Evaluation of Next Generation Sequencing software in mapping and assembly

SuYing Bao[1], Rui Jiang[2], WingKeung Kwan[3], BinBin Wang[4], Xu Ma[4], You-Qiang Song[1]

[1]Department of Biochemistry, Center for Reproduction, Development and Growth, The University of Hong Kong, Hong Kong, Hong Kong
[2]Ministry of Education Key Laboratory of Bioinformatics and Bioinformatics Division, Department of Automation and Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China
[3]Computer Center, The University of Hong Kong, Hong Kong, Hong Kong
[4]National Research Institute for Family Planning, Beijing, China

Correspondence: Dr YQ Song, Department of Biochemistry, Centre for Reproduction, Development and Growth, The University of Hong Kong, 21 Sassoon Road, Hong Kong, Hong Kong.
E-mail: songy@hku.hk

Abstract

Next-generation high-throughput DNA sequencing technologies have advanced progressively in sequence-based genomic research and novel biological applications with the promise of sequencing DNA at unprecedented speed. These new non-Sanger-based technologies feature several advantages when compared with traditional sequencing methods in terms of higher sequencing speed, lower per run cost and higher accuracy. However, reads from next-generation sequencing (NGS) platforms, such as 454/Roche, ABI/SOLiD and Illumina/Solexa, are usually short, thereby restricting the applications of NGS platforms in genome assembly and annotation. We presented an overview of the challenges that these novel technologies meet and particularly illustrated various bioinformatics attempts on mapping and assembly for problem solving. We then compared the performance of several programs in these two fields, and further provided advices on selecting suitable tools for specific biological applications.

Keywords: next generation sequencing (NGS); NGS tools; NGS platforms; short reads mapping; de novo assembly

INTRODUCTION

'Next-generation sequencing' (NGS) platforms has been introduced and are wildly available recently,[1, 2] although large-scale sequencing laboratories were significant contribute to Human Genome Project.[3, 4] The limitations of the conventional Sanger (or di-deoxy terminator[5]) strategy urgently required certain new technologies for sequencing human genomes in parallel despite these dramatic improvements in this era. Thanks to the recent availability of optical instruments and the application of molecular biology,[1] a series of new massively parallel sequencing technologies, the NGS technologies, have tremendously changed this scenario.

Three platforms have been availabile: the Roche/454 FLX (30) (http://454.com/products-solutions/454-sequencing-system-portfolio.asp), the Illumina/Solexa Genome Analyzer (7) (http://www.illumina.com/pages.ilmn?ID=203) and the Applied Biosystems SOLiDTM System (http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing.html). These methods are all based on a template amplification phase before sequencing. Two new systems, the Helicos HeliscopeTM (www.helicosbio.com) and Pacific Biosciences SMRT (www.pacificbiosciences.com) instruments,[6] which avoid the amplification step and use single molecule as template, were also introduced recently.

These new technologies are advantageous because of their high throughput and low cost per base with over one billion reads per run incurring significantly lower base-cost,[2] which have given great impetus to the achievement of the 1000 Genomes Project goal.[7] These important characteristics permit the ultra-deep sequencing technologies to be widely used in the field of biology and medical research. NGS technologies have also made a huge and ongoing impact on transcriptome, gene annotation and RNA splice identification in addition to the traditional applications of DNA sequencing in genome resequencing and SNP discovery, Metagenomic[8] and genome methylation analysis[9] have also benefited from these new technologies. A new applications is also likely to be unveiled in the coming years.[1] The most fundamental steps for almost all of these applications are the mapping of the reads to the reference genome and the assembly of the reads to attain the desired DNA sequence for analysis.[10]

However, certain obstacles stemming from the NGS's inherent characteristics need to be eliminated before these technologies can be extensively used. The limitations on short read lengths (typically 35–400 bp compared with 650–800 bp of Sanger-based technology reads), low reading accuracy in homopolar stretches of identical bases, and non-uniform confidence in base calling require more efficient software and algorithms to help these new technologies develop further in the immediate future. Massive tools for NGS reads mapping and assembly have been flooding the market until now. We will only discuss some of the software, which we have first-hand experience on (considering the rapid developments in this field), and compare their working efficiency in terms of sensitivity, accuracy, speed and random-access memory (RAM) requirement.

MAPPING

Mapping tools overview

The most important step in NGS analysis is the mapping of reads to the original sequences.[1] Alignment, as a classical problem in bioinformatics, requires finding the most credible source for the sequenced DNA,[11] using the information of which species the reads have been generated. We also have to consider two fundamental issues aside from the shorter reads that are produced by NGS (compared with those from gel-capillary technology). One is the significantly greater amount of data, which requires optimized memory usage and speed, and the other is the different error profiles of data from the previous technologies. These call for algorithms that can be used to obtain as much information as possible from the sequencing data.[10] The traditional methods such as the pure Smith-Waterman dynamic programming, BLAT or BLAST may map the reads in a few days (given a large and expensive computer grid), however, such grids are not available to everyone. Some of the previous programs that are performing for the Sanger sequencing reads have not yet adapted to the huge volumes of data produced by NGS. Moreover, certain error characteristics with second generation sequencing, for example, Roche 454, have the tendency to have insertion or deletion errors during homopolymer runs,[12] therefore, they need to be considered when designing analysis tools.

Many methods are introduced and tools or programs based on these algorithms have been reported on an almost weekly basis to meet these challenges.[13] Doruk Bozdag and Umit Catalyurek from the Ohio State University proposed six parallelization methods to improve the hash/index-based short-sequence mapping: partitioning reads only, partitioning genome only, partition reads and genome, suffix-based assignment (SBA), SBA after partitioning reads and SBA after partitioning genome (see Bozdag *et al.*[14] for the details of the algorithms). CloudBurst, presented by Schatz *et al.*,[15] is a sensitive parallel seed-and-extend read-mapping algorithm, optimized for mapping single-end (SE) reads. BreakDancer, consisting of two complementary algorithms (BreakDancerMax and BreakDancerMini), supports pooled analysis across multiple samples and libraries.[16] Clement *et al.*[17] introduced a program called GNUMAP (Genomic Next generation Universal MAPper), which uses the quality score to get more accurate results from fewer sequencing runs (which are often costly). Other tools such as PASS,[18] SOAP2,[19] Bowtie,[20] CloudBurst,[15] MAQ,[21] ZOOM,[22] SHRIMP,[23] PERM[24] and others are also designed recently for NGS data.

Some researchers categorized the tools based on whether the genome or reads are indexed.[1, 25] Certain software, such as CloudBurst,[15] Eland, MAQ,[21] RMAP,[26] SeqMap,[27] SHRiMP[23] and ZOOM,[22] work by constructing hash tables for short reads and mapping them to the original genome sequences. The memory occupancy of these programs depends on the amount of reads that they processed, but it would be time consuming to scan the whole-genome when few reads are mapped.[25] Some programs such as BFAST,[28] Bowite,[20] BWA,[25] MOM,[29] MosaikAligner (http://bioinformatics.bc.edu/marthlab/Mosaik), NovoAlign (http://www.novocraft.com), SOAP,[19] PASS,[18] PerM,[24] ProbeMatch,[30] SSAHA2,[31] index genomic sequence. This kind of software can easily be parallelized to work on multithreading at the cost of larger memory occupancy if the

original genome is large such as the human genome sequence. However, this limitation can be ignored if more efficient strategies are involved in the indexing process, similar to what Bowtie, SOAP2 and BWA do. In fact, indexing the genome and mapping the reads to the index usually occupy similar RAM as in the case of inverse operation (indexing the reads and mapping the reads to the genome).[1] The third category that includes Slider I and Slider II[32] achieves short-reads alignment by merge-sorting the subsequences of the genome and the tags from NGS platforms (mainly Illumina/Solexa).

