*Hans Journal of Computational Biology* 计算生物学, 2013, 3, 7-14
http://dx.doi.org/10.12677/hjcb.2013.32002 Published Online June 2013 (http://www.hanspub.org/journal/hjcb.html)

# Genome Assembly Algorithms: A Survey*

**Shuaibin Lian, Xianhua Dai**

School of Information Science and Technology, Sun Yat-sen University, Guangzhou
Email: shuai_lian@qq.com

Received: Dec. 10th, 2012; revised: Dec. 25th, 2012; accepted; Jan. 24th, 2013



**Abstract:** During the last twenty years, genome sequencing technology has made a great development. With the appearance of the Next Generation Sequencing Technology characterized by higher throughput, shorter read and even lower cost, the time and cost of sequencing the whole genomes of a species are sharply decreased. The genome assembly algorithms and software are developed in succession, currently there are almost twenty ones. Due to the complexity of genome assembly itself, there is no survey aiming at the special steps, operational environment, applications of the different genome algorithms. In this perspective, we survey these typical twelve genome assembly algorithms briefly and then analyze the special steps, principles, operational environments and applications. This survey can provide a guidance on how to design or develop a genome assembly algorithm or software, and which genome assembly algorithm of software is more suitable according to different genome sequencing data.

**Keywords:** Genome Assembly; De Bruijn Graph; Next Generation Sequencing (NGS)

# 基因组装算法：调研*

**连帅彬，戴宪华**

中山大学信息科学与技术学院，广州
Email: shuai_lian@qq.com

收稿日期：2012 年 12 月 10 日；修回日期：2012 年 12 月 25 日；录用日期：2013 年 1 月 24 日

**摘　要：**基因测序技术在过去的二十几年里取得了突飞猛进的发展，随着以高通量，短读取，低成本为特点的新一代基因测序技术的问世，测序一个物种全基因的时间和成本大大降低。基于下一代测序技术的全基因组装算法和软件相继开发出来，目前比较成熟的基因组装算法大约有二十种左右。由于基因组装问题本身的复杂性，目前还没有针对不同组装算法的具体设计步骤，操作环境，应用范围等方面的调研。基于此本文简要调研了现有的十二种具有代表性的基因组装算法，系统的分析了每种算法的设计步骤，算法原理，操作环境以及应用。这篇调研对于如何设计基因组装算法，对于不同的基因数据如何选择更加合适的基因组装算法和软件提供了一定的指导。

**关键词：**基因组装；De Bruijn 图；下一代测序技术(NGS)

## 1. 引言

基因测序技术已经取得了巨大的发展，目前出现以 Sanger 测序为主第一代测序技术和以短读取高通量为特点的下一代测序技术(NGS)并存的局面。主流的商业 NGS 平台有 Roche/454，Illumina/Solexa，Life/APG，Pacific Biosciences and Helicos BioSciences[1-5]。Sanger 测序技术的[6]主要特点为：低通量，

*资助信息：国家自然科学基金项目(G61174163)。

长读取(1000~2000 bp)，高正确率(99.99%)，高费用(0.5美元/1000)。下一代测序技术(NGS)的特点：高通量(1~10 Gb/lane)，短读取(30~150 bp)，低费用，高误差率(1%~5%)。高通量，低费用的特点使得测序一个物种的全基因组的时间和费用大大缩减。这些特点使得基于 NGS 的全基因组装算法和软件相继开发出来。全基因组装就是从大量的短读取中进行染色体重构[7,8]。但是不同的 NGS 平台产生的数据具有不同的读取长度，误差率，覆盖偏差，覆盖深度等[9]。这些特点都为基因组装带来了挑战。这些挑战主要包括以下几个方面：

1) 复杂的重复结构[10-12]。在全基因组中，片段的短重复是大量存在的，这些重复有完全重复和高相似重复。这些重复的存在导致基因组装在重复面前束手无策。

2) 短读取。NGS 的重要特点就是短读取，读取越短传递的信息就会越少，在基因组装过程中可以利用的重叠就越短，而且更短的读取需要更高的覆盖，而更高的覆盖产生更大的数据量，大大增加了计算上的复杂性。

