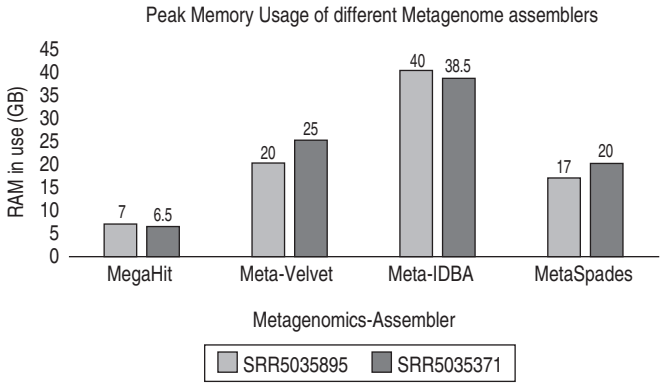

Fig 2 Comparison of peak memory usage for selected assemblers for KBF and MFS datasets.

comparison to MFS dataset across all assemblers is less due to the fact that this dataset is less diverse and have high read redundancy as comparison to MFS dataset. The runtime requirement for metagenome assembly for both datasets is less in case of MetaVelvet, as it partition the graph into sub-graph with the help of statistical distribution of k-mers. Although, Meta-IDBA also makes sub-graphs from DBG graph based on identification of particular graph pattern of k-mers which may be computationally less efficient.

The performance of MEGAHIT in terms of runtime assembly is likely to be much better than MetaVelvet on GPU machine. Run time performance of metaSPAdes is on expected lines, as this assembler is specifically developed for single cell and sequencing data based on TSLR technology. Peak memory requirements for assembly in MEGAHIT is least due to the reason that it dynamically removes the edges and normalization as well as partitioning of graph is not required. This has been followed by less memory requirement of metaSPAdes which may be due to removal of large buldges and longer tips during the construction of graph.

Conclusion and future research directions

In general, it may be difficult to identify the computationally best assembler from the existing assemblers as their performance may vary based on characteristic of dataset as well as architecture of the computational resources available for metagenome assembly. However, from this study it can be concluded that the performance of MetaVelvet is considerably better than its counterparts. Although, the performance of MEGAHIT is likely to be better on GPU environment. The computational performance of metagenome assembly process can be further improved in future by (i) computationally efficient graph partitioning to handle highly diversified multi-genomic data, (ii) application of better time and memory management data structures, (iii) implementation of novel parallelization approaches and Big Data technologies, and (iv) incorporating recently introduced auxiliary information such as Nextra Mate Pair libraries with existing recent TSLR technology in existing metagenome assemblers.

REFERENCES

Afiahayati S K and Akakibara Y S. 2013. An extended genovo metagenomic assembler by incorporating paired-end information. *Peer J* **1**: e196.

Afiahayati S K and Sakakibara Y. 2015. MetaVelvet-SL: an extension of the Velvet assembler to a de novo metagenomic assembler utilizing supervised learning. *DNA Research* **22**(1): 69–77.

Antipov D, Korobeynikov A, McLean J S and Pevzner P A. 2016. HYBRIDSPADES: an algorithm for hybrid assembly of short and long reads, *Bioinformatics* **32**(7): 1009–1015.

Boisvert S, Raymond F, Godzaridis É, Laviolette F and Corbeil J. 2012. Ray Meta: scalable de novo metagenome assembly and profiling. *Genome Biology* **13**: R122.

Bokhari S H and Sauer J R. 2005. A parallel graph decomposition algorithm for DNA sequencing with nanopores. *Bioinformatics* **21**(7)**:** 889-896.

Boucher C, Bowe A, Gagie T, Puglisi S J and Sadakane K. 2015. Variable-order de Bruijn graphs. *(In) Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, April 7-9, pp 383–392.

Breitwieser F P, Lu J and Salzberg S L. 2017. A review of methods and databases for metagenomic classification and assembly. *Briefings in Bioinformatics* bbx120.