These mapping tools for NGS, when referring to indexing strategies, can also be divided into two main categories: hash table-based algorithms and Trie/Burrows–Wheeler Transform (BWT)-based algorithms. The former approach that basically follows seed-and-extend paradigm was the first wave of alignment programs. Many improvements have been developed since the very first hash-based algorithm, BLAST, to adapt to the specific characteristics of NGS reads mapping. First, the concept of spaced seed is introduced by Lin *et al.*[22] on the seeding approach, and several programs[23, 33] have implemented q-*gram* filter and multiple seed hits while seeding. Another development was on the seed extension aspect, in which CPU SIMD instructions are involved to achieve parallelize alignment and dynamic programming was used to accelerate alignment speed. Most of the software available now (all the programs mentioned above, excluding Bowtie, BWA and SOAP2) are based on this strategy. The trie-based algorithms efficiently cut down the complexity of inexact matching problem to the exact matching problem.[34] However, the memory used to hold the full occurrence array and prefix/suffix array is huge. The introduction of BWT algorithm[35] has significantly reduced the memory desired and led to the development of several tools like SOAP2 and Bowtie. Readers who are interested to know more about the Trie-based algorithm and BWT concept can refer to Li and Durbin.[25]

The software mentioned above can also be classified into two groups based on whether the 'quality scores' of nucleotide is involved during the mapping. Quality scores that come with reads from NGS platforms (mainly from Illumina) are, arguably, crucial in preventing the possibility of trivial matches during the mapping. Most of the tools[18, 19, 20, 21, 22, 23, 24, 25, 26, 28] available now use base quality information when they do mapping tasks, although some of them may not fully use it to advance mapping accuracy. However, there are also some programs, such as CloudBurst, SeqMap, MOM, ProbeMatch and Slider, that involve nucleotide information only for short reads alignment. Slider, on another hand, fully utilizes short reads' probability information (given in the *prb* file from Illumina Sequence Analyzer) to reduce the alignment problem space.[32] More details on the tools mentioned above are in Table 1.

Evaluation of mapping tools

To illustrate the performance of these mapping tools, we basically consider the following statistic indexes: mapping speed, RAM occupancy, sensitivity (measured as the percentage of reads mapped) and accuracy (in terms of the percentage of reads mapped correctly). We evaluated the performance of several tools, namely, SOAP_2.2, Bowtie_0.12.5, SeqMap_1.0.13, MOM_0.6, SHRiMP_2.0.1, PASS_v1.2, BWA_0.5.9, RMAP_v2.05, Mosaik_1.1.0021 and SSAHA2_v2.5.3, either using simulated data or the real data from Illumina platform. Those tools, with versions currently available during the time of our research, are widely used in the fields of Illumina reads

mapping analysis. We first performed a simulation work on the chosen tools and summarized their efficiencies in terms of speed, memory usage, sensitivity and accuracy. Then we evaluated their mapping capacities on real applications, with Illumina reads from 1000 Genomes Project Database (http://www.1000genomes.org/data). Based on the evaluated tools' own heuristics, we fixed parameters so as to get all programs' equally best matches, with up to two mismatches.

Evaluation on simulation data

We used dwgsim, a utility for whole-genome Illumina reads simulation, contained in DNAA_0.1.2 (http://sourceforge.net/projects/dnaa/), to generate Illumina-like short sequences, using the default empirical error model illustrated on DNAA's Whole-Genome Simulation web (http://sourceforge.net/apps/mediawiki/dnaa/index.php?title=Whole_Genome_Simulation). In total, we generated 15 million reads with 76 bp length using the complete human genome (hg18) as a reference. Details of the codes used to run those tools mentioned above with the simulation data can be found in Supplementary Information S1. Table 2 provides us the results of the simulation work with statistics on the number of reads mapped, the amount of reads correctly mapped, time consumed and RAM required.

From Table 2, we found that for Illumina SE reads mapping, SHRiMP provided the highest true mapping percentage (around 99%) among all programs, at the expanse of consuming much more time and RAM than others. BWA, which is the second most accuracy (around 4% less than that of SHRiMP), performed tremendously faster than SHRiMP and occupied least memories among all tools. Other tools, including Bowtie, Mosaik, RMAP, SeqMap and SOAP, can all correctly catch more than 75% genuine matches, with SOAP most speedy while Bowtie most RAM-saved. For paired-end (PE) mapping tasks, the validate alignments of BWA (who can correctly map more than 98% of all reads to human reference, with the least RAM usage and acceptable completion time) are remarkably more than the alignments of other tools. SSAHA2 and SHRiMP behaved similarly as BWA did in terms of mapping sensitivity and accuracy. However, they occupied tremendously more RAM and time than BWA did for the same task.

Evaluation on real data

To further compare the behavior of those tools on real applications, we used around 12 million Illumina SE reads with length of 76 (AC:ERR008834) and 17 million pairs of 76 reads (AC: SRR043391) from Sequence Reads Achieve to align against the whole human genome sequences (assembly: NCBI36.1/hg18). Table 3 illustrates the results of this evaluation experiment. Compared with the results on Table 2, Table 3 indicated that the conclusions of evaluation on real applications are generally consistent with the results from simulation work, except that Mosaik acted slightly better than BWA, and SHRiMP performed not as well as it did in PE mapping. Thus, the parameters, such as sequence errors, fraction of indels and outer distance between the two ends, set in our simulation experiment seemed to have little effect on capturing the general divergences of mapping performance between those tools selected.

As additional remarks to the experiments mentioned above, several points needed to be stated here: (1) MOM has also been tested with our simulation data and real reads from 1000 Genomes Project, however, this program seems not so stable to input file formats and no certain bug information

was given to guide users to resolve the problem. (2) Although a 'PE' section has been posted on PASS website, it seems that PASS was still on developing of this application. (3) All experiments are run on our 64-bit quad-core Linux system, with 32 GB RAM.

Discussions on mapping tools

Generally speaking, Bowtie, BWA, Mosaik, SHRiMP and SOAP all provide satisfactory mapping results in both SE and PE Illumina reads alignments, with BWA using much less RAM than the others, which is mostly owed to its BWT-based algorithm, whereas SOAP providing the fastest performance among all tools, which is likely benefited from its core algorithm (2way-BWT). The differences of those methods on mapping sensitivities could mostly be attributed to the heuristics applied by different algorithms in detecting imperfectly matching positions.[1] The apparently excellent performance of BWT-based aligners in time consumption and memory occupancy could mainly be attributed to their multithreading processing characteristic and independence from the amount of reads to be aligned.[25] Although certain programs, such as SHRiMP, have elegant performance in terms of mapping sensitivity and accuracy, the enormous time consuming and RAM occupancy need to be considered once again before using them as an aligner for large mammalian genomes. However, it would also be an option when it comes to mapping small genomes, like Drosophila.

Till now, only a few open source tools, such as Mosaik, PASS and SSAHA2, are available for 454 mapping and their sensitivities in catching mapping positions are not so satisfied, which calls for an urgent need for developing novel software supporting 454-like longer (typically 400–1000 bp) NGS reads. Although several programs, such as Mosaik, PASS, Bowtie, SHRiMP and/or some other tools, are declared as color-space-mapping available, their capabilities in matching SOLiD-specific reads are pretty low, which may mainly due to the specific design of ABI outputs. Algorithms involved with advanced spaced seeds would be a considerable modification for SOLiD mappers, as in Laurent Noe *et al.*[36] As this review mainly focuses on comparing the capacities of Illumina aligners, no certain evaluation results about 454 and SOLiD-supported tools are provided here. But authors also has performed simple testing studies on the tools declared as 454-bared, namely Mosaik, SSAHA2, PASS, and tools called themselves as color-space-tolerated, including Mosaik, PASS, Bowtie and SHRiMP, using 454 and SOLiD real reads from Sequence Reads Achieve (http://www.ncbi.nlm.nih.gov/sra). Readers with interests in applying those programs for 454 and SOLiD reads mapping could refer to Supplementary Information S2 and S3, in which details of the data involved and results of the experiments are represented, respectively.

