第二代高通量测序组装综述

孙钊

January 9, 2013

Abstract

第二代高通量测序产生了大规模的短序列。大规模输入，指数级的序列重叠，高概率的序列重复使得早期基于“overlap-layout-consensus”的方法不再适用。新的基于de bruijn图的方法在短序列组装中取得了成功。本文以一个纯计算机专业的学生的角度阐述了第二代高通量测序方法，序列组装的挑战以及如何进行序列组装的问题。在文末描述了我自己序列组装领域做出的一些努力。

1 开始之前

首先我想解释一下我写这篇文章的动机。这篇文章的主题是关于基因工程的计算机辅助过程–序列组装。序列组装实际上是生物信息学的范畴。作为一个纯计算机背景的学生，我在2010年的夏天开始在这个领域进行研究。在读了大量的相关领域论文，与国内外的生物科研人员充分交流，尝试了各种各样的生物信息学中的工具之后，我没有取得任何创新性的成果，但是获得了非常宝贵的领域经验，并希望与那些对该领域一无所知又徘徊在领域门口的计算机的学生分享这些经验。本文我讲逐一回答“什么是基因测序”,“什么是序列组装以及它的挑战”, “如何组装序列”, “如何评价组装软件”这些问题，以及在最后列举我个人在这个领域做出的一些努力。本文中，我不会涉及太多算法的细节，我主要的目的是给读者一些关于基因序列组转中“做什么”和“如何做”的直觉印象，并且，我会提供大量的相关文章，材料和链接来帮助你构建一个完整的领域框架。

2 什么是基因测序

在基因工程和生物化学领域，测序意味着确定无支链生物聚合物的一级结构，例如DNA, RNA和蛋白质。本文将除去蛋白质而只关注DNA和RNA的序列分析。DNA测序就是确定一段给定DNA链的核苷酸序列的过程。具体而言，DNA是由4种分别用‘A’, ‘G’, ‘C’, ‘T’来表示的核苷酸组成的序列，也就是包含‘A’，‘G’，‘C’，‘T’的一个字符串。但是，生物上应用最广泛的鸟枪法测序，随机的把原始的DNA链条打断成碎片DNA，然后对这些碎片DNA进行测序。这些DNA碎片测出来的DNA序列称为一个read。为了提高read的覆盖度和质量，大量的由PCR扩增技术通过细菌作为模板的DNA复制链被同时测序Fig. 1显示了鸟枪法测序的过程，同时你会发现，仅碎片DNA的两端被测序了而不是全部。这种只测两端的现象是由生物测序的方法决定的，但是通过加入特别的方法，这两对的read的距离可以被估计出来。这时，这两个read被称作是pair-end 的read，它们之间的距离叫做insert length。

DNA分子是有两条互补链组成的双螺旋结构。根据碱基配对的原则–‘A’配对‘T’，‘G’配对‘C’，可由DNA分子中的一条链推导出互补的那条链。一些用3’和5’表示的特殊序列用来表示两条链的方向。为了简化说明起见，如果一条链定义为3’到5’方向，则其互补链为5’到3’方向。例1显示了一个双链的碎片DNA，其中红色部分对pair-end的read。注意到pair-end的两条read是分别位于两条互补链上的，但他们测序的方向都是从3’到5’。因此，例1中的两条read对应的序列字符串为‘AGCTAA’ and ‘GCCAA’。

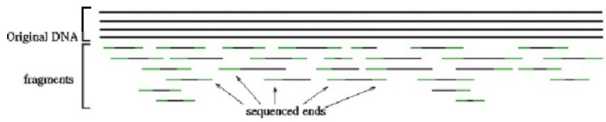

Figure 1:

