SCIENCE CHINA
Life Sciences

THEMATIC ISSUE:
• RESEARCH PAPER •

November 2014 Vol.57 No.11: 1–9
doi: 10.1007/s11427-014-4752-9

Sequence assembly using next generation sequencing data—challenges and solutions

CHIN Francis Y.L.*, LEUNG Henry C.M.* & YIU S.M.*

Department of Computer Science, The University of Hong Kong, Hong Kong, China

Received May 29, 2014; accepted August 25, 2014

Sequence assembling is an important step for bioinformatics study. With the help of next generation sequencing (NGS) technology, high throughput DNA fragment (reads) can be randomly sampled from DNA or RNA molecular sequence. However, as the positions of reads being sampled are unknown, assembling process is required for combining overlapped reads to reconstruct the original DNA or RNA sequence. Compared with traditional Sanger sequencing methods, although the throughput of NGS reads increases, the read length is shorter and the error rate is higher. It introduces several problems in assembling. Moreover, paired-end reads instead of single-end reads can be sampled which contain more information. The existing assemblers cannot fully utilize this information and fails to assemble longer contigs. In this article, we will revisit the major problems of assembling NGS reads on genomic, transcriptomic, metagenomic and metatranscriptomic data. We will also describe our IDBA package for solving these problems. IDBA package has adopted several novel ideas in assembling, including using multiple *k*, local assembling and progressive depth removal. Compared with existence assemblers, IDBA has better performance on many simulated and real sequencing datasets.

keywords

Citation: Chin FYL, Leung HCM, Yiu SM. Sequence assembly using next generation sequencing data—challenges and solutions. Sci China Life Sci, 2014, 57: 1–9, doi: 10.1007/s11427-014-4752-9

Deoxyribonucleic acid (DNA) is a sequence of nucleotides adenine (A), cytosine (C), guanine (G) and thymine (T) which is used to encode all genetic information for controlling development and functioning of most organisms in the world except some virus. Each cell in the same organism contains near identical set of DNA sequences which can be used to represent the organism. The whole set of DNA in a cell for storing genetic information is called genome. Along the genome, there are regions called gene, which can be copied to form a ribonucleic acid (RNA), a sequence of nucleotides adenine (A), cytosine (C), guanine (G) and uracil (U). The RNA can support most metabolic activities in the cell either directly or indirectly by decoding itself to different proteins. The whole set of RNAs produced in a cell is called transcriptome. Determining the nucleotide sequences for the genomes and transcriptomes of different organisms is important for analyzing different biological properties of the organisms, determining the interaction of the organisms with the environment and other organisms, determining the evolutionary relationship for a set of organisms. In particular, sequencing the human genome can be applied for determining different genomic diseases, personal medicine for each individual and human evolutionary history.

Determining the whole genome or transcriptome of an organism is difficult. Early work on sequencing was done on single short RNA sequence only. In 1964, Holley et al. [1,2] determined the first RNA sequence (alanine transfer RNA) with less than 100 nucleotides (nt). In 1972, Fiers et

*Corresponding author (email: chin@cs.hku.hk; cmleung2@cs.hku.hk; smyiu@cs.hku.hk)

al. [3] determined the gene for Bacteriophage MS2 coat protein. In 1976, Fiers et al. [4] determined the complete genome (RNA) of Bacteriophage MS2. The first DNA genome was determined in 1977 by Sanger et al. [5] on the Bacteriophage φX174 with 5368 nt. After that, the Sanger et al. developed shotgun sequencing technology was applied on different organisms with longer and longer genomes. In 2003, the first whole human genome (with three billion nucleotides) was determined after 13 years' analysis.

Determining genome or transcriptome of an organism can be divided into two major steps, sequencing and assembling. Sequencing step randomly samples fragments, called read, from a DNA or RNA sequence (or fragment of a DNA or RNA after preprocessing) and determines the sequence of the DNA fragments. Assembling step analyzes the set of fragments sampled from unknown locations and determines the original DNA or RNA sequence (or fragments of DNA or RNA if not possible). The traditional shotgun sequencing technology can sample a read of length around 1000 nt per hour with 0.1% error per nucleotide. Since the throughput of Sanger sequencing is low compared with the length of genome (the length of human genome is three billion and the lengths of some plant and fish genomes can be over 100 billion), it is not suitable for sequencing long genome because of the high cost of sequencing enough reads.

In recent years, several next generation sequencing (NGS) technologies were developed, e.g., pyrosequencing (454), sequencing by synthesis (Illumina), ion semiconductor (ion torrent sequencing). These NGS technologies can produce shorter reads (75–500 nt) with higher throughput (million or even billion reads per day) but higher error rate (0.5%–2% error per nucleotide). With the help of the NGS technology, it is possible to determine more genomes in a much faster way. However, it also introduces new challenges in the assembling process. When assembling genomic and transcriptomic data, there are multiple problems due to high sequencing error rate, repeat patterns in the sequence and uneven sequencing depths[1)]. Some of these problems appears in both genomic and transcriptomic data while some problems appear in particular data type. Moreover, the high throughput of NGS technologies provides an opportunity to sequence sample with multiple micro-organisms (metagenomic and metatranscriptomic data), e.g., group of bacteria, such that scientists can understand the interactions among micro-organisms. However, assembling these metagenomic and metatranscriptomic data also introduces more challenges in assembling. In this article, we will first describe the approaches of common assemblers in Section 1. Then we will explain the general problems of assembling NGS data (Section 2) and the possible solutions introduced by IDBA package (Section 3). Finally we will discuss the specific problems of assembling transcriptomic, metagenomic and metatranscriptomic data in Sections 4, 5 and 6 respectively with some experimental results in Section 7. IDBA package (http://www.cs.hku.hk/~alse/hkubrg/) is a collection of software based on IDBA for different sequencing data. A short summary is given in Table 1.