Overall, decisions on choosing an appropriate method against another should mostly depend on the amount of reads to be mapped, the reference genome to be considered, and the computing equipment available. The final goals of certain experiments may also determine or help determine the choice.

ASSEMBLY

Assembly strategies

The lengths of individual sequencing read from either Sanger-based technology or novel NGS platforms are significantly shorter than the desired length of DNA sequence.[10] A so-called technology 'Assembly', first designed for cosimid[37] and then used in genomic analysis, was introduced in the late 1980s and early 1990s to resolve the problem. The fundamental concept in this technology is to group the random fragments of a significantly longer DNA sequence into contigs and then contigs into scaffolds to reconstruct the original DNA sequence. It can be divided into two different approaches: *de novo* approach and comparative (resequencing) approach based on the different focus of this technology.[38]

The *de novo* approaches mainly focus on reconstructing genomes that have never been sequenced, although it is sufficient for comparative approaches to map the reads to the guided sequence to characterize a newly sequenced organism. The *de novo* methods are irreplaceable, especially in discovering new, previously unknown sequences—this is essential for characterizing biological diversity of our world—but they are mathematically more complex and needs larger memory than the comparative ones. There are mainly two factors that influence the complexity of *de novo* assembly technology: the length and the volume of the reads. Shorter reads may complicate the layout phase of an assembly (because it is more difficult for *de novo* assemblers to handle repeats with short reads) but they are easier to be aligned. More reads also pose quadratic or even exponential complexity to the underlying algorithms but they promise better identification of sequence overlaps. Managing the large volumes of reads with even shorter length (typically 35–400 bp, which is significantly shorter than the traditional ones' 600–800 bp) from NGS and fully exploiting the deeper coverage produced by NGS technologies have become the most crucial issues being considered when researchers design assemblers for NGS.

These challenges lead to more considerable efforts being exerted in the modification of three widely used *de novo* assembly strategies:[10, 39] greedy, overlap–layout–consensus and Eulerian or de Bruijin graph.[40] The success of the recently introduced NGS assemblers is mainly caused by the development of pragmatic engineering and heuristics on assembly algorithms.[39] Some of the tools, such as SSAKE,[41] SHARCGS,[42] VCAKE,[43] and QSRA,[44] work by using greedy graph strategy. Programs applying this algorithm undertake one basic operation: iterative extension (that is, given any read or contig, it will merge with the one with the largest overlap). The three programs (SSAKE, VCAKE and QSRA) have been developed to handle imperfectly matching reads,[41, 43, 44] whereas SHARCGS is widely used on uniform-length, high-coverage and unpaired short reads. QSRA, the most recently developed software in this category, has an advantage in quality-value scores to help users deal with base call errors. It provides better and more preferable performance in terms of speed and output quality[44] compared with the other tools mentioned above. The second category of software that includes CABOG,[45] Edena,[46] Newbler[47] and Shorty[48] are based on overlap-layout-consensus. This strategy involves three main steps. First, assemblers

compare the reads to each other to construct an overlap graph in the first overlap discovery stage. Second, the overlap graph is analyzed and the appropriate paths traversing through the graph are identified in the layout stage. Third, consensus sequence will be determined through multiple sequence alignment. Newbler, among the overlap-layout-consensus-based software, was specifically designed to handle the ambiguity in the length of 454's homopolymer runs, whereas the other widely used programs (distributed by Illumina/Solexa), including Shorty, can also be applied to ABI/SOLiD and Helicos. CABOG, Newbler and Shorty can manage base calling error and repeats with their specific schemes, whereas Edena was designed for unpaired reads with uniform length. Newbler particularly applies instrument metrics to overcome inaccurate calls caused by homopolymer repeats in 454.[39] CABOG uses a so-called 'rocks and stones' technique,[49, 50] whose main procedure could be summarized as 'unitig-contig-scaffolds', for base call correction.[45] Shorty innovatively estimates the intercontig distances from the mate pairs using a few seeds of 300–500 bp length. The third category of software based on de Bruijn graph approaches[40] are widely used in assembling data from the Solexa and SOLiD platforms. The tools in this category (such as ABySS,[51] ALLPATHS,[52] EULER-SR,[53] SOAPdenovo[54] and Velvet[55]) have applied certain heuristic strategies to reduce the complexity of the de Bruijn graphs, which trivialize assembly problem by finding the path that would traverse each edge of the graph exactly once. EULER-SR[52] mitigates error sequencing impact by constructing different K-mer sizes De Bruijn graphs and reduces graph complexity by applying low-quality read ends and PE constraints. Velvet[55] uses an error-avoidance read filter for error calls correction and adopts a pebble smoothing technique, involving read threading and mate pairs for graph reduction. ABySS is an scalable assembly software and designed to overcome memory limitations in large genome assembly by distributing graph and graph computation across a compute grid. ALLPATHS targets large genomes and invokes tow pre-processors, read-correction processor and 'unipaths' creation processor, for erroneous base call correction and graph simplification. Finally, SOAPdenovo is, by far, the only software amalgamating de Bruijin graph and overlap-layout-consensus strategies together, in which a contig graph is constructed by the de Bruijin graph method although its complexity is reduced by cutting transitive edges and isolating multi-path involved contigs. Its transitive link deduction scheme is similar to CABOG's 'rocks and stones' method and to Velvet's breadcrumbs and pebble techniques.[39] Table 4 shows more details on the assembly programs. Several papers[10, 38, 39] have also provided significant insights on the technical strategies and tools of the *de novo* assembly of short reads.

Evaluation on assembly tools

The efficiency of assemblers is basically assessed through two indexes: size and accuracy of the assemblies' contigs and scaffolds.[39] However, N50, one of the widely used statistics for size measurement, can only be comparable between different assemblers when each is measured with the same combined length value. On another hand, the accuracy of assemblies is generally difficult to measure, although certain inherent accuracy measurement may be used for specific assembler. In our study, we applied six statistical values, namely, maximum contig length, minimum contig length, average contig length, genomic coverage (measured as the total length of reads used for constructing contigs divided by the length of all queries), total processed time and RAM occupancy, to illustrate the trade-offs between contig length and genomic coverage that certain assemblers have made while they are treating with large volume of short reads. Six widely

used assembly tools were involved, including QSRA,[44] SSAKE_v3-5,[41] Edena_2.1.1,[46] AByss_1.2.6,[56] SOAPdenovo_1.05[54] and Velvet_1.0.09.[55] Limited by our computer RAM available now (32 GB), we extracted 1.5 million reads and pairs from SE reads file ERR008834 and PE reads file SRR043391, respectively, as input queries. The results are shown in Table 5.

From Table 5, we see that, in SE test, SOAPdenovo and QSRA yielded distinctly higher genomic coverage than the other tools, around 60% higher, with generally a larger number of short contigs. As a contrast, SSAKE and Edena usually produce longer contigs with much lower genomic coverage. Among all the tools been tested, SOAPdenovo and AByss were the fastest, whereas Edena and QSRA were the most memory-efficient. For mate reads assemblies, wherein QSRA and Edena are not available, SOAPdenovo granted the most elegant performance with the highest genomic coverage and the least time and RAM requirement. AByss yielded the longest contigs, whereas reads from SSAKE were longer in general. Pop[38] and Miller *et al.*[39] have given further insights on the performance of the other *de novo* tools and assembly algorithm of NGS.