3) 测序误差[13,14]。NGS 的误差普遍比较大(1%~5%)，误差的存在使得高相似重复与单核苷酸多态性(SNP)更加难以区分。

4) 测序覆盖偏差[15]。不均匀的测序过程，导致了测序覆盖偏差的存在，这些偏差增加了估计重复的数目的难度。

5) 海量数据[16]。全基因组的海量数据对计算平台有了更加苛刻的要求，特别的大物种的全基因组装需要专门的高性能计算平台的支持。

全基因组装算法的设计和实施是非常复杂的过程，基于下一代测序平台的全基因组装算法和软件大致分为：基于 Overlap/Layout/Consensus(OLC)，基于 Greedy Graph，基于 de Bruijn Graph。其中基于 OLC 的算法主要有：CAP3[17]，EDENA[18]，Newbler[19]，Arachne[20]；基于贪心图的算法有：SSAKE[21]，SHARCGS[22]，VCAKE[23]，QSRA[24]，TIGR Assembler[25]；基于 de Bruijn 图的算法有：EULER-SR[26]，ALLPATHS[27]，Velvet[28]，ABySS[29]，SOAPdenovo[30]。尽管大量的基因组装算法相继提出，但是最近的研究比较发现[31]，不同算法之间的差异性很大，基因组装算法的性能与测序数据以及技术平台有密切关系，同时基因组装还面临着多重因素的挑战。本文系统分析了三类不同方法的 11 种典型算法的设计步骤，原理，开发环境，适用范围等。对于如何开发和设计基因组装算法，如何选择合适的基因组装软件进行基因组装具有一定的指导意义。

## 2. 基于 OLC 的组装算法

图是计算机科学中常用的一种数据结构，被广泛应用于很多学科领域。基于 OLC 的基因组装算法采用重叠图的原理进行短读取基因组装。重叠图表示测序读取以及读取之间的重叠关系，这些重叠关系必须通过两两排列事先计算出来。理论上，采用读取表示重叠图的结点，读取间的重叠关系表示重叠图的边；实际上，利用重叠图进行真实的全基因测序组装会出现以下几个问题：

1) 死结点(dead-end divergences)。这些短的死结点往往是由读取末端的测序误差所引起的。

2) 泡结点(bubbles)。这些泡结点往往是由于读取中间的测序误差或者单核苷酸多态性(SNP)引起。

3) 多通路。真实基因数据中存在大量的短重复(完全重复)，这些重复会导致多通路的产生。

基于 OLC 的组装算法主要使用于 Sanger 测序数据，文献[32-34]对此做了详细的综述。基于 OLC 的典型算法主要有：CAP3，EDENA，Newbler，Arachne。

### 2.1. CAP3

CAP3 是经典的基因 OLC 的组装算法，并且是第三代 CAP 组装算法，能够快速的找出读取之间的重叠关系，并且能够有效的计算和评价重叠关系，同时利用双端约束进行误差校正。

方法与步骤：

1) 误差校正与计算重叠：首先 5’和 3’端误差较大的区域被检测出来并滤除掉，然后计算出读取之间的重叠关系，并去掉假重叠。

2) 组建 contig：按照重叠关系的分值大小，将读取连接成 contig，然后使用正反互补约束进行误差校正。

3) 组建一致性序列：首先建立短读取的多序列排序，根据每个碱基的质量值对每个 contig 计算出一个

一致性序列。

CPA3 对于误差有较好的容忍度，能够分别从 5’和 3’端滤除低质量值的读取，同时使用质量值组建多序列排序[35]，并且可以使用双端约束组建 contigs。

应用范围：适用于中小基因组，可以在 PC 机上完成小基因组装。

计算平台：Intel Pentium CPU 2.4-GHz computer supplied with 4.0 Gb of RAM.