1 Existing approaches

Since the location of each read in the DNA or RNA sequence is unknown, assembling process is needed to combine these reads into the original sequence. As a huge number of reads are sampled randomly along the sequence, the overlapping information of the sampled reads is used for assembling. There are three major approaches for assembling reads: (i) overlap-and-extend, (ii) string graph, and (iii) de Bruijn graph.

The overlap-and-extend approach, e.g., SSAKE [6], VCAKE [7] and SHARCGS [8], first determines overlapped reads, i.e., the suffix of a read is the same as the prefix

Table 1 Summary of IDBA package for solving different assembling problems[a)]

Data type	Solutions	Description
Genome	IDBA-UD	A general tool for genomic and metagenomic data. It can also handle single-cell sequencing data.
	IDBA-Hybrid	Extension of IDBA for assembling genomic data with a similar reference genome.
Transcriptome	IDBA-Tran	Extension of IDBA for assembling eukaryotic transcriptomic data.
Metagenome	IDBA-UD	Mentioned above
	Meta-IDBA	Extension of IDBA for assembling metagenomic data. Multiple strands from the same species will be merged in a single consensus representation.
Metatranscriptome	IDBA-MT	Extension of IDBA for assembling prokaryotic metatranscriptomic data. It applies paired-end read data for resolving chimeric contigs.
	IDBA-MTP	Extension of IDBA for assembling prokaryotic metatranscriptomic data. It assembles reads by applying known protein reference sequences.

a) The software can be downloaded at http://www.cs.hku.hk/~al se/hkubrg/.

1) Uneven sequencing depths problem also occurs in traditional Sanger sequencing. However, this problem is more serious in the next-generation sequencing technology especially when performing single-cell sequencing.

of another reads and the length of overlapped region is longer than a predefined threshold, and then extend the first read using the overlapped read to construct a longer read. This process is repeated until either (i) there is no overlapped read, or (ii) there are more than one overlapped reads and the extensions by these overlapped reads are not consistent. There are several problems when overlap-and-extend approach is applied on NGS reads. (i) There is higher sequencing error rate on reads which introduces errors on extension and prevents some true positive overlap. (ii) Data structure such as prefix tree is needed to determine the overlapped reads. As the data structure is large and there is huge number of reads, a large amount of memory is required for the assembling process. (iii) Finding overlapped reads from huge number of reads is time-consuming.

Instead of considering each read separately, the string graph approach, e.g., Edena [9] and BOA [10], considers a global view of all reads when assembling. These assemblers construct a string graph for the reads in which each read is represented by a vertex and there is a directed edge from vertex u to vertex v if read u overlaps with read v, i.e., the suffix of at least k nucleotides of read u is the same as the prefix of read v. The value of k is the number of overlapping nucleotides for the two consecutive reads. If there are no errors and repeated patterns in the genome, the string graph should be a simple path without any branches, i.e., every vertex, except the first and the last vertices, points to (and is pointed by) exactly one vertex. However, because of the existence of sequencing errors and repeated patterns, string graph algorithms only report maximal paths without branches as contigs (fragments of genome). Similar to the overlap-and-extend approach, since the data structure used for storing the string graph is large, it requires a large amount of memory and takes a long time for finding overlapped reads.

The de Bruijn graph approach, e.g., Velvet [11], Abyss [12], IDBA [19,21,22–25], Euler-SR [13,14] and AllPaths [15], constructs a de Bruijn graph for the reads in which each vertex represents a length-k substring (k-mer) in a read and there is a directed edge from vertex u to vertex v if u and v are consecutive k-mers in a read, i.e., the last k–1 nucleotides of the k-mer represented by u is the same as the first k–1 nucleotides of the k-mer represented by v. Similar to the string graph approach, maximal paths without branches in the graph corresponding to contigs are outputted. There are two main advantages against the other two approaches. (i) Utilize information in erroneous reads. When a read has errors, it is difficult to use the read information for assembling using the above two approaches. Consider a length-100 read with 1% sequencing error rate, the probability of sampling an erroneous read is $1-(1-1\%)^{100}=0.63$, i.e., 63% of sampled reads will be wasted. The main challenge is that we do not know which reads are erroneous. However, since at most k k-mers overlap with an erroneous nucleotide, consider a length-l reads with l–k+1 k-mers, there should be at least l–2k+1 correct k-mers obtained from the read for assembling even there is one error in the read. (ii) Less memory consumption. Since the k-mer can be stored effectively using a hash table [16] or burrows-wheeler transform [17,18] and the storage is independent of the number of reads. As a result, de Bruijn graph approach is widely used in assembling NGS reads.