Discussions on assembly tools

As an interim conclusion, in our experiments SOAPdeovo offered more satisfactory performance, in terms of speed, memory usage and genomic coverage, than other tools in both SE and mate-end conditions, whereas QSRA behaved inferiorly in individual reads assembly. However, reads from both of those two programs are usually short. On another hand, SSAKE and Edena generally produce longer contiges with lower coverage rates. AByss could produce longest contigs using mate reads, although the average length of contigs from AByss is short. Among those tools been tested, Velevet, SSAKE and AByss cost more computer memory for the same task. In our experience more than 32GB of memory is needed to handle larger volumes (for example, more than ten million) of input reads using these programs. Also, compared with other assemblers, Velvet and SSAKE are more time consuming, which may limit their applications in the filed of *de novo* assembly. In summary, such approaches mentioned above all have to make a balance between the length of contigs and the coverage of genome.

Nevertheless, the scale of the analysis and the types of assay may decide the tool(s) to be used. Moreover, the heuristics for real reads error and genomes repeats owed by a certain assembler, and the computer source available may also profoundly influence the program's success in *de novo* assembly filed.

CHALLENGES AND PROSPECTS

Despite the strikingly attractive success of NGS in genomics and post genomics, three main challenges, which could be summarized as Computational Challenge, Developmental Challenge and Cross-Platform Unification Challenge, are blocking, or in a not short period will still block, the development of these new technologies from infancy to mature.

The growing gap between massive output data from NGS platforms and the computer source available to process and analyze them has to be bridged in an urgent need. Aligning millions or even billions of reads against a large mammalian genome as a complete experiment becomes common in today's genomic studies. However, super computers with abundant memories to handle such big headaches are not always available to every user. Timing is also an inevitable question while dealing with NGS tasks. Thus, an extraordinarily efficient algorithm is then urgently needed to reduce computing costs. Parallelization strategies, like BWT algorithm applied by BWA, Bowtie and SOAP2, have been proposed and managed to help aligners speed up their execution time and reduce their computer memory requirement with uncompromising results accuracy.[14]

As long as NGS technologies go on changing, developers of short reads mapping and assembly software have to keep pace with these novel techniques. To keep up or even exceed Sanger sequencers in terms of read length, which has critical effects on detecting split mapping signatures and *de novo* sequencing, NGS sequencing machines all try to produce longer reads. Thus, future mappers for short reads or NGS tools available now need to be adjusted as programs compatible with longer reads. Furthermore, unfamiliar data formats from so-called next–next-generation sequencers, such as Helicos HeliscopeTM and Pacific Biosciences SMRT, explosive mass of different experiments and divergent scale of analysis all call for more robust and efficient algorithms in automatically redressing parameters for specific demands.

Another main challenge met by developers of NGS mappers and assemblers comes from the standards inconformity in size of inserts between mates, error profiles and ‘true match’ benchmarks across diverse NGS platforms. Different sizes of inserts, which are common in variant NGS platforms, also have different potency in detecting variants.[57] Shorter insert sizes, compared with long inserts (which offer advantages in detecting larger events), increase the sensitivity of smaller events.[58, 59] Therefore, a combination of multiple libraries with varying insert sizes will be a good choice in future studies.[58, 60, 61] Furthermore, as different platforms produce reads with different error models and also isolate ‘real alignment’ from multiple possible matches with their own criterions, investigators are often embarrassed when they explore the data from several platforms. Thus, a unified standard for determining genuine match and a critical evaluation of the quality of data from these technologies are in urgent need.[62] In addition, considering that ‘NGS users are always puzzled by a complicated maze of base calling, alignment, assembly, and analysis tools with often incomplete documentation and providing no ideas on how to compare and validate the outputs, Paul Medvedev *et al.*,[57] recommended that new methods should combine the previous approaches and possess different types of signatures to support an event’.

Nevertheless, NGS approaches are undoubtedly here to stay and will propel the development of bioinformatics in several areas such as mapping, assembly, detecting variants, and other related areas, for many years.[1, 62] Their advantages in speed and cost[62] and their higher capabilities in detecting divergent types of variants[56, 59, 60, 61, 63] granted their wide applications in the field of medical research and diagnostics.[64] Moreover, genomics,[64] functional genomics,[9] proteomics,[64] transcriptome analysis,[65] epigenetic research[66] and the characterization of new virus[67] and bacterium[68, 69] all benefited from these technologies immediately after their introduction into the market.

CONCLUSION

Challenges definitely remain to be justified for the further development of NGS. More efforts need to be done, not only in the fields of mapping and assembly, but also on the areas of so-called ‘downstream analysis’, such as metagenomics, transcriptome analyses, small RNA detection and/or other related areas. New considerations and questions will continue to emerge, thus novel programs have to evolve rapidly to keep up with the pace of NGS and the changes in adoption of these techniques.

REFERENCE

1. Horner D.S., Pavesi G., Castrignano T., De Meo P.D., Liuni S., Sammeth M., et al. Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing. *Brief Bioinform.* (2009).
2. Metzker M.L. Applications of Next-Generation Sequencing Sequencing Technologies - the Next Generation. *Nat Rev Genet.* **11**, 31-46 (2010).
3. Tilford C.A., Kuroda-Kawaguchi T., Skaletsky H., Rozen S., Brown L.G., Rosenberg M., et al. A physical map of the human Y chromosome. *Nature.* **409**, 943-945 (2001).
4. Lander E.S., Linton L.M., Birren B., Nusbaum C., Zody M.C., Baldwin J., et al. Initial sequencing and analysis of the human genome. *Nature.* **409**, 860-921 (2001).
5. Sanger F., Nicklen S., Coulson A.R. DNA Sequencing with Chain-Terminating Inhibitors. *P Natl Acad Sci USA.* **74**, 5463-5467 (1977).
6. Mardis E.R. Next-generation DNA sequencing methods. *Annu Rev Genomics Hum Genet.* **9**, 387-402 (2008).
7. Service R.F. Gene sequencing - The race for the $1000 Genome. *Science.* **311**, 1544-1546 (2006).
8. Schuster S.C. Next-generation sequencing transforms today's biology. *Nature Methods.* **5**, 16-18 (2008).
9. Morozova O., Marra M.A. Applications of next-generation sequencing technologies in functional genomics. *Genomics.* **92**, 255-264 (2008).
10. Flicek P., Birney E. Sense from sequence reads: methods for alignment and assembly. *Nature Methods.* **6**, S6-S12 (2009).
11. Trapnell C., Salzberg S.L. How to map billions of short reads onto genomes. *Nat Biotechnol.* **27**, 455-457 (2009).
12. Flicek P., Birney E. Sense from sequence reads: methods for alignment and assembly (vol 6, pg S6, 2009). *Nature Methods.* **7**, 479-479 (2010).
13. Bateman A., Quackenbush J. Bioinformatics for Next Generation Sequencing. *Bioinformatics.* **25**, 429-429 (2009).
14. Bozdag D., Barbacioru C.C., Catalyurek U.V. Parallel Short Sequence Mapping for High Throughput Genome Sequencing. *Int Parall Distrib P.* 1033-1042 (2009).
15. Schatz M.C. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics.* **25**, 1363-1369 (2009).
16. Chen K., Wallis J.W., McLellan M.D., Larson D.E., Kalicki J.M., Pohl C.S., et al. BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nature Methods.* **6**, 677-U676 (2009).
17. Clement N.L., Snell Q., Clement M.J., Hollenhorst P.C., Purwar J., Graves B.J., et al. The GNUMAP algorithm: unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. *Bioinformatics.* **26**, 38-45 (2010).
18. Campagna D., Albiero A., Bilardi A., Caniato E., Forcato C., Manavski S., et al. PASS: a program to align short sequences. *Bioinformatics.* **25**, 967-968 (2009).
19. Li R.Q., Li Y.R., Kristiansen K., Wang J. SOAP: short oligonucleotide alignment program.