3'→5'
AGCTAATGCTATCTTGGC
TCGATTACGATAGAACCG
5'→3'
Example 1

测序方法和平台是随着时间的发展而发展的。第一代测序平台的代表是Sanger[14], 它测序出来的read平均长度大约为800bp（一般为500-600bp）。最近，新的测序技术已经出现[10]。已经可以商业应用的第二代测序平台有454测序仪[11]，Illumina测序仪[2]和SOLID测序仪(www.appliedbiosystems.com)。和传统的Sanger测序平台相比，这些技术节省了大量的花费并且输出的吞吐量很高。但是，第二代测序技术产生的read序列要比传统的Sanger序列短很多，一般来说454可以达到400-500bp，Illumina为50bp，而SOLID为35bp。正是因为他们长度太短，和早起的测序项目相比，他们必须提高read的覆盖度输出跟大规模的read序列。
测序芯片把输出的read序列保存在用FASTA或者FASTQ格式的文件中。FASTA格式的文件中，每一条read序列由一行描述行开始，然后紧跟着read序列字符串的行。描述行一般都由符号“>”开始，read序列建议每行不超过80个字符。Fig. 2显示了一个FASTA格式的序列文件。FASTQ文件使用4行来表示一个read序列，第一行一般由'@'符号开始表示一个序列标识符。第二行是原始的read序列字符串。第三行由'+'号开始一行新的描述标识符。第四行用ascii编码质量数值，其中每个字符的ascii数字对应了第二行同样位置的序列的质量。如果想了解更多关于质量打分行的细节，可以参考维基百科上的对FASTQ 格式的介绍。
最后需要一提的是RNA测序。由于RNA是由DNA转录获得，它所有信息都已经存储在细胞的DNA中。通常对RNA的测序方法首先采取逆转录的反应，将RNA逆转录为cDNA，然后对cDNA采用DNA的测序方法即可。
总之第二代测序平台对DNA或RNA进行测序，输出了一个用FASTA或FASTQ文件格式的序列文件。

3 什么是序列组装

测序阶段输出的仅仅是一堆碎片DNA的序列，而我们需要的是原始的完整DNA的核苷酸序列。序列组装就要要对齐和合并这些碎片DNA序列为原始序列。直觉上，需要利用read序列之间的重叠区域来把他们粘成一个更长的序列。下面列出的一些挑战使得序列组装成为一个十分复杂的任务：