When all reads are error-free and the number of sequenced reads is large compared with the genome length (high sequencing depth), both string graph and de Bruijn graph approaches work well. However, because of the existence of erroneous reads and the repeated patterns exist in the genome, these two approaches may not perform well on some sequencing data.

2 General problems of assembling NGS data

2.1 False positive vertices

Errors in reads introduce false positive vertices which make the string graph and de Bruijn graph larger and more complicated. Besides consuming more memory for storage, the false positive vertices also introduce false positive edges in the graph which introduce more branches and reduce the lengths of simple paths, thus resulting in shorter contigs. For example, considering a human genome with 3G nucleotides and assuming no sequencing errors, there should be at most 3G different reads and fewer than 50G memory will be needed. However, for a typical human genome data with 30× sequencing depth and around 1% sequencing error, the memory requirement of Velvet [11] and Abyss [12] is more than 250 G. Simulated data of *E. coli* shows that when the sequencing error rate increases from 1% to 2%, the N50 of contigs produced by Velvet and Abyss reduces by 33% and 50% respectively [19].

2.2 Gap problem

Due to non-uniform or low sequencing depth for some sequencing data by the NGS technologies, reads may not be sampled for every position in the genome. For the string graph and de Bruijn graph approaches, when all the (possible l–k for length-l read) reads covering consecutive length-k region (k-mer) are missing, there will be a true positive edge missing in the graph and the contig which originally contains this edge will be shortened. A large k value will make the gap problem more serious because it will reduce the number of required missing l–k reads covering a particular region. Therefore, using a small k can prevent the gap problem.

2.3 Branching problem

If there is a repeat region in the genome with length at least k and the reads covering and near this repeat region are

sampled, there will be branches adjacent to the node containing or representing the repeat regions in the string graph and de Bruijn graph. As mentioned before, erroneous reads may also introduce branches in the graph. Since many assemblers [11−13] will produce contigs which stop at branches and it is not possible to extend the length of a contig beyond without additional information, branching problem will lead to shorter contigs (if the assemblers stop at branches) or erroneous contigs (if the assemblers extend beyond branches). Since using small k will lead to more branches resulting from short repeat regions, large k can be used to avoid serious branching problem even though this problem is not solvable if the length of repeat region is longer than the read length.

2.4 Under utilization of paired-end reads information

Besides sequencing each reads independently (single-end read), current NGS technology allows sequencing a pair of reads, called paired-end reads, separated by an approximated distance (insert distance) in the genome. Mate-pair reads are called for longer insert distance. Since paired-end reads and mate-pair reads play similar roles in the assembling process, without loss of geniality, paired-end reads stand for both in this paper. Since the relative positions of the paired-end reads are known, existing assemblers [11−13,15,19] usually align paired-end reads to different contigs and use the paired-end reads information to determine the relative positions of the contigs, called scaffolds. Although paired-end reads can be used to construct scaffolds, the information of the paired-end reads can also be used in assembling as they can be used to resolve branches in the graph. Some assemblers [20] try to apply paired-end reads information directly on assembling by considering a rectangle graph instead of de Bruijn graph. A contig can be represented by some traversal of the rectangle graph with the insert distance restrictions. Although the rectangle graph-based algorithm might be able to produce longer contigs for small error in insert distance, it fails to construct long contigs in practice as the insert distance usually varies a lot.

3 Solution for assembling NGS reads

We have developed an IDBA (iterative de Bruijn graph short read assembler) package based on de Bruijn graph approach [19,21,22−25] to solve those problems mentioned in Section 2 and construct longer and more accurate contigs from NGS reads. In this section, we will describe several novel ideas in the genome assembler IDBA-UD [22] in the package addressing each of the above-mentioned problems including (i) filtering erroneous k-mers for small k by multiplicity; (ii) using multiple k from small to large values iteratively; (iii) local assembling using paired-end reads information; (iv) reads and contigs correction. Table 2 summarizes which problems are solved by these novel ideas. We will describe each idea one by one in this section. Figure 1 shows the workflow of IDBA-UD for assembling genome.

3.1 Filtering short k-mers by multiplicity

As mentioned before, although the error rate is low, the number of erroneous reads can still be larger than the number of correct reads. The number of erroneous k-mers can be much larger than the number of correct k-mers. Since the

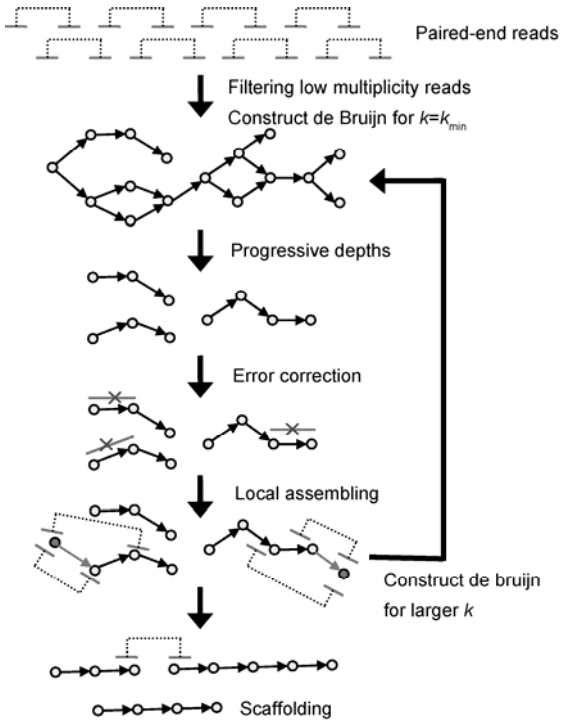

Figure 1 Flowchart of main steps in IDBA package.