Bioinformatics. **24**, 713-714 (2008).

20. Langmead B., Trapnell C., Pop M., Salzberg S.L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.* **10**, R25 (2009).
21. Li H., Ruan J., Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.* **18**, 1851-1858 (2008).
22. Lin H., Zhang Z., Zhang M.Q., Ma B., Li M. ZOOM! Zillions of oligos mapped. *Bioinformatics.* **24**, 2431-2437 (2008).
23. Rumble S.M., Lacroute P., Dalca A.V., Fiume M., Sidow A., Brudno M. SHRiMP: accurate mapping of short color-space reads. *PLoS Comput Biol.* **5**, e1000386 (2009).
24. Chen Y., Souaiaia T., Chen T. PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics.* **25**, 2514-2521 (2009).
25. Li H., Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics.* **25**, 1754-1760 (2009).
26. Smith A.D., Chung W.Y., Hodges E., Kendall J., Hannon G., Hicks J., et al. Updates to the RMAP short-read mapping software. *Bioinformatics.* **25**, 2841-2842 (2009).
27. Jiang H., Wong W.H. SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics.* **24**, 2395-2396 (2008).
28. Homer N., Merriman B., Nelson S.F. BFAST: an alignment tool for large scale genome resequencing. *PLoS One.* **4**, e7767 (2009).
29. Eaves H.L., Gao Y. MOM: maximum oligonucleotide mapping. *Bioinformatics.* **25**, 969-970 (2009).
30. Kim Y.J., Teletia N., Ruotti V., Maher C.A., Chinnaiyan A.M., Stewart R., et al. ProbeMatch: rapid alignment of oligonucleotides to genome allowing both gaps and mismatches. *Bioinformatics.* **25**, 1424-1425 (2009).
31. Ning Z., Cox A.J., Mullikin J.C. SSAHA: a fast search method for large DNA databases. *Genome Res.* **11**, 1725-1729 (2001).
32. Malhis N., Butterfield Y.S.N., Ester M., Jones S.J.M. Slider-maximum use of probability information for alignment of short sequence reads and SNP detection. *Bioinformatics.* **25**, 6-13 (2009).
33. Weese D., Emde A.K., Rausch T., Doring A., Reinert K. RazerS-fast read mapping with sensitivity control. *Genome Res.* **19**, 1646-1654 (2009).
34. Li H., Homer N. A survey of sequence alignment algorithms for next-generation sequencing. *Brief Bioinform.* **11**, 473-483 (2010).
35. Noe L., Girdea M., Kucherov G. Designing Efficient Spaced Seeds for SOLiD Read Mapping. *Adv Bioinformatics.* (2010).
36. Staden R. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Res.* **6**, 2601-2610 (1979).
37. Pop M. Genome assembly reborn: recent computational challenges. *Briefings in Bioinformatics.* **10**, 354-366 (2009).
38. Miller J.R., Koren S., Sutton G. Assembly algorithms for next-generation sequencing data. *Genomics.* **95**, 315-327 (2010).
39. Pevzner P.A., Tang H., Waterman M.S. An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci U S A.* **98**, 9748-9753 (2001).
40. Warren R.L., Sutton G.G., Jones S.J.M., Holt R.A. Assembling millions of short DNA sequences

using SSAKE. *Bioinformatics.* **23**, 500-501 (2007).

41. Dohm J.C., Lottaz C., Borodina T., Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Res.* **17**, 1697-1706 (2007).
42. Jeck W.R., Reinhardt J.A., Baltrus D.A., Hickenbotham M.T., Magrini V., Mardis E.R., et al. Extending assembly of short DNA sequences to handle error. *Bioinformatics.* **23**, 2942-2944 (2007).
43. Bryant D.W., Jr., Wong W.K., Mockler T.C. QSRA: a quality-value guided de novo short read assembler. *BMC Bioinformatics.* **10**, 69 (2009).
44. Miller J.R., Delcher A.L., Koren S., Venter E., Walenz B.P., Brownley A., et al. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics.* **24**, 2818-2824 (2008).
45. Hernandez D., Francois P., Farinelli L., Osteras M., Schrenzel J. De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer. *Genome Res.* **18**, 802-809 (2008).
46. Margulies M., Egholm M., Altman W.E., Attiya S., Bader J.S., Bemben L.A., et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature.* **437**, 376-380 (2005).
47. Hossain M.S., Azimi N., Skiena S. Crystallizing short-read assemblies around seeds. *BMC Bioinformatics.* **10**, - (2009).
48. Myers E.W., Sutton G.G., Delcher A.L., Dew I.M., Fasulo D.P., Flanigan M.J., et al. A whole-genome assembly of Drosophila. *Science.* **287**, 2196-2204 (2000).
49. Venter J.C., Adams M.D., Myers E.W., Li P.W., Mural R.J., Sutton G.G., et al. The sequence of the human genome. *Science.* **291**, 1304-1351 (2001).
50. Simpson J.T., Wong K., Jackman S.D., Schein J.E., Jones S.J.M., Birol I. ABySS: A parallel assembler for short read sequence data. *Genome Res.* **19**, 1117-1123 (2009).
51. Butler J., MacCallum I., Kleber M., Shlyakhter I.A., Belmonte M.K., Lander E.S., et al. ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res.* **18**, 810-820 (2008).
52. Chaisson M.J., Pevzner P.A. Short read fragment assembly of bacterial genomes. *Genome Res.* **18**, 324-330 (2008).
53. Li R.Q., Zhu H.M., Ruan J., Qian W.B., Fang X.D., Shi Z.B., et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.* **20**, 265-272 (2010).
54. Zerbino D.R., Birney E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* **18**, 821-829 (2008).
55. Simpson J.T., Wong K., Jackman S.D., Schein J.E., Jones S.J., Birol I. ABySS: a parallel assembler for short read sequence data. *Genome Res.* **19**, 1117-1123 (2009).
56. Medvedev P., Stanciu M., Brudno M. Computational methods for discovering structural variation with next-generation sequencing. *Nature Methods.* **6**, S13-20 (2009).
57. Bentley D.R., Balasubramanian S., Swerdlow H.P., Smith G.P., Milton J., Brown C.G., et al. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature.* **456**, 53-59 (2008).
58. Bashir A., Volik S., Collins C., Bafna V., Raphael B.J. Evaluation of paired-end sequencing strategies for detection of genome rearrangements in cancer. *Plos Computational Biology.* **4**, - (2008).
59. Campbell P.J., Stephens P.J., Pleasance E.D., O'Meara S., Li H., Santarius T., et al. Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel

paired-end sequencing. *Nat Genet.* **40**, 722-729 (2008).