开发语言：C

操作系统：Linux

### 2.2. Edena

EDENA 是一款专门为 Illumina sequencing platform 设计的短读取全基因组装软件，它也是基于经典的重叠图理论。EDENA 算法的组装效果利用 Sanger 长读取的基因数据 Staphylococcus aureus strain MW2 and Helicobacter acinonychis strain 进行验证，同时采用 Lander-Waterman statistics 计算[36]基因的覆盖深度，增加了正确 contig 组建的概率[37]。

方法与步骤：

1) 减少读取冗余：首先去除冗余读取，每个读取只留下一个拷贝，每个读取出现的频率被索引下来用于计算覆盖深度。

2) 重叠计算阶段：通过索引非冗余读取集计算重叠，最小重叠数目不能太大也不能太小，这是一个经验值，太大或者太小会影响算法的性能。

3) 去除过度边：在重叠图中的大多数边均为过度边，这些过度边对于表示一致性序列并不重要，通过去除过度边，减少了图计算的复杂性。

4) 净化图：在每一个分支节点处通过局部图检测进行净化操作。首先去除短死节点通路(dead end path)，然后确定短泡节点通路(bubbles)。

5) 使用严格和非严格模式组装：检查每一个分支节点，保留最大重叠值的边，当对应最大重叠长度的边唯一时，进行净化操作。

应用范围：适用于 Illumina Genome Analyzer technology 的中小基因组，效率相对较高可以在 PC 上完成小基因组装。

计算平台：Intel Pentium D CPU 2.8-GHz computer supplied with 4.0 Gb of RAM.

## 3. 基于贪心图的组装算法

在文献[38,39]中综述了第一个基于贪心策略的基因组装算法，它的基本原理：给定一个读取或者种子，只要它满足条件就对这个种子不断的延长直到所有的数据用完或者超过设定阈值；每次循环延长都采用最优选择标准。基于贪心策略的基因组装算法往往可以到达局部最优，但是很难实现全局最优，因此贪心组装算法需要一种能避免错误延长的机制。

### 3.1. SSAKE

SSAKE 是典型的贪心图组装算法，通过贪心选择最大重叠进行组装。

方法和步骤：

1) 将测序数据以 fasta 格式读入内存，并将读取集进行唯一化处理，使用 hash 表存储唯一化处理后的读取以及它们在读取集中出现的频率。

2) 使用前缀树组织每个序列和它们的互补链通过检测 5’端的头 11 个碱基。

3) 每一个没有使用的读取 u 轮流用于作为组装的种子。

4) 通过最佳匹配 r 的无匹配的 3’端延长 u，然后在 hash 表和前缀树中去除 r。

5) 重复上述步骤一直到每一个 u 都被延长。

6) 输出 2 个文件，一个是组装和延长的所有 contigs。另一个是剩余没有组装的序列读取。

使用范围：中小基因组组装。

计算平台：a single 4 核 1.4 GHz AMD Opteron™_ CPU with 32 GB RAM.

开发语言：Perl

操作系统：Linux

### 3.2. VCAKE

VCAKE 和 SAKE 很相似，也是贪心算法的一种，但是也有不同的地方。

方法与步骤：

1) 2 个序列文件单独读入内存，并且从读取池中建立 hash 表。

2) 从序列的 3’端找出满足最小重叠的所有精确匹配的 k-mers。

3) k-mer 的头 11 个碱基被用于搜索最佳匹配并且

 

和剩下的 k-mer 进行检查。

4) 计算阵列匹配序列，每一个序列都对第一个 overhanging 碱基提供一个“vote”，超过阈值 c 的碱基被增加到 contig 中。

5) 一次延长一个碱基一直到没有最佳匹配被检测。互补链用相同的方法进行延长。

6) 以 muti-fasta 格式输出 contigs，然后选择一个没有使用的种子重复组装。

使用的数据有分为两部分，仿真数据和真实数据

1) 仿真数据：从全基因组序列 SARS-TOR2 and Pseudomonas syringae pv. tomato str. DC3000 中以不同的覆盖水平随机产生 30 mer 的仿真读取。

2) 真实数据：Solexa 测序公司 3 lanes 的 DC3,000 数据，总共 202,884,352 个碱基，6,340,136 条读取，达到覆盖深度为 31.7。

使用范围：中小基因组组装。

计算平台：a single 4 核 1.4 GHz AMD Opteron™_ CPU with 32 GB RAM.