Table 2 Summary of the ideas of IDBA package for solving the problems in assembling

Problems	Solutions
False positive vertices	Filtering short k-mers for small k by multiplicity; reads and contigs correction
Gap problem	Using multiple values of k iteratively
Branching problem	Using multiple values of k iteratively; local assembling
Paired-end read information	Local assembling

same correct *k*-mers at a particular position may be sampled by different reads while erroneous *k*-mers might only be sampled once or twice, erroneous *k*-mers are unlikely to be generated multiple times and they can be filtered out based on their low multiplicities.

In IDBA-UD, k_{min}-mers (k_{min}=25) appear no more than m times (m=1) will be removed by default. The values of k_{min} and m can be user-specified depending on the sequencing depth and genome length. As shown in [19], the probability that a correct *k*-mer being removed can be as low as 1.14×10^{-9} (expected number of false removal is 0.0047<<1) when sampling 1.6 M length-75 reads with 1% error rate from a genome of length 4.1 M (45× coverage) while 50%–80% of erroneous *k*-mers can be removed. Thus, IDBA can much reduce the memory usage for storing the erroneous *k*-mers and the complexity of the de Bruijn graph. Other assemblers or read correction algorithms may apply similar approach for filtering or correcting erroneous *k*-mers or reads. For an effective filtering of erroneous *k*-mers or reads, a small *k* is required. However, in order to prevent branching problem, the assemblers need to apply a large *k* to construct de Bruijn graph which reduces the effectiveness of the error filtering steps. Read correction algorithm can apply shorter *k*-mers, say *k*=15, for filtering erroneous *k*-mers and correcting the input reads using a set of confident *k*-mers [26]. However, this process is time-consuming and might not be effective as each read will be determined independently without considering its overlap with others.

3.2 Multiple *k*

The regions in a genome can be classified into five classes according to their sequencing depth and the existence of repeat patterns (Table 3). Regions with high sequencing depth and no repeat pattern can be assembled easily independent of the value of *k* (A). Regions with low sequencing depth and no repeat pattern can be assembled using a small *k* (but not a large *k*) because using large *k* will introduce a gap problem (B). On the other hand, regions with high sequencing depth and some short repeat patterns can be assembled using a large *k* (but not a small *k*) because using large *k* can overcome the branching problem introduced by the short repeat patterns (C). Regions with low sequencing depth and short repeat patterns are difficult to assemble using single-end reads (D). Regions with repeat patterns longer than the read length are unsolvable using single-end reads (E). In this section, we will describe how to assemble region A, B and C using single-end reads. We will describe how to assemble regions D and E using paired-end reads later.

Since small *k* value helps solve the gap problem while a large *k* value helps solve the branching problem, existing algorithms usually apply a single *k* in between as a trade-off between these two problems. Thus these algorithms can always assemble the region A, but only part of the region B and part of the region C depending on the value of the chosen *k*. Instead of using one single *k* value, IDBA-UD applies multiple *k* from small to large for constructing the de Bruijn graph and assembling the reads. By combining the advantages of different *k* values, IDBA-UD can assemble region A, B and C successfully.

Table 3 Classification of genome regions

	A	B	C	D	E
Sequencing depth	High	Low	High	Low	Any
Repeat	No	No	Short	Short	Long
Best *k*	Any	Small	Large	No	No

IDBA-UD first constructs a de Bruijn graph $G_{k\text{min}}$ based on the k_{min}-mer, say k_{min}=25, from the input reads after filtering the erroneous k_{min}-mers by multiplicity. Since the value of *k* is small, the gap problem is not serious and the genome should be represented by a path in the de Bruijn graph. However, the branching problem may be very serious (Region C) as any repeat patterns with length at least k_{min}–1 will introduce branches in the de Bruijn graph. Although the branching problem is serious, regions A and B can be assembled easily and contigs $C_{k\text{min}}$ can be constructed to represent these regions. Then IDBA-UD increases the value of *k*, say $k=k_{min}+1$ and using the (k_{min}+1)-mers in the reads and contigs $C_{k\text{min}}$ to construct another de Bruijn graph $G_{k\text{min}+1}$. Since the value of *k* increases, some of the branching problems (repeat patterns with length k_{min}–1) may be solved, i.e., part of the region C can be assembled. Since (k_{min}+1)-mers obtained from the contigs in $C_{k\text{min}}$ representing low sequencing depth region B are connected in $G_{k\text{min}+1}$, there is no gap problem for a larger *k* value. Then IDBA repeats this process by increasing the value of *k* to k_{min}+2 and constructs a de Bruijn graph $G_{k\text{min}+2}$ using (k_{min}+2)-mers in the reads and contigs $C_{k\text{min}+1}$. By iteratively increasing the value of *k* (up to $k_{max}\leq$ read length), region B can be assembled without losing regions A and C of the genome.