60. Korbel J.O., Abyzov A., Mu X.J., Carriero N., Cayting P., Zhang Z.D., et al. PEMer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data. *Genome Biol.* **10**, - (2009).
61. Pop M., Salzberg S.L. Bioinformatics challenges of new sequencing technology. *Trends in Genetics.* **24**, 142-149 (2008).
62. Mardis E.R. The impact of next-generation sequencing technology on genetics. *Trends Genet.* **24**, 133-141 (2008).
63. Ansorge W.J. Next-generation DNA sequencing techniques. *New Biotechnol.* **25**, 195-203 (2009).
64. Morozova O., Hirst M., Marra M.A. Applications of new sequencing technologies for transcriptome analysis. *Annu Rev Genomics Hum Genet.* **10**, 135-151 (2009).
65. Hurd P.J., Nelson C.J. Advantages of next-generation sequencing versus the microarray in epigenetic research. *Brief Funct Genomic Proteomic.* **8**, 174-183 (2009).
66. McHardy A.C., Adams B. The Role of Genomics in Tracking the Evolution of Influenza A Virus. *Plos Pathog.* **5**, - (2009).
67. Holt K.E., Parkhill J., Mazzoni C.J., Roumagnac P., Weill F.X., Goodhead I., et al. High-throughput sequencing provides insights into genome variation and evolution in Salmonella Typhi. *Nat Genet.* **40**, 987-993 (2008).
68. Engstrand L. How will next-generation sequencing contribute to the knowledge concerning Helicobacter pylori? *Clin Microbiol Infect.* **15**, 823-828 (2009).

Table 1 Tools for the analysis of next generation sequencing data

Program	Website	Open Source	Quality score involved	Mapping strategy	Description	Ref
CloudBurst	http://sourceforge.net/apps/mediawiki/cloudburst-bio/index.php?title=CloudBurst	Yes	No	Hash the reads	either all alignments or the unambiguous best alignment for each read with any number of mismatches or difference would be reported; running time required is linearly increase with the number of reads mapped, and near linearly decrease as the number of processors increase	15
Eland	None	No	Yes	Hash the reads	Probably the first read aligner; works only for 32-bp single-end reads by itself, with GAPipeline extending its ability	
Maq	http://maq.sourceforge.net	Yes	Yes	Hash the reads	based on a so called "spaced seed indexing" strategy, it can efficiently winnow the candidate locations within the reference	21
RMAP	http://rulai.cshl.edu/rmap/	Yes	Yes	Hash the reads	can map reads with or without quality scores; supports paired-end reads or bisulfite-treated reads mapping; no limitations on read widths or number of mismatches	26
SeqMap	http://biogibbs.stanford.edu/~jiangh/SeqMap/	Yes	No	Hash the reads	maps dozens of millions of reads to a genome with several billions bp length; can deal with mutations, insertions/deletions; supports various input/output formats, command option lines are also available	27
SHRiMP	http://compbio.cs.toronto.edu/shrimp/	Yes	Yes	Hash the reads	SAM output format; supports both letter space and color space reads; allows paired-end reads alignment, parallel computation	23
ZOOM	http://www.bioinfor.com	No	Yes	Hash the reads	based on spaced seed strategy; 100% sensitivity for a wide range of read length and mismatches; a single CPU with 6.5G memory, is capable to map 15X coverage of a human genome in one day	22
BFAST	http://sourceforge.net/projects/bfast/files/	Yes	Yes	Hash the genome	fast and accurate mapping of tags to genome sequences	28
MOM	http://mom.csbc.vcu.edu/	Yes	No	Hash the genome	no indels are allowed while mapping, but mismatches are tolerant;	29

					establishs a seed hash table for exactly matching short seeds between reference sequence and short reads	
Mosaik	http://bioinformatics.bc.edu/marthlab/Mosaik	Yes	Yes	Hash the genome	based on Smith-Waterman algorithm; supports pair-wise alignments and produces reference-guided assemblies with gapped alignments; written in highly portable C++	
SSAHA2	http://www.sanger.ac.uk/resources/software/ssaha2/	Yes	Yes	Hash the genome	support most sequencing platforms (ABI-Sanger, Roche 454, Illumina-Solexa); wild range of output formats(SAM, CIGAR, PSL etc.) are available; A separate package for pile-up pipeline analysis and genotype calling is also included	31
NovoAlign	http://www.novocraft.com	No	Yes	Hash the genome	allows gaps up to 7bp on single-end reads, even longer on paired end reads aligns with up to 8 or more mismatches per read, up to 16 on paired end reads	
PASS	http://pass.cribi.unipd.it	Yes	Yes	Hash the genome	improves the execution time and sensitivity; performs fast gapped and ungapped alignments of short reads onto a reference genome; implemented in C++, supported on Linux and Windows	18
PerM	http://code.google.com/p/perm/	Yes	Yes	Hash the genome	High sensitivity and speed contributed by the use of periodic spaced seeds with higher weight; no paired-end mapping available now	24
ProbeMatch	http://www.cs.wisc.edu/□jignesh/probematch/	Yes	No	Hash the genome	tolerant for gapped and ungapped alignments with up to 3 errors; uses gapped q-grams and q-grams of various patterns to identify target hits to a query sequence;	30
Slider	http://www.bcgsc.ca/platform/bioinfo/software/slider	Yes	No	Merge sorting	High alignment accuracy and efficiency; with probabilities while matching bases, it reduces the percentage of base mismatches; high SNP discovery rate	32
Slider II	http://www.bcgsc.ca/platform/bioinfo/software/slider	Yes	No	Merge sorting		32
Bowtie	http://bowtie.cbcb.umd.edu	Yes	Yes	BWT-based,	borrows a technique called Burrows-Wheeler transform(BWT), the	20

				index the genome	algorithm is more complicated than Maq's, but more than 30-fold faster	
BWA	http://bio-bwa.sourceforge.net/bwa.shtml	Yes	Yes	BWT-based, index the genome	implements two different algorithms, both based on Burrows-Wheeler Transform (BWT), the first algorithm is based on bwa-short for short queries up to ~200bp with low error rate(<3%) and supports paired-end reads, the second algorithm, BWA-SW, is designed for long reads with more errors.	25
SOAP2	http://soap.genomics.org.cn/#	Yes	Yes	BWT-based, index the genome	a updated version of SOAP, in super fast and accurate alignment for large amounts of short reads from illumina; supports a wide range of read length	19

Table 2 Results of mapping simulated illumina reads against human genome sequences(hg18)

Task	Tools	Reads mapped	Reads mapped correctly	Total processed time (m)	RAM (GB)
SE	Bowtie_0.12.5	11878078 (79.19%)	11857489 (79.05%)	271.37	5.09
	BWA_0.5.9	14416728 (96.11%)	13881061 (92.54%)	324.31	3.17
	Mosaik_1.1.0021	11774573 (78.50 %)	11641578 (77.61%)	315.26	20.61
	PASS_v1.2	1097876 (73.19%)	1050319(70.02%)	100.48	18.69
	RMAP_v2.05	11292461 (75.28%)	11261662 (75.08%)	397.845	6.1
	SeqMap_1.0.13	11878407 (79.19%)	11416970 (76.11%)	5049.433	8.01
	SHRiMP_2.0.1	14990830 (99.93%)	14442127 (96.28%)	9389.71	~32
	SOAP_2.2	11877778 (79.19%)	11800703 (78.67%)	96.61	8.25
	SSAHA2_v2.5.3	--	--	--	--
PE	Bowtie_0.12.5	9378024 (62.52%)	9370657 (62.47%)	332.5	5.10
	BWA_0.5.9	14919378 (99.46%)	14752604 (98.35%)	616.8	3.2
	Mosaik_1.1.0021	11777394 (78.52 %)	11638676 (77.59%)	576.8	20.67
	PASS_v1.2	--	--	--	--
	RMAP_v2.05	--	--	--	--
	SeqMap_1.0.13	--	--	--	--
	SHRiMP_2.0.1	14270212 (95.13%)	14150450 (94.34%)	15846.21	~32
	SOAP_2.2	9377074 (62.51%)	9364090 (62.43%)	116.27	12.63
	SSAHA2_v2.5.3	14675759 (97.84%)	14400877 (96.01%)	2884.5	13.38

Here, “SE” refers to Single-End reads mapping while “PE” stands for Paired-End reads mapping. The index “Total processed time” includes the time used for indexing genome or query sequences, the time used to splice genome or query sequences file (the whole genome sequence file or the query file has to be spliced into smaller ones when the RAM needed for a certain task exceeds the RAM available), and the time for mapping. “RAM” is measured as the maximum RAM used during the whole mapping process, including indexing and alignment.