开发语言：Perl

操作系统：Linux

### 3.3. SHARCGS

SHARCGS 算法也是贪心算法的一种，它的设计是基于理想假设：没有测序误差，读长固定。在理想假设条件下，设计如何处理数据丢失，如何从数据池中检测包含误差的测序读取数据。算法主要有滤除，组装，融合等三个步骤。

1) 读取确认与滤除：首先获得等长的读取以及每一个碱基的质量值。令 $P_{\text{correct}}$ 表示读取没有测序误差，它可以通过所有正确碱基的成绩估计得到 $P_{\text{correct}} = \left(1 - P_{\text{error}}\right)^r$，$r$ 表示读取的长度。

1) 延长 contig：通过前缀树查找潜在的延伸，只要有一个读取的前缀能精确的匹配到 contig 的末端，就在末端延长 contigs。

2) 融合 contigs：分别使用弱，中，强滤波参数对 contig 进行融合。

3) 参数微调：SHARCGS 当中有两个参数：一个是滤波错误读取，一个是最小重叠数目。对于读取滤波确认阶段，n/2 个读取通过滤波为强滤波条件，n 个读取通过滤波为弱滤波条件，其中 n 为序列长度。最小重叠数目依赖于输入数据间隙的大小，间隙的概率为：$P_{gap} = \left(1 - P_{\text{miss}}\right) P_{\text{miss}}^{gap}$，$P_{\text{miss}} = n_{avail} / n$

数据：仿真数据和真实数据

1) 仿真数据：从 60 个平均长度为 110kb 的 BAC 昆虫序列中产生 30-mer 理想读取。

2) 真实数据：*H. acinonychis* 基因使用 Illumina 1 G sequencing instrument 产生 12.3 million36-mer 的读取，总覆盖深度达到 272。

使用范围：中等基因组组装

计算平台：不同的数据集使用不同的计算平台，最大平台为：4 GB of RAM on an AMD Opteron 2.8-GHz 64-bit Linux machine

开发语言：Perl

操作系统：Linux

### 3.4. QSRA

SHARCGS 算法也是贪心算法的一种，与 VCAKE 非常相似，但是通过改进 VCAKE 使得速度大幅提高，而且能产生更长的 contigs N50/N80。并且对于复杂基因的组装算法提出了更近的目标。算法主要有滤除，组装，融合等三个步骤。方法与步骤：

1) 建立 hash 表和前缀树：hash 表中存放读取序列和对应序列出现的频率值。

2) 找到和种子满足最小匹配碱基数目 u 的所有 n 个 k-mer 读取，u 是用户设定的一个参数。

3) 每一个匹配的读取以及它出现的频率被存放在索引链表中。

4) 计算匹配的总数目。

5) 使用最大匹配延长 contig。如果没有匹配满足条件，使用用户设定的最小质量值延长 contig。

数据：Pinus pinaster (pina) and Pinus gerardiana (gera03)基因使用 Illumina 1G sequencing instrument 测序。Pina 数据覆盖深度为 479，gera03 覆盖深度为 376。