In practice, IDBA-UD needs not reconstruct a de Bruijn graph in each step, G_{k+1} can be constructed by updating G_k, and reads appearing in contig C_k can be removed from the input. As a result, the speed of IDBA-UD is similar to the performance of existing algorithms [19]. Moreover, we can jump from *k* to *k*+*s* instead of considering every *k* between k_{min} and k_{max} to speed up the assembling process. As using too large *s*, say *s*>20, may reduce the quality of contigs, it is suggested to use $s\leq20$ as a trade-off between the efficiency of the algorithm and the quality of the contigs.

3.3 Local assembling

Repeat patterns will introduce branches in the de Bruijn graph. In order to determine which branches a path should extend, single-end reads containing the repeat with two ends representing unique patterns near the repeat can be used to resolve the branches. Although IDBA-UD can utilize the

single-end read information to resolve some of the repeat patterns by applying multiple k values, there are some repeat patterns in low sequencing depth regions (region D) which are covered by very few or even no reads and some repeats longer than the read length (region E). As a result, these regions cannot be assembled using single-end reads only. However, since the insert distance (300–5000) of paired-end reads can be longer repeats, they could be used to resolve branches during assembling.

Consider a repeat pattern R longer than read length occurring twice in the genome (region E) adjacent to unique regions U_1 and U_2 on the left and unique regions U_3 and U_4 on the right (Figure 2 shows the corresponding de Bruijn graph). Since we cannot determine whether the pair of contigs U_1RU_3 and U_2RU_4 or the pair of contigs U_1RU_4 and U_2RU_3 occur in the genome using single-end reads, five contigs representing R, U_1, U_2, U_3 and U_4 will be obtained. If there are two paired-ends P_1 and P_2, where the two ends of P_1 can be aligned to U_1 and U_3 and the two ends of P_2 can be aligned to U_2 and U_4, then we can resolve the branches and determine that the contigs U_1RU_3 and U_2RU_4 are correct. A similar but more complicated situation is that region R represents a series of short repeats each of which occurs multiple times along the genome. When the sequencing depth is low (region D), the series of short repeats in region R cannot be determined easily because of the existence of a huge number of branches in the de Bruijn graph representing the interconnected short repeats. Consider there are two unique regions U_1 and U_3 adjacent to region R in the genome, we can divide those paired-end reads with one end aligned to the short repeats in region R into two groups: (i) those paired-end reads with the other ends aligned to R, U_1 or U_3 (which should be sampled from the region U_1RU_3); (ii) those paired-end reads with the other ends cannot be aligned to U_1, R nor U_3 (which should be sampled from other regions in the genome). Thus, when determining the sequence in region R, only paired-end reads in (i) should be considered. By reducing the number of reads, the de Bruijn graph representing region U_1RU_3 will become less complicated (without any branches) and the sequence of region R can be determined.

IDBA-UD adopts the above idea by introducing the local assembling step in each iteration of k. After constructing a set of contigs C_k using single-end reads, the input paired-end reads will be aligned to each contig in C_k, those paired-end reads aligned to the same contig c will be grouped together with contig c to construct a smaller and simpler de Bruijn graph. Since the branching problem is less serious in this smaller de Bruijn graph, longer contigs representing regions near or adjacent to contig c can be assembled. The contigs assembled in the local assembling step will be mixed with C_k and be used for constructing de Bruijn graph with larger k, say G_{k+1}. Since paired-end information is used, repeats whose lengths are longer than read length can be resolved and k-mers with length longer than read length could be constructed for resolving branches. As a result, the value of $k_{\max}$ can be increased to or more than the read length.

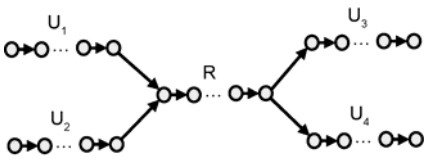

Figure 2 Example of de Bruijn graph for long repeat.

3.4 Reads and contigs correction

Although some erroneous k-mers are filtered based on multiplicity, there are still some unfiltered errors in reads and thus contigs. During assembling process, since erroneous k-mers can be removed using topological based methods such as removing tips (short dead-end path, say shorter than 2× read length, in the de Bruijn graph) and merging bubbles (combining two paths in the de Bruijn graph representing similar sequences), the contig sequences are usually more accurate than a read. Thus, by aligning reads to the contigs, errors in reads can be detected and be corrected. For each position in a contig, IDBA-UD will determine whether the nucleotide in that position is "confident" by checking the number of reads aligned to the contigs covering that position. If over 80% of aligned reads have the same nucleotide at a particular position, that position is confident and aligned reads with different nucleotides at that position will be corrected. Note that reads which can be aligned to multiple contigs will not be counted.

Besides correcting reads by contigs, corrected reads can also assist better assembling and construct more accurate contigs. IDBA-UD provides an optional pre-error-correction step for correcting reads before assembling. A medium k value and filtering threshold will be used to assemble reads to construct contigs. Erroneous reads are corrected based on its alignment with the contigs. These corrected reads will then be used for assembling as in the normal process.

4 Problems and solutions for assembling transcriptomic data

Transcriptomic data contains multiple RNAs produced by the same species. The RNA sequences are different and mixed. The NGS technology can only sample reads from the mixture of RNAs without knowing the association between a read and the RNAs.