Table 3 Results of mapping illumina real reads against human genome sequences (hg18)

Task	Tools	Reads mapped	Total processed time (m)	RAM(GB)
SE	Bowtie_0.12.5	10188613(80.09%)	308.77	5.09
	BWA_0.5.9	11279913 (88.67%)	236.36	3.17
	Mosaik_1.1.0021	10722310 (84.3 %)	351.63	20.67
	PASS_v1.2	1044693 (82.13 %)	120.60	20.15
	RMAP_v2.05	10104883 (79.44%)	366.54	5.62
	SeqMap_1.0.13	10323104 (81.15%)	5583.95	5.94
	SHRiMP_2.0.1	11037849 (86.77%)	8681.61	26.58
	SOAP_2.2	10201730(80.20%)	96.57	8.26
	SSAHA2_v2.5.3	--	--	--
PE	Bowtie_0.12.5	11001276 (61.29%)	505.4	5.15
	BWA_0.5.9	14440897 (80.46%)	614.26	3.17
	Mosaik_1.1.0021	14968995(83.4 %)	757.45	20.77
	PASS_v1.2	--	--	--
	RMAP_v2.05	--	--	--
	SeqMap_1.0.13	--	--	--
	SHRiMP_2.0.1	9581693 (53.38%)	19795.43	~32
	SOAP_2.2	10454273 (58.25%)	122.71	18.07
	SSAHA2_v2.5.3	12794188 (71.28%)	6635.5	14.36

Table 4 Tools for *de novo* assembly analysis

Program	Website	Strategy	NGS platforms	Overview	Ref
QSRA	http://qsra.cgrb.oregonstate.edu/	Greedy	Sanger, Solexa	**Q**uality-value guided **S**hort **R**ead **A**ssembler, it is created to take advantage of quality-value scores to handle base call errors	43
SHARCGS	http://sharcgs.molgen.mpg.de/index.shtml	Greedy	Solexa	**SH**ort-read **A**ssembler based on **R**obust **C**ontig extension for **G**enome **S**equencing, suitable for un-paired reads (25-40 bp) with high coverage	41
SSAKE	http://www.bcgsc.ca/platform/bioinfo/software/ssake	Greedy	Solexa (SOLiD? Helicos?)	**S**hort **S**equence **A**ssembly by progressive **K**-mer search and 3' read **E**xtension, with a prefix tree, it would progressively search for perfect 3'-most k-mers;	40
VCAKE	http://sourceforge.net/projects/vcake/	Greedy	Solexa (SOLiD?, Helicos?)	**V**erified **C**onsensus **A**ssembly by **K**-mer **E**xtension, by using high depth coverage, it could assemble millions of short reads even in the presence of sequencing error	42
CABOG	http://sourceforge.net/apps/mediawiki/wgs-assembler/index.php?title=Main_Page	OLC	Sanger, 454, Solexa	**C**elera **A**ssembler with the **B**est **O**verlap **G**raph, robust to homopolymer run length uncertainty, high read coverage and heterogeneous read lengths	44
Edena	http://www.genomic.ch/edena.php	OLC	Solexa	**E**xact **DE N**ovo **A**ssembler, based on **overlap layout paradigm**, uniform-length reads are indexed in a **prefix array** and all perfect, error-free contigs are produced	45
Newbler	http://contig.wordpress.com/	OLC	454, Sanger	particularly designed for 454 platforms, customs receive frequent updates, the source code is not generally available.	46
Shorty	http://www.cs.sunysb.edu/~skiena/shorty/	OLC	Helicos, Solexa, SOLiD	using a few (5-10) seeds of length 300-500 bp to assemble short-paired reads; can accurately estimate intercontig distance from multiple spanning mate pairs.	47
ABySS	http://www.ncbi.nlm.nih.gov/pubmed/19251739	DBG	Solexa, SOLiD	**A**ssembly **B**y **S**hort **S**equences, a parallelized sequence assembler	50
ALLPATHS	ftp://ftp.broadinstitute.org/pub/crd/ALLPATHS/	DBG	Solexa, SOLiD?	two key concepts in the algorithm: 1). *finding all paths* across a given read pair 2). localization, using pairs to isolate regions of the genome and assemble them	51
EULER-SR	http://euler-assembler.ucsd.edu/portal/	DBG	Sanger, 454, Solexa, SOLiD	**Euler**ian approach-based assembler, stated to be the assembler generating optimal short read assemblies of bacterial genomes	52
SOAPdenovo	http://soap.genomics.	DBG	Solexa	has been integrated into the **s**hort **o**ligonucleotide	53

	org.cn/soapdenovo.html			**a**lignment **p**rogram (SOAP) package; designed for large-genome assembly in a cost-effective way	
Velvet	http://www.ebi.ac.uk/~zerbino/velvet	DBG	Sanger, 454, Solexa, SOLiD	ideal for short reads(25-50bp) and paired-ends reads to produce contigs with significant length; tolerant color space reads;	54

Note: all the items in the fourth column, excluding Shorty and ALLPATH EULER-SR, which were further checked by the author, were cited from http://en.wikipedia.org/wiki/Sequence_assembly.

Table 5 Assembly results using real illumina single end and paired end reads from SRA

Task	Tools	Max contig length (bp)	Min contig length (bp)	Ave contig length(bp)	Genomic coverage	Total processed time (m)	RAM required (GB)
SE	QSRA	1577	76	76.37	63.71%	69.57	1.35
	SSAKE_v3-5	16652	77	126.90	0.34%	147.80	3.80
	Edena_2.1.1	1437	100	145.25	0.13%	18.77	0.37
	AByss_1.2.6	9020	25	32.13	4.13%	11.32	2.51
	SOAPdenovo_v1.05	2134	24	71.54	72.66%	4.05	2.07
	Velvet_1.0.09	1399	21	44.82	4.58%	136.08	4.24
Task	**Tools**	**Max contig length (bp)**	**Min contig length (bp)**	**Ave contig length(bp)**	**Genomic coverage**	**Total processed time (m)**	**RAM required (GB)**
PE	QSRA	--	--	--	--	--	--
	SSAKE_v3-5	4367	79	159.84	0.11%	540.06	8.51
	Edena_2.1.1	--	--	--	--	--	--
	AByss_1.2.6	12804	25	37.38	5.95%	31	9.61
	SOAPdenovo_v1.05	859	24	71.36	61.40%	9	4.12
	Velvet_1.0.09	2285	21	61.497765	17.47%	357.26	8.73