计算平台：3.0 GHz Xeon processor with 32 GB memory

开发语言：C++

操作系统：UNIX

### 3.5. TIGR Assembler

TIGR 组装算法采用基于核酸的快速比较方法，提高算法的执行速度，而且能够检测出潜在的重区域，有较大的改进。

方法与步骤：

1) 计算重叠：在整个数据集中进行片段成对比较计算出所有可能的潜在重叠。

2) 使用潜在重叠数目的分布情况，标记每一个读取是否为重复片段。

3) 选择一非重复片段作为组装的种子，如果没有非重复片段剩余的情况下，可以选择重复片段作为组装种子。

4) 使用潜在重叠列表融合种子和非重复读取片段。

5) 当没有非重复片段的潜在重叠时，增加匹配的标准后，和重复片段进行融合。

6) 如果由于和重复片段的融合，一个非重复片段增加到潜在重叠列表中，返回步骤 4。

7) 当在潜在重叠列表中没有片段剩余时，输出现在的组装结果并返回步骤 3。

使用范围：中，大基因组组装

操作系统：UNIX

开发语言：C

计算平台：Solaris 4.2 on Sun work Stations

## 4. 基于 de Bruijn 图的组装算法

De Brujn 图并不是为基因测序而开发的数据结构，但是却被广泛应用于短序列基因组装中。De Brujn 图中结点表示定长子串，边表示重叠的前缀后缀关系。基于 de Brujn 图的基因组装算法主要使用与 Solexa 和 SOLiD 测序平台。基于图论的组装算法中，大多数都是要找到能够遍历所有节点的一条最短路径或者最长路径，但是目前还没有办法证明最短路径和最长路径是合理的[40]。而且在通路的过程中，可以利用哈密尔顿通路也可以应用欧拉通路，哈密尔顿通路是一个 NP 问题[41]，目前还没有好的解决办法，而欧拉通路问题有简单的计算方法[42]。

### 4.1. Euler

Euler[43,44]组装算法最初是为 Sanger reads 设计的，Euler 使用谱排列方法[42]检测碱基错误，滤除低质量的读取数据进行读取质量校正，利用校正后的读取构建 K-mer 图。

方法与步骤：

1) 误差校正：使用谱排列算法滤除质量值比较低的读取。谱排列：给定一个字符串 S 和一个普 T，找出从 S 变化到 T 所经历的最小变异数目。

2) 欧拉超通路：给定一个欧拉图以及这个图里面的一系列通路，欧拉图就是能够包含所有这些通路为子通路的图。首先等价转化图 G 以及通路系统 P 到一个新的图和 G1 和新通路系统 P1。这种等价变化的条件就是存在一个一对一的欧拉超通路。因此欧拉超通路的问题就可以通过一系列的等价转化将图转化为每一个通路都为单边的通路系统。

应用范围：中小基因组

开发语言：C++

操作系统：Linux

### 4.2. Velvet

Velvet 是一种基于 de Bruijn 的基因组装算法，能有效的进行 de Bruijn 图化简消除误差，解决重复。在仿真数据和真实数据上对 Velvet 进行验证，最大 N50 能达到 50 kb。

方法与步骤：

1) 组建 de Bruijn 图。全部读取集放进 hash 表中构建索引，计算出 k-mer 重叠，利用 k-mer 关系构建 de Bruijn 图。

2) 图化简。在不丢失信息的情况下，化简 de Bruijn 图，通过简化 blocks，chains。当结点 A 有一个输出边指向 B，而结点 B 中有一个输入边，那么两个结点 AB 融合为一个结点。

3) 误差去除。使用基因序列期望覆盖和随机误差之间的差异性进行误差去除。因此去除地覆盖的结点可以有效的去除误差。

4) 去除孤立点。如果一个通路长度小于 2 K，就将这个通路当做孤立点去掉。

5) 去除泡结点。使用 Dijkstra-like breadth-first 搜索检测重复路径，然后融合重复路径。

6) 去除错误连接。

数据：仿真数据和真实数据

仿真数据：*Escherichia coli*, *Saccharomyces cerevisiae*, *Caenorhabditis elegans*, and *Homo sapiens* 覆盖深度为 50。

真实数据：一个人类 BAC 基因 bCX98J21 总长 173,428 bp，有 Solexa 测序平台产生平均覆盖深度为

970，读长 35 bp，k-mer = 31 bp

应用范围：全基因组(小，中，大基因组)组装

计算平台：physical memory are at least 12GB

开发语言：Perl

操作系统：64 bit Linux environment

### 4.3. ABySS

ABySS 是目前唯一一个可以并行组装的基因组装算法，最大的特点就是采用 de Bruijn 图的分布式表示，从而允许组装算法在多台计算机网络中进行并行计算。主要有两个步骤：首先从序列读取中产生所有可能的 k-mers，并进行误差校正组建 contig。其次使用配端信息解决 contig 间的模糊重叠问题延长 contig。

De Bruijn 图的分布式表示：首先一个给定 k-mer 的位置必须确定而且可以从序列 k-mer 中可以计算。其次 k-mers 之间的邻近信息必须和 k-mer 的位置独立存储。