Chatterji S, Yamazaki I, Bai Z *et al*. 2008. CompostBin: a DNA composition-based algorithm for binning environmental shotgun reads. *Research in Computational and Molecular Biology (LNCS),* Vol 4955, pp 17-28. Vingron M and Wong L (Eds). Springer, Berlin, Heidelberg.

Chevreux B, Pfisterer T, Drescher B, Driesel A J, Müller *et al.* 2004. Using the miraEST assembler for reliable and automated mRNA transcript assembly and SNP detection in sequenced ESTs. *Genome Research* **14**(6): 1147–59.

Chikhi R and Rizk G. 2013. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. *Algorithms for Molecular Biology* **8**: 22.

Conway T C and Bromage A J. 2011. Succinct data structures for assembling large genomes. *Bioinformatics* **27**(4): 479–86.

Dinh H and Rajasekaran S. 2011. A memory-efficient data structure representing exact-match overlap graphs with application for next-generation DNA assembly. *Bioinformatics* **27**(14): 1901–7.

El-Metwally S, Hamza T, Zakaria M and Helmy M. 2013. Next-Generation sequence assembly: Four stages of data processing and computational challenges. *PLoS Computational Biology* **9**(12): e1003345.

Ghurye J S, Cepeda-espinoza V and Pop M. 2016. Metagenomic assembly: Overview, challenges and applications. *Yale Journal of Biology and Medicine* **89**: 353–62.

Gonnella G and Kurtz S. 2012. Readjoiner: a fast and memory efficient string graph-based sequence assembler. *BMC Bioinformatics* **13**: 82.

Haider B, Ahn T, Bushnell B, Chai J, Copeland A and Pan C. 2014. Omega: an overlap-graph de novo assembler for metagenomics. *Bioinformatics* **30**(19): 2717–22.

Huang X, Wang J, Aluru S, Yang SP and Hillier L. 2003. PCAP: a whole-genome assembly program. *Genome Research* **13**(9): 2164–70.

Kashtan N *et al.,* 2014. Single-cell genomics reveals hundreds of coexisting subpopulations in wild. *Prochlorococcus Science* **344**: 416–20.

Kececioglu J D and Myers E W. 1995. Combinatorial algorithms for DNA sequence assembly. *Algorithmica* **13**(1-2): 7-51.

Kleftogiannis D, Kalnis P and Bajic VB. 2013. Comparing memory-efficient genome assemblers on stand-alone and cloud infrastructures. *PlosONE* **8**(9): 1–11.

Laserson J, Jojic V and Koller D. 2011. Genovo: De Novo assembly for metagenomes. *Journal of Computational Biology* **18**(3): 48–53

Leung H C M, Yiu S M and Yang B *et al*. 2011. A robust and accurate binning algorithm for metagenomic sequences with arbitrary species abundance ratio. *Bioinformatics* **27**(11): 1489–95.

Li D, Liu C, Luo R, Sadakane K and Lam T. 2015. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics* **31**(10): 1674–6.

Li D, Luo R, Liu C, Leung C, Ting H *et al*. 2016. MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods* **102**: 3–11.

Li R, Zhu H, Ruan J, Qian W, Fang X *et al.* 2010. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research* **20**(2): 265.

Lin Y and Pevzner P A. 2014. Manifold de Bruijn Graphs. *Algorithms in Bioinformatics: 14th International Workshop Lecture Notes in Computer Science,* Vol 8701, pp 296–310. Frith M, Pedersen C N S(Eds). Springer, Berlin.

Lin Y, Yuan J, Kolmogorov M, Shen M W, Chaisson M and Pevzner P A. 2016. Assembly of long error-prone reads using de Bruijn graphs. *Proceeding of National Academy of Sciences of United States of America,* Vol 113, pp E8396–E8405. Waterman M S (Eds). USA.

Liu C, Luo R and Lam T. 2014. GPU-accelerated BWT construction for large collection of short reads. arXiv , **1401**: 7457.