- 一个细胞中含有多条染色体，继而含有多条DNA链。这些DNA链同时被

```
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA for prepro cortistatin like peptide, complete
cds.||len=368
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```

Figure 2:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

Figure 3:

测序，不同链上的序列可能会有重叠。

- DNA是双链结构，一条链上的序列可能和它的互补链序列重叠。
- 随机打断序列的过程并不是一个标准的泊松随机过程，可能会有一部分DNA区域没有被read覆盖到。
- DNA序列中可能会有重复出现的序列，这些序列会被错误的当做重叠区域而合并。
- 测序错误是的read序列出现插入，删除，突变的字符。
- 第二代高通量测序输出较短的read序列，是的read之间重叠的概率增大。

Among those above challenges, the repeat problem is the most challenge thing for short read assembly. A simple example is shown in Fig. 4A, where the assembler incorrectly collapses the two copies of repeat A leading to the creation of two contigs instead of one Fig. 4B.

The basic strategies for assembling sequences are categorized by two groups.

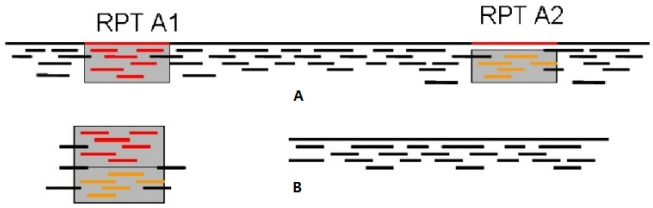

Figure 4:

One is reference based assembly, and the other one is de novo assembly. Reference based methods using a mapping tool to align reads against an existing reference sequence and build a sequence that is similar but not necessarily identical to the reference. The de novo assembly strategy does not use a reference sequence and is much useful for species whose reference sequence are unknown. This article only covers the de novo assembly and will illustrate the details in the subsequent sections.
Ideally, an assembly program should produce one contig for every chromosome of the genome being sequenced. However, because of challenge3 and challenge4, the output of the assembler is interrupted into a set of contigs. In summary, Inputting a FASTA/FASTQ read file, the assembler outputs a set of contigs.

4 How to assembly sequence

Overlap-consensus-layout [12] is the traditional approach for first generation sequence. It denotes each read as a separate node, where two reads presenting a clean overlap are connected by a bidirected edge. Although overlap-consensus-layout approach is both intuitive and robust, especially in the case of long reads, it is very costly for pair-wise overlap computing when assembling high quantity short reads produced by second generation sequencing platform like SOLID and Illumina. Only one microread assembler, EDENA [6] was developed using this approach. Later, Idury and Waterman [7] introduced the use of a sequence graph to represent an assembly, and this idea has been developed into the most popular framework for short read assembly–De Bruijn graph. Sequence De Bruijn graph is constructed from a set of kmers(k is a parameter) as nodes, and two kmers are connected if one kmer's last k-1 nucleotides is identical to the first k-1 nucleotides. kmers are produced by moving a fixed length(k) sliding window on the original read. Examples of kmers and de bruijn graph are show in Fig. 5 and Fig. 6. The sequence de bruijn graph has following properties:

- Read with length of n generates n-k+1 kmers.
- Read is mapping to a unique path in the de bruijn graph.
- One node has at most 4 successors and 4 predecessors by adding 'A', 'G', 'C', 'T' at the first or last of the k-1 nucleotides.

According the property2 and property3, the node scale of the de bruijn graph is bounded by O(n-k+1) and the edge scale of it is bounded by O(8(n-k+1)). By the way, the length of human is about 3G base pairs whose de bruijn graph can be stored in a single server machine or cluster configured with 128GB. genome Property3 release the heavy cost of searching for all pair-wise overlap, thus making de bruijn graph approach feasible. Softwares include Velvet [17], ABYSS [16], ALLPATHS [3], SOAPDenovo [9] follows the de bruijn graph method. ABYSS and ALLPATHS claimed that they successfully assembled human's genome. A brief comparison of these softwares is show in Table 1. Software links can be found in Table 2.

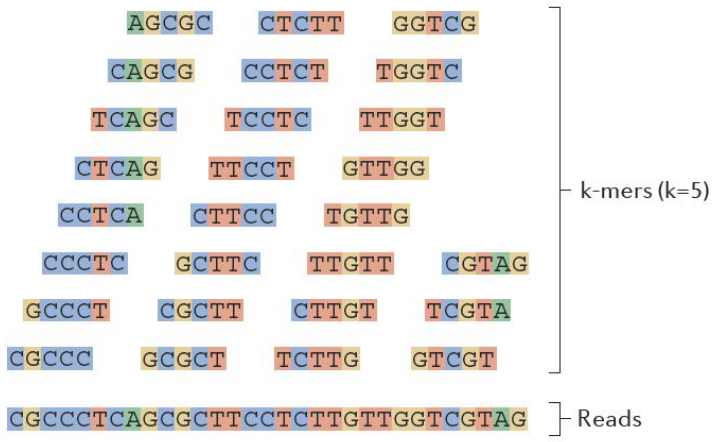

Figure 5:

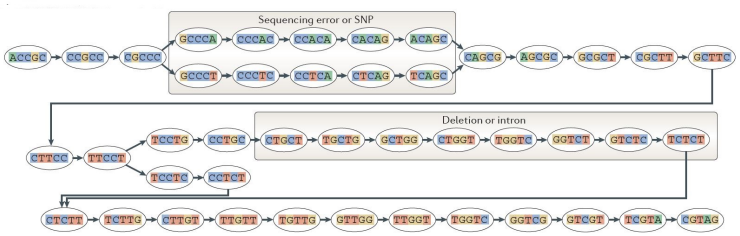

Figure 6:

I've read code and run programs of Velvet and ABYSS and provide some details about them. Velvet consists of two modules named velveth and velvetg where 'h' means hash and 'g' means graph. Velvet designed a novel graph construction algorithm and claim that it is efficient and memory saved approach on a single machine. velveth sequentially reads the FASTA/FASTQ read file and for each k-mer observed in the set of reads, a hash table records the ID of the first read encountered containing that k-mer and the position of its occurrence within that read. Each k-mer is recorded simultaneously to its reverse complement sequence. To ensure that each k-mer cannot be its own reverse complement, k must be odd. This first scan allows each read to be re-written as a set of new k-mers combined to overlaps with previously hashed reads. This new representation of the reads sequence is called a roadmap which is written into a file named ROADMAP. velvetg reads the ROADMAP file and creats a second database with the opposite information. It records, for each read, which of its original k-mers are overlapped by subsequent reads. The ordered set of original k-mers