方法与步骤：

1) 建图：数据首先调入分布式 de Bruijn 图中，所有 k-mer 以及它们的邻接关系从序列读取中被组建和计算，并且将 k-mer 放置在分布式 de Bruijn 图中。

2) 误差校正：首先检查出包含死节点(dead end)的分支，然后回溯到模糊边界，如果这个分支小于阈值，将此分支从图中去除。

3) 顶点融合：在图当中去除模糊的边，利用确定的边对顶点进行融合，从而创建初始 contig。

4) Contig 融合：利用配端信息确定可以被连接的 contig。

数据：Illumina 测序平台产生非洲男性 35 亿个 paired-end reads。

使用范围：全基因组(小，中，大基因组)组装

开发语言：C++

操作系统：64bit Linux

### 4.4. SOAPdenovo

Soapdenovo 是一个基于 de Bruijn 图的短读取基因组装算法，能够精确的组装大基因组，比如人类基因组。

方法与步骤：

1) 误差校正：k-mer 通常由基因大小，读取长度，计算机内存等多因素共同确定。误差校正使用 k-mer 频率信息，频率(<3)的 k-mer 被去除。

2) De Bruijn 图的构建：对于 de Bruijn 图，每个节点都是一个 k-mer。重叠 k-1 个碱基的两个节点被连接成为一条边。设定 k-mer = 25。

3) 尖端去除：一个碱基对上的测序误差会导致图中出现泡(bubble)，死节点(dead end)，这些节点都要被去除。

4) 解决微小重复(tiny repeats)：如果每一个有 N 条入边都有通路支持 N 条出边并且没有冲突，那么去除重复节点并且分裂成 N 条平行通路。

5) 融合泡节点(merging bubbles)：如果泡节点中的两条平行路径非常相似，即融合为一条单通路。

6) Contig 连接图：读取之间的配端关系转换为 contig 之间的连接。使用两个 contigs 之间的配端读取数目去加权连接并且使用配端插入大小估计 contig 之间的间隙大小。

7) 支架阶段(Scaffolding)：首先子图线性化——一组 contigs 之间的过度连接被去除，contig 被融合为一个节点；然后去除重复——重复 contigs 被去除掉。

使用范围：全基因组(小，中，大基因组)组装

开发语言：C

操作系统：Linux

计算平台：eight Quad-core AMD 2.3 GHz CPUs with 512 Gb of memory

### 4.5. Arapan-S

它通过简化 de Bruijn 图并且将覆盖信息转换为 k-mer 频率进行组装的一种方法。

方法与步骤：

1) 输入数据构建图：全部 k-mer 数据集被记录在 hash 表中。所有 k-mer 构成 de Bruijn 图的节点，任意两个 k-mer 之间的 k-1 个碱基的重叠构成图的边。

2) 图化简：一个通路就是一串节点链。对于两个节点 X，Y，如果只有一个输出的 X 被连接到只有一个输入的 Y，那么将 X，Y 融合为一个节点。所有连续的节点都被融合为一个节点。

3) 解决泡节点(resolving bubbles)：一个泡节点就是一个由多个单通路组成的子图，这些单通路共有一个起始节点和一个终止节点。所有的泡节点通过净化程序被松弛。

1) 去除尖端(tips)：一个尖端是只有一端连接的节点。去除之后所有剩余节点的度(degrees ≥ 2).

2) 检测图的连通性：有两种情况需要检测图的连通性，第一种情况——太小或者太大的 k-mer 和它的补链；第二种情况——当初始 k-mer 太长时造成的稀疏图。

使用范围：全基因组

计算平台：对大基因组，至少 150 GRAM

开发语言：C/C++

操作系统：64-bit/32-bit Linux

## 5. 总结与讨论

本文系统的综述了目前主流的十二种基因组装算法。这些典型的基因组装算法主要分为三大类：基于 OLC 的基因组装算法，基于贪心策略的组装算法和基于 de Bruijn 图的基因组装算法。对于每一类方法，分别挑选出 3~5 种有代表性的算法进行详细的介绍，主要集中在算法的设计过程，算法的理论基础，应用范围，开发语言，操作环境，计算平台的要求等。这对如何设计基因组装算法和软件，如何选取适合自己的基因组装算法进行组装以及需要什么样的计算平台都有一定的指导意义。