Namiki T, Hachiya T, Tanaka H and Sakakibara Y. 2012. MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Research* **40**(20): e155.

Nurk S, Meleshko D, Korobeynikov A and Pevzner P A. 2017. metaSPAdes: a new versatile metagenomic assembler. *Genome Research* **27**: 824–34.

Mande S S, Mohammed M H and Ghosh T S. 2012. Classification of metagenomic sequences: methods and challenges. *Briefing in Bioinformatics* **13**(6): 669–81.

Miller J R, Koren S and Sutton G. 2010. Assembly algorithms for next-generation sequencing data. *Genomics* **95**: 315–27.

Myers E W, Sutton G G, Delcher A L, Dew I M, Fasulo D P. *et al.* 2000. A whole-genome assembly of *Drosophila*. *Science* **287**(5461): 2196–204.

Myers E W. 2005. The fragment assembly string graph. *Bioinformatics* **21**(2): ii79-ii85.

Peng Y, Leung H C M, Yiu S M and Chin F Y L. 2011. Meta-IDBA: a de Novo assembler for metagenomic data. *Bioinformatics* **27**: i94–i101.

Peng Y, Leung H C M, Yiu S M and Chin F Y. 2012. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* **28**: 1420–1428.

Pevzner P A, Tang H and Waterman M S. 2001. An Eulerian path approach to DNA fragment assembly. *Proceedings of National Academy of Sciences of United States of America,* Vol 98, pp 9748–9753. Waterman M S (Eds). USA.

Pevzner P A and Tang H. 2001. Fragment assembly with double barreled data. *Bioinformatics* **17(1)**: S225-S233.

Pevzner P, Tang H and Tesler G. 2004. De novo repeat classification and fragment assembly. *Genome Research* **14**: 1786–96.

Prjibelski A D *et al*. 2014. ExSPAnder: a universal repeat resolver for DNA fragment assembly. *Bioinformatics* **30**: i293–i301.

Ruby J G, Bellare P and DeRisi J L. 2013. PRICE: Software for the targeted assembly of components of (Meta). *Genomic Sequence Data* **3**: 865-880.

Sharon I, Kertesz M, HugL A, Pushkarev D, Blauwkamp T A *et al*. 2015. Accurate, multi-kb reads resolve complex populations and detect rare microorganisms. *Genome Research* **14**(2): 55.

Salikhov K, Sacomoto G and Kucherov G. 2014. Using cascading Bloom filters to improve the memory usage for de Brujin graphs. *Algorithms for Molecular Biology* **9**(2): 48–65.

Simpson J T and Durbin R. 2010. Efficient construction of an assembly string graph using the FM-index. *Bioinformatics* **26**: 367–73.

Simpson J T, Wong K, Jackman S D, Schein J E, Jones S J M and Biro I. 2009. ABySS: A parallel assembler for short read sequence data. *Genome Research* **19**: 1117–23.

Teeling H, Waldmann J, Lombardot T *et al*. 2004. TETRA: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in DNA sequences. *BMC Bioinformatics* **5**: 163.

Vasilinetc I, Prjibelski A D, Gurevich A, Korobeynikov A and Pevzner P. 2015. Assembling short reads from jumping libraries with large insert sizes. *Bioinformatics* **31**: 3262–8.

Vollmers J, Wiegand S and Kaster A K. 2017. Comparing and evaluating metagenome assembly tools from a microbiologist's perspective – Not only size matters! *PLoSONE* **12**(1): e0169662.

Wu Y W and Ye Y. 2011.A novel abundance-based algorithm for binning metagenomic sequences using l tuples. *Journal of Computational Biology* **18**(3): 523–34.

Ye C, Ma Z S, Cannon C H, Pop M and Yu D W. 2011. Sparse assembler: de novo assembly with the sparse de Bruijn graph. *arXiv preprint arXiv*, 1106.2603.

Zerbino D R and Birney E. 2008. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Research* **18**: 821–9.