4.1 Problems of assembling transcriptomic data

Compared with assembling genomic data, there are two

additional problems in assembling transcriptomic data. (i) Different expression levels of RNAs. The expression levels of different RNAs can vary a lot (three orders of magnitude, i.e., 1−1000). Thus, some RNAs appear with high abundance in the sample while some RNAs appear sparingly. Since reads are uniformly sampled from the RNAs, more reads are sampled from the high-expressed RNAs while fewer reads are sampled from the low-expressed RNAs. As a result, the number of erroneous *k*-mers and reads sampled from high-expressed RNAs can be much more than the number of correct *k*-mers and reads sampled from low-expressed RNAs. The erroneous *k*-mers with high multiplicity cannot be filtered easily and introduces many branching problems. On the other hand, the correct *k*-mers with low multiplicity may be filtered and introduce the gap problems. (ii) Alternative splicing. Since different regions, called exons, of a single gene can be copied to form different RNAs through alternative splicing, RNAs (isoforms) constructed from the same genes can have very similar sequences with long repeat regions. These long repeats cannot be easily solved using local assembling or multiple *k*.

4.2 Solutions for assembling transcriptomic data

Isoforms from the same gene should be represented by a connected component in the de Bruijn graph. However, because of the existence of erroneous reads, the components representing different genes may be connected in the de Bruijn graph and cannot be separately easily. IDBA-Tran [25] models the multiplicities of *k*-mers in the same component by a multi-normal distribution which depends on the expression levels of the corresponding isoforms. Based on the multi-normal distribution, erroneous *k*-mers in the component with relative low multiplicity can be determined and be removed. By removing the erroneous *k*-mers, large components representing multiple genes can be divided into smaller components. New multi-normal distributions can be learned from the smaller components and erroneous *k*-mers in these small components can be determined and be removed. By repeating this step, IDBA-Tran obtains a number of small components each of which represents a single gene.

Since each paired-end read should be sampled from a single isoform, an isoform should be represented by a path in a small component with supports from multiple paired-end reads (paired-end read aligned to the path with the distance of the two ends matches with the insert distance). For each small component, IDBA-Tran recovers the isoform sequences by searching paths in the component with maximum support from the paired-end reads. As the number of branches in each small component is small, this search process is efficient and the resultant contigs are accurate compared with searching paths in the original de Bruijn graph.

5 Problems and solutions for assembling metagenomic data

Metagenomic sample contains multiple genomes from different species. During the sequencing process, reads are sampled from these genomes without any information of which genome a particular read comes from.

5.1 Problems of assembling metagenomic data

Besides the problems mentioned in Section 2, there are two additional problems when assembling metagenomic data. (i) Repeat patterns from similar species. Species with similar genomes, say subspecies from the same species, may appear in the sample. There are many common patterns (repeats) among the similar genomes. As a result, there are more and longer repeats in the metagenomic data than the genomic data. The repeats will connect the de Bruijn graph for different species into a larger and more complicated de Bruijn graph. (ii) Different abundances. The abundances of different species can vary (two or more orders of magnitude). Similar to assembling transcriptomic data, the erroneous *k*-mers cannot be filtered easily because erroneous *k*-mers sampled from high abundance species have higher multiplicity than correct *k*-mers sampled from low abundance species.

5.2 Solutions for assembling metagenomic data

The repeat problem can be solved partially using the multiple *k* and local assembling technology as in the genomic data. However, as the length of repeats can be longer than the insert distance, those long repeats are difficult to resolve completely. Instead of resolving all repeats, Meta-IDBA [21] proposes to distinguish those repeats occurring in different subspecies of the same species and the repeats occurring in different species. Since similarity of the genomes of different subspecies of the same species is high, there are many repeats among them and the distances of adjacent repeats in the genomes are short. For the genomes of different species, although there are repeats, there are relatively fewer repeats among different species and these repeats are further apart in the genomes. When considering the de Bruijn graph of different subspecies of the same species, the path between a repeat and its closest repeat is short while in the de Bruijn graph of different species, the path between a repeat and its closest repeat is long. Based on this property, the repeats among different species can be determined and resolved such that the de Bruijn of different species can be separated.

In order to solve the abundance problem, IDBA-UD [22] applies another approach for separating the de Bruijn graph to smaller components representing genomes of different species. Although the sequencing depths may vary a lot along the genome, sequencing depths for neighboring re-

gions in the genome should change slowly and are similar. Even the multiplicities of some *k*-mers within a high sequencing depth region are higher than the global average multiplicity, they may be erroneous if they have relatively much lower multiplicity than the nearby *k*-mers. Similarly, when the multiplicities of some *k*-mers within a low sequencing depth region are lower than the global average multiplicity, these *k*-mers may not be erroneous. Based on this idea, instead of using a global average of the multiplicity of all *k*-mers, IDBA package adopts variable relative thresholds depending on the sequencing depths of their neighboring contigs. Short contigs with relative low multiplicity than adjacent contigs will be removed progressively (repeatedly remove those contigs with lowest multiplicity first and recalculate the average multiplicity of the rest contigs).