of that read is cut each time an overlap with another read begins or ends. For each uninterrupted sequence of original k-mers, a node is created. Finally, reads are traced through the graph using the roadmaps. Knowing the correspondence between original k-mers and the newly created nodes, it is possible to proceed from one node to the next, creating a new directed arc or incrementing the multiplicity of an existing one as appropriate at each step. The final output of velvet is produced by velvetg and saved as config.fa. ABYSS constructs de bruijn graph directly following the definition. The MPI version uses a hash function to determine the host machine for a node. The structure of ABYSS program is organized by a make file driver named "abyss-pe". abyss-pe automatically finds the dependency of abyss modules and perform a minimum number of running modules. The final output is saved as *-contigs.fa, *-scanffolds.fa, *-unitigs.fa and a statistic file named *-stats.

After building the de bruijn graph, a graph correction process is applied. The

Table 1:

Software	Parallelism	Large Scale Assembly	Memory Cost
Velvet	OpenMP	No	High
ABYSS	OpenMP+MPI	Yes	Low
ALLPATH	?	Yes	?
SOAPDenovo	?	Yes	Low

Table 2:

Software	Software Available
Velvet	http://www.ebi.ac.uk/ zerbino/velvet/
ABYSS	http://www.bcgsc.ca/platform/bioinfo/software/abyss/
ALLPATH	http://www.broadinstitute.org/science/programs/genome-biology/crd
SOAPDenovo	http://soap.genomics.org.cn/soapdenovo.html

idea of the graph correction process is to simplify the original graph into a long path by removing branches. There are two types of branches–tips and bubbles. A tip is a chain of nodes which is disconnected on one end and a bubble is a circle with two common end points. Whenever a node A has only one outgoing arc that points to another node B that has only one incoming arc, the two nodes can be merged into a long node. Tips and bubbles are shown in Fig .7. Note that repeat region also creates circles in the graph, and we can not determine how many times should the repeat sequence occur. In the last step, if pair-end reads are provided, we can estimate the distance of seed nodes and find a feasible path between them.

The last thing need to mention is that RNA assembly also named transcript assembly. RNA assembly is more complex than DNA assembly, observing the

reasons:

- some transcripts have low coverage, whereas others are highly expressed.
- reads with sequencing errors derived from a highly expressed transcript may be more abundant than correct reads from a transcript that is not highly expressed.
- transcripts encoded by adjacent loci can overlap and thus can be erroneously fused to form a chimeric transcript.

Directly using DNA assemblers to assemble RNA sequences only outputs rather bad results. Velvet and ABYSS have extend themselves into Velvet_Oases [15] and Trans-ABYSS [13] separately to address the additional challenge presented by RNA assembly. Recently, Trinity [4] was specially designed for transcript assembly. Zhao et. al [18] made a early comparison of transcript assembly and give a full view of performance, result, resource usage evaluation. It shows Trinity has a relative better assembling ability than Velvet_Oases and Trans-ABYSS but much slower and memory cost while assembling large scale sequences. A group of HPC developers [5] made efforts to accelerating trinity and reducing its memory usage which make trinity a first choice when assembling transcript sequences.

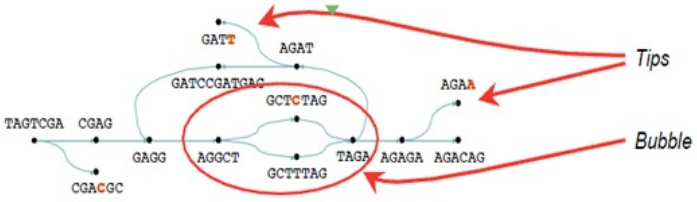

Figure 7:

5 How to Evaluate Assembler

To evaluate present assemblers or assemblers designed by ourselves, the first thing to do is to prepare input data. Usually, two kinds of data are used to evaluate assemblers, synthetic data and real data. dwgsim [1] is whole genome simulator based on a reference genome. Reference genome can be found on ftp://ftp.ncbi.nlm.nih.gov/refseq/, ftp://ftp.ncbi.nlm.nih.gov/genomes/ or NCBI's nucleotide database (http://www.ncbi.nlm.nih.gov/nucleotide). Details about refseq is explained on http://www.ncbi.nlm.nih.gov/books/NBK50679/. NCBI's SRA database [2] provides a large collection of read read sequences. Users search

[1] http://sourceforge.net/apps/mediawiki/dnaa/index.php?title=Whole_Genome_Simulation
[2] http://www.ncbi.nlm.nih.gov/sra