S1: Codes for evaluation experiments on mapping and assembly tools

Evaluation work on mapping tools		
SE	Bowtie	1).bowtie-build hg18.fa hg18 2).bowtie -t -p 8 -v 2 -a bowtie/hg18 -q ERR008834.filt.fastq >bowtie.map
	BWA	1).bwa index -a bwtsw hg18.fa 2).bwa aln -t 8 -M 2 hg18.fa ERR008834.filt.fastq > bwa.sai 3).bwa samse hg18.fa bwa.sai ERR008834.filt.fastq > bwa.sam
	Mosaik	1).MosaikBuild -fr hg18.fa -oa hg18.dat 2).MosaikJump -ia hg18.dat -out hg18_15 -hs 15 3).MosaikBuild -q ERR008834.filt.fastq -out ERR008834.dat -st illumine 4).MosaikAligner -in ERR008834.dat -out mosaikAligned.dat -ia hg18.dat -hs 15 -mm 2 -mhp 100 -bw 29 -act 20 -j hg18_15 -p 8
	PASS	pass -p 1111110111111 -pst PST/W7M1m0G0X0.pst 11 -flc 1 -fid 90 -g 5 -cpu 8 -query_size 1000 -i ERR008834.filt.fa -d . hg18.fa -gff -info_gff -o pass.gff
	RMAP	rmap -m 2 -o rmap.bed -c hg18.fa ERR008834.filt.fa –v
	SeqMap	seqmap 2 ERR008834.filt.fa hg18.fa seqmap.map /available_memory:30000 /output_statistics /no_store_key /do_not_output_probe_without_match /skip_N

	SHRiMP	# split genome:
		shrimp/utils/split-db.py --ram-size 25 --prefix hg18 hg18.fa
		# index:
		shrimp/utils/project-db.py --shrimp-mode ls hg18-25gb-*.fa
		# alignment:
		for((i=1; i<=2; i++))
		do
		shrimp/bin/gmapper-ls -L hg18-25gb-12_12_12_12seeds-${i}of2-ls ERR008834.filt.fa -N 8 -h 80% -E > shrimp.map.db${i}of2.sam

		done # merge results: shrimp/utils/merge-hits-same-qr-diff-db --unpaired --dest-file shrimp.map.sam shrimp.map.db?of2.sam
	SOAP	1).soap/2bwt-builder hg18.fa 2).soap -p 8 -r 2 -a ERR008834.filt.fastq -D hg18.fa.index -o soap.map
	SSAHA2	1).ssaha2/ssaha2Build -solexa -skip 6 -save hg18 hg18.fa 2).ssaha2/ssaha2 -solexa -skip 6 -output sam -outfile ssaha2.sam -save hg18 ERR008834.fastq
PE	Bowtie	1).bowtie-build hg18.fa hg18 2).bowtie -t -p 8 -v 2 -a –I 0 –X 1000 hg18 -1 SRR043391_1.filt.fastq -2 SRR043391_2.filt.fastq > bowtie.map
	BWA	1).bwa index -a bwtsw hg18.fa 2).bwa aln -t 8 -M 2 hg18.fa -1 SRR043391_1.filt.fastq > bwa.1.sai 3).bwa aln -t 8 -M 2 hg18.fa -2 SRR043391_2.filt.fastq > bwa.2.sai 4).bwa sampe hg18.fa bwa.1.sai bwa.2.sai SRR043391_1.filt.fastq SRR043391_2.filt.fastq > bwa.sam
	Mosaik	1). MosaikBuild -fr hg18.fa –oa Chg18.dat 2).MosaikJump -ia hg18.dat -out hg18_15 -hs 15 3).MosaikBuild -q SRR043391_1.filt.fastq -q2 SRR043391_2.filt.fastq -out SRR043391.dat -st illumine 4).MosaikAligner -in SRR043391.dat -out mosaikAligned.dat -ia hg18.dat -hs 15 -mm 2 -mhp 100 -bw 29 -act 20 -j hg18_15 -p 8

	SHRiMP	# split genome:
		shrimp/utils/split-db.py --ram-size 25 --prefix hg18 hg18.fa
		# index:
		shrimp/utils/project-db.py --shrimp-mode ls hg18-25gb-*.fa
		# alignment:
		for((i=1; i<=2; i++))
		do
		shrimp/bin/gmapper-ls -L hg18-25gb-12_12_12_12seeds-${i}of2-ls SRR043391_1-2.filt.fa -p opp-in -N 8 -E > shrimp.map.db${i}of2.sam

		done #merge results: shrimp/utils/merge-hits-same-qr-diff-db --paired --dest-file shrimp.map.sam shrimp.map.db?of2.sam
	SOAP	1).soap/2bwt-builder hg18.fa 2).soap -p 8 -r 2 -a SRR043391_1.filt.fa -b SRR043391_2.filt.fa -D hg18.fa.index -o soap.PEmap -2 soap.SEmap -m 0 -x 1000
	SSAHA2	1).ssaha2Build -solexa -skip 6 –save hg18 hg18.fa 2).ssaha2 -solexa -skip 6 -pair 0,1000 -output sam -outfile mapped.sam -save hg18 SRR043391_1.filt.fastq SRR043391_2.filt.fastq
Evaluation work on assembly tools		
SE	QSRA	qsra -f ERR008834.filt.fa -k 76
	SSAKE	SSAKE -f ERR008834.filt.fa -p 0
	Edena	1). edena -r ERR008834.filt.fa -p ERR008834.edena 2). edena -e ERR008834.edena.ovl -p ERR008834.edena
	AByss	ABYSS -k25 ERR008834.filt.fa -o abyss.contigs.fa
	SOAPdenovo	SOAPdenovo31mer all -s soap1.config -o soapSE
	Velvet	1). velveth VelvetResult 21 -long ERR008834.filt.fa 2). velvetg VelvetResult >velvetSE.log
PE	SSAKE	SSAKE -f SRR043391_1_2.filt.fa -z 20 -m 17 -o 4 -r 0.7 -p 1 -c 1 -e 0.75 -k 2 -a 0.6
	AByss	abyss-pe k=25 n=5 in='SRR043391_1.filt.fa SRR043391_2.filt.fa' name=abyssPE
	SOAPdenovo	SOAPdenovo31mer all -s soap2.config -o soapPE
	Velvet	1). velveth VelvetResult 21 -fasta -long SRR043391_1.filt. fa -long SRR043391_2.filt. fa 2). velvetg VelvetResult -ins_length 1000 -exp_cov auto>velvetPE.log

S2: 454 and Solid reads files for program testing

Roche/454		
	SE	**PE**
Acc No.	SRR033700- SRR033709	SRR081266
Reads	10849703	12500000x2
Reads length	-	-
Source URL	http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?cmd=viewer&m=data&s=viewer&run=SRR033700	http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?cmd=viewer&m=data&s=viewer&run=SRR081266
AB/Solid		
	SE	**PE**
Acc No.	SRR010631	SRR001662
Reads	12720049	12000000x2
Reads length	35	25
Source URL	http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?cmd=viewer&m=data&s=viewer&run=SRR010631	http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?cmd=viewer&m=data&s=viewer&run=SRR001662

S3: Results of tests on 454 and Solid-supported tools

Task	Tools	Reads mapped	Total processed time (m)	RAM(GB)
454SE	Mosaik_1.1.0021	3862664 (35.6 %)	2134.27	20.64
	SSAHA2_v2.5.3	10833772 (99.85%)	3834.22	15.30
	PASS_v1.2	10678503 (98.42%)	6284.583	19.20
454PE	Mosaik_1.1.0021	12020933 (96.2 %)	517.45	20.64
	SSAHA2_v2.5.3	8776079 (70.20%)	2657.35	14.26
	PASS_v1.2	--	--	--
SolidSE	Bowtie_0.12.5	4755394 (37.39%)	487.2	10.12
	Mosaik_1.1.0021	--	--	>32
	SHRiMP_2.0.1	5945467 (46.74%)	309.73	28.15
	PASS_v1.2	6532777 (51.36 %)	177.95	18.69
SolidPE	Bowtie_0.12.5	3881 (0.03%)	37.57	2.86
	Mosaik_1.1.0021	--	--	>32
	SHRiMP_2.0.1	736148 (6.13%)	376.78	~32
	PASS_v1.2	--	--	--