6 Problems and solutions for assembling metatranscriptomic data

Metatranscriptomic data contain reads sampled from different RNAs from different species without knowing the association of read to RNA or species. As the number of RNAs is large (compared with transcriptomic data) and the abundances of RNAs vary a lot (large variation than metagenomic and transcriptomic data). It is very difficult to assemble metatranscriptomic data.

6.1 Problems of assembling metatranscriptomic data

Assembling metatranscriptomic data has similar problems as assembling genomic, transcriptomic and metagenomic data but the problems are more serious. (i) Repeats problem. RNAs from different species with similar function usually have similar sequences. As some RNA functions are essential for life, many RNA sequences have similar patterns even they are from different species. This large amount of repeats cannot be solved by the above methods completely. (ii) Huge variations in abundances. Since the abundances of different species vary and the expression levels of different RNAs from the same species also vary, the abundances of different RNAs in the sample can vary a lot. Because of these problems, existing assemblers usually combine different regions of different RNAs into a single wrong contig, called chimeric contig [23].

6.2 Solutions for assembling metatranscriptomic data

IDBA-MT [23] and IDBA-MTP [24] solve the chimeric contigs by two different approaches. When the chimeric contigs are constructed from RNAs with different abundances, the multiplicities of the *k*-mers along the chimeric contig usually vary a lot. Moreover, since paired-end reads are sampled from the same RNA, there should be no paired-end reads with two ends aligned to regions from different RNAs. Although the above two ideas can be applied to determine chimeric contigs, the false positive rate is still high and many correct contigs are considered as chimeric contigs. IDBA-MT applies a probability model, based on both multiplicity and paired-end reads information, to determine chimeric contigs with higher true positive and lower false positive than the straightforward approach.

IDBA-MTP applies another approach to resolve chimeric contigs. A major type of RNA is messenger RNA (mRNA) which can be decoded to produce proteins. Since the protein sequences of many microbes are known and are similar, IDBA-MTP constructs contigs in de Bruijn by aligning known protein sequence to the de Bruijn graph. Given a protein sequence, IDBA-MTP finds a path in the de Bruijn graph representing an mRNA such that the decoded protein sequence is similar to the input protein sequence. By using known protein sequence information, the repeat problem and varying abundance problem can be greatly alleviated.

Table 4 The assembly results on simulated 10× lenght-100 reads of *Lactobacillus plantarum* (~3.3 Mb) with 1% error rate

	k	Contigs						Scaffolds						Time	Mem
		No.	N50	Max len	Cov (%)	Sub. err	err # (len)	No.	N50	Max len	Cov (%)	Sub. err	err # (len)		
IDBA-UD	20–100	210	36513	201860	99.56	0.0225%	10 4437	83	194322	406269	99.55%	0.0218%	5 3784	63 s	432 M
SOAPdenovo	31	3346	1584	8691	98.36	0.0572%	1079 112 k	147	121214	246514	92.50%	0.0483%	1087 283 k	31 s	852 M
Velvet	21	473	13761	48489	98.09	0.0323%	5 15 k	111	111871	225438	96.81%	0.0291%	6 67 k	43 s	526 M

Table 5 The assembly results on real single cell sequencing data of *Escherichia coli* (~4.64 Mb)[a)]

	k	Contigs						Scaffolds						Time	Mem
		No.	N50	Max len	Cov (%)	Sub. err	err # (len)	No.	N50	Max len	Cov (%)	Sub. err	err # (len)		
IDBA-UD	40–100	187	82007	224018	95.01	0.0017%	87 347 k	148	98306	224018	95.00	0.0016%	73 346 k	35 min	3.2 G
SOAPdenovo	75	6008	6428	50965	92.42	0.0421%	693 335 k	4419	25244	118128	86.49	0.0448%	588 653 k	30 min	19 G
Velvet	45	1736	7679	68395	92.69	0.0095%	160 266 k	1707	7795	68395	92.59	0.0095%	154 272 k	91 min	11 G
Velvet-SC	55	372	34454	157931	92.74	0.0019%	78 279 k							46 min	8.3 G

a) The read length is 100, insert distance is about 215, and the average depth is about 600×.

7 Experimental results

IDBA package outperforms existing genome assemblers in multiple simulated and real datasets [19,21,22–25]. In this section, we show the experimental results on one simulated dataset and one real dataset for demonstration. Tables 4 and 5 show the performance of IDBA-UD on assembling simulated and real single cell sequencing genomic data. The simulated data was generated from the genome of *Lactobacillus plantarum*. Length-100 reads with 1% sequencing error rate and depth 10× were generated and assembled by Velvet [11], SOAPdenovo [27] and IDBA-UD [22]. A contig is considered as correct if it can be aligned to a region of the genome with 95% similarity. As showed in Table 4, when the sequencing depth is low (10×), IDBA-UD produced longer contigs (more than double N50 and five times of the maximum contig length) and scaffolds (60% increases in N50 and 65% increases in the maximum scaffold length) with the minimum errors. Similar result obtained in the real single cell sequencing data of *Escherichia coli*. The read length was 100 and the insert distance was around 215 nt. The average sequencing depth of the data was 600× but the sequencing depth for different region varied a lot. When the variation of sequencing depths is two order of magnitudes, IDBA-UD can still produce longer contigs (more than double N50 and 43% increases in the maximum contig length) and scaffolds (four times increases in N50 and double the maximum scaffold length) with the second minimum error in contigs and the minimum error in scaffolds.