this database using an access code begin with SRP(studies), SRS(samples), SRX(experients) or SRR(runs). Multiple SRRs of same samples can be used together as input. By the way, files downloaded from SRA database are in sra format and users should dump FASTA/FASTQ format using fastq-dump from NCBI SRA Toolkit. If pair-end data are expected to be dumped, pass '–split-3' or '–split-files' to fastq-dump. Metrics for evaluate assembler can be N50 length, running time, memory cost, 95% map ratio and so on. Given a set of contigs(output of assemblers) of varying lengths, the N50 length is defined as the length N for which half of all bases in the sequences are in a sequence of length L $<$ N. Obviously, the larger the N50 length, the better the assembled contigs. On linux like systems, users can use '/usr/bin/time -v' command to record the running time and peak memory usage of a user process. The 95% mapping ratio is defined as the number of contigs with at least 95% identity mapping to the reference dividing by the total number of contigs. BLAST [1] and BLAT [8] are two popular mapping tools for aligning contigs to references. I often choose BLAT for reasons of efficiency. BLAT outputs a very big table(PSL format) to store the mapping info for each query sequence. Users who are strange to PSL format and the meaning of each column should refer to PSL format. Each query may have multiple candidate alignment positions, and BLAT program doesn't provide a percent identity value column or score column like the web BLAT [3]. Users can replicate the two columns referring to http://www.genoscope.cns.fr/blat-server/cgi-bin/vitis/webBlat and use the best hit to calculate 95% mapping ratio.

6 My Own Contribution

My own contribution to the sequence assembly is shown in Table 3.

Table 3:

Contribution	Links
Velvet and Oases for windows	https://github.com/zixiaojindao/velvet.git
ABYSS for windows	https://github.com/zixiaojindao/ABYSS.git
MemoryUsageMonitor for windows	https://github.com/zixiaojindao/MemoryUsageMonitor.git
blat-statistics for windows	https://github.com/zixiaojindao/blat-statistics.git
Sequence Assembly Information	https://github.com/zixiaojindao/RNA-sequence-info.git

7 Summary

In a programmer's perspective, sequence assembly is piecing string fragments into a set of long strings using overlap among them. However, things are much

[3] http://www.genoscope.cns.fr/blat-server/cgi-bin/vitis/webBlat

complicated in practice due to biological challenges. Most of assembly softwares follow the de bruijn graph approach and reach a satisfying point. However, the running time and memory cost for large scale sequence is still not very good as well as assembled contigs. Hope new computer algorithms and biological methods could help address this problem much better in the future.

Reference

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, et al. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] D.R. Bentley. Whole-genome re-sequencing. *Current opinion in genetics & development*, 16(6):545–552, 2006.

[3] J. Butler, I. MacCallum, M. Kleber, I.A. Shlyakhter, M.K. Belmonte, E.S. Lander, C. Nusbaum, and D.B. Jaffe. Allpaths: De novo assembly of whole-genome shotgun microreads. *Genome research*, 18(5):810–820, 2008.

[4] M.G. Grabherr, B.J. Haas, M. Yassour, J.Z. Levin, D.A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, et al. Full-length transcriptome assembly from rna-seq data without a reference genome. *Nature biotechnology*, 29(7):644–652, 2011.

[5] R. Henschel, P.M. Nista, M. Lieber, B.J. Haas, L.S. Wu, and R.D. LeDuc. Trinity rna-seq assembler performance optimization. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*, page 45. ACM, 2012.

[6] D. Hernandez, P. François, L. Farinelli, M. Østerås, and J. Schrenzel. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome research*, 18(5):802–809, 2008.

[7] R.M. Idury and M.S. Waterman. A new algorithm for dna sequence assembly. *Journal of Computational Biology*, 2(2):291–306, 1995.

[8] W.J. Kent. Blat鈥攖he blast-like alignment tool. *Genome research*, 12(4):656–664, 2002.

[9] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, 20(2):265–272, 2010.

[10] E.R. Mardis et al. The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133, 2008.

[11] M. Margulies, M. Egholm, W.E. Altman, S. Attiya, J.S. Bader, L.A. Bemben, J. Berka, M.S. Braverman, Y.J. Chen, Z. Chen, et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, 2005.

[12] E.W. Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290, 1995.

[13] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S.D. Jackman, K. Mungall, S. Lee, H.M. Okada, J.Q. Qian, et al. De novo assembly and analysis of rna-seq data. *Nature methods*, 7(11):909–912, 2010.

[14] F. Sanger, GM Air, BG Barrell, NL Brown, AR Coulson, JC Fiddes, P-M Slocombe, and M. Smith. Nucleotide sequence of bacteriophage (d x174 dna. 1977.

[15] M.H. Schulz, D.R. Zerbino, M. Vingron, and E. Birney. Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, 2012.

[16] J.T. Simpson, K. Wong, S.D. Jackman, J.E. Schein, S.J.M. Jones, and İ. Birol. Abyss: a parallel assembler for short read sequence data. *Genome research*, 19(6):1117–1123, 2009.

[17] D.R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[18] Q.Y. Zhao, Y. Wang, Y.M. Kong, D. Luo, X. Li, and P. Hao. Optimizing de novo transcriptome assembly from short-read rna-seq data: a comparative study. *BMC bioinformatics*, 12(Suppl 14):S2, 2011.