This work was supported in part by Hong Kong GRF HKU 7111/12E, 719611E, Shenzhen Basic Research Project JCYJ20120618143038947 (SIRI/04/04/2012/05) and Outstanding Researcher Award (102009124).

1 Holley RW, Everett GA, Madison JT, Zamir A. Nucleotide sequences in the yeast alanine transfer ribonucleic acid. J Biol Chem, 1965, 240: 2122–2128
2 Holley RW, Apgar J, Everett GA, Madison JT, Marquisee M, Merrill SH, Penswick JR, Zamir A. Structure of a ribonucleic acid. Science, 1965, 147: 1462–1465
3 Min Jou W, Haegeman G, Ysebaert M, Fiers W. Nucleotide sequence of the gene coding for the bacteriophage MS2 coat protein. Nature, 1972, 237: 82–88
4 Fiers W, Contreras R, Duerinck F, Haegeman G, Iserentant D, Merregaert J, Min Jou W, Molemans F, Raeymaekers A, Van den Berghe A, Volckaert G, Ysebaert M. Complete nucleotide sequence of bacteriophage MS2 RNA: primary and secondary structure of the replicase gene. Nature, 1976, 260: 500–507
5 Sanger F, Air GM, Barrell BG, Brown NL, Coulson AR, Fiddes CA, Hutchison CA, Slocombe PM, Smith M. Nucleotide sequence of bacteriophage phi X174 DNA. Nature, 1977, 265: 687–695
6 Warren RL, Sutton GG, Jones SJ, Holt RA. Assembling millions of short DNA sequences using SSAKE. Bioinformatics, 2007, 23: 500–501
7 Jeck WR, Reinhardt JA, Baltrus DA, Hickenbotham MT, Magrini V, Mardis ER, Dangl JL, Jones CD. Extending assembly of short DNA sequences to handle error. Bioinformatics, 2007, 23: 2942–2944
8 Dohm JC, Lottaz C, Borodina T, Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res, 2007, 17: 1697–1706
9 Hernandez D, François P, Farinelli L, Osterås M, Schrenzel J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res, 2008, 18: 802–809
10 Myers EW. The fragment assembly string graph. Bioinformatics, 2005, 21(Suppl 2): ii79–85
11 Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res, 2008, 18: 821–829
12 Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I. ABySS: a parallel assembler for short read sequence data. Genome Res, 2009, 19: 1117–1123
13 Chaisson MJ, Brinza D, Pevzner PA. De novo fragment assembly with short mate-paired reads: does the read length matter? Genome Res, 2009, 19: 336–346
14 Chaisson MJ, Pevzner PA. Short read fragment assembly of bacterial genomes. Genome Res, 2008, 18: 324–330
15 Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB. ALLPATHS: de novo assembly of whole-genome shotgun microreads. Genome Res, 2008, 18: 810–820
16 Salikhov K, Sacomoto G, Kucherov G. Using cascading Bloom filters to improve the memory usage for de Brujin graphs. In: Darling A, Stoye J, eds. Algorithms in Bioinformatics, Lecture Notes in Computer Science, vol. 8126. Berlin Heidelberg: Springer, 2013. 364–376
17 Burrows M, Wheeler DJ. A block sorting lossless data compression algorithm. Technical Report 124. Palo Alto, CA: Digital Equipment Corporation, 1994
18 Rodland EA. Compact representation of k-mer de bruijn graphs for genome read assembly. BMC Bioinformatics, 2013, 14: 313
19 Peng Y, Leung HSM, Yiu SM, Chin FYL. IDBA—A practical iterative de Bruijn graph de novo assembler. In: Research in Computational Molecular Biology. Proceedings of the 14th Annual International Conference, Lisbon, Portugal, 2010. 426–440
20 Vyahhi N, Pham SK, Pevzner P. From de Bruijn graphs to rectangle graphs for genome assembly. In: Algorithms in Bioinformatics. Proceedings of the 12th International Workshop, Ljubljana, Slovenia, 2012. 249–261
21 Peng Y, Leung HC, Yiu SM, Chin FY. Meta-IDBA: a de novo assembler for metagenomic data. Bioinformatics, 2011, 27: i94-101
22 Peng Y, Leung H, Yiu SM, Chin F. IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics, 2012, 28: 1420–1428
23 Leung H, Yiu SM, Parkinson J, Chin F. IDBA-MT: de novo assembler for metatranscriptomic data generated from next-generation sequencing technology. J Comput Biol, 2013, 20: 540–550
24 Leung HCM, Yiu SM, Chin FYL. IDBA-MTP: a hybrid metatranscriptomic assembler based on protein information. In: Research in Computational Molecular Biology. Proceedings of the 18th Annual International Conference, Pittsburgh, PA, USA, 2014.160–172
25 Peng Y, Leung HCM, Yiu SM, Lv MJ, Zhu XG, Chin FYL. IDBA-tran: a more robust de novo de Bruijn graph assembler for transcriptomes with uneven expression levels. Bioinformatics, 2013, 29: i326–334
26 Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. Genome Biol, 2010, 11: R116
27 Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, Li S, Yang H, Wang J, Wang J. De novo assembly of human genomes with massively parallel short read sequencing. Genome Res, 2010, 20: 265–272