但是基因组装还有很多问题需要解决，组装算法还面临着多重挑战。最近的比较研究发现，不同的组装算法之间差异性很大，即使使用相同的数据和计算平台所得到的基因组装结果也存在较大差异。

在 GAGE[31]和 Assemblathon[45]，中，可以得出：

1) 数据的质量对组装结果的影响远大于组装算法本身。

2) 组装结果的连续度在不同的组装算法和基因中也存在较大差异。

3) 组装的正确性需要通过排列到参考基因进行比较，经过比较发现不同组装算法的组装结果各有优劣，算法结果的正确性也不统一，而且在不同基因之间还存在较大差异。

这些差异性进一步说明，基因组装还远远没有结束，正确，高效的组装全基因组的问题还非常棘手。现在大家普遍使用的标准为 contigN50，一般认为 N50 越大，组装的结果越好，组装算法的性能会越好，事实上，contigs 只是组装的中间结果，N50 的大小并不能完全衡量组装的优劣。比如：如果 contig 连接错误而导致 N50 迅速增加时，这个标准就不正确了，因此，衡量组装算法优劣的标准过于单一，全基因组装算法迫切需要一个“黄金标准”用以衡量组装算法的优劣，遗憾的是目前还没有这样的标准。因此全基因组装问题还有很长一段路要走，它需要基因测序技术的不断改进，增加读取长度，降低测序的误差率将有助于基因组装结果的改进。

## 参考文献 (References)


[1] M. L. Metzker. Sequencing technologies—The next generation. Nature Review Genetic, 2010, 11(1): 31-46.

[2] E. R. Mardis. The impact of next-generation sequencing technology on genetics. Trends in Genetics, 2008, 24(3): 133-141.

[3] O. Morozova, M. A. Marra. Applications of next-generation sequencing technologies in functional genomics. Genomics, 2008, 92(5): 255-264.

[4] R. L. Strausberg, S. Levy and Y. H. Rogers. Emerging DNA sequencing technologies for human genomic medicine. Drug Discovery Today, 2008, 13(13-14): 569-577.

[5] E. Pettersson, J. Lundeberg and A. Ahmadian. Generations of sequencing technologies. Genomics, 2009, 93(2): 105-111.

[6] C. A. Hutchison III. DNA sequencing: Bench to bedside and beyond. Nucleic Acids Research, 2007, 35(18): 6227-6237.

[7] M. Pop. Genome assembly reborn: Recent computational challenges. Briefings in Bioinformatics, 2009, 10(4): 354-366.

[8] R. Staden. A strategy of DNA sequencing employing computer programs. Nucleic Acids Research, 1979, 6(7): 2601-2610.

[9] N. J. Loman, R. V. Misra and T. J. Dallman. Performance comparison of benchtop high-throughput sequencing platforms. Nature Biotechnology, 2012, 30(5): 434-439.

[10] C. Alkan, S. Sajjadian and E. E. Eichler. Limitations of next-generation genome sequence assembly. Nature Methods, 2011, 8(1): 61-65.

[11] P. Green. Whole-genome disassembly. Proceedings of the National Academy of Sciences of the USA, 2002, 99(7): 4143-4144.

[12] T. J. Treangen and S. L. Salzberg. Repetitive DNA and next-generation sequencing: Computational challenges and solutions. Nature Reviews: Genetics, 2012, 13(1): 36-46.

[13] D. A. Wheeler, et al. The complete genome of an individual by massively parallel DNA sequencing. Nature, 2008, 452: 872-876.

[14] D. R. Bentley, et al. Accurate whole human genome sequencing using reversible terminator chemistry. Nature, 2008, 456(7218): 53-59.

[15] Y. Benjamini, T. P. Speed. Summarizing and correcting the GC content bias in high throughput sequencing. Nucleic Acids Research, 2012, 40(10): e72.

[16] M. Pop. Genome assembly reborn: Recent computational challenges. Briefings in bioinformatics, 2009, 10(4): 354-366.

[17] X. Huang, S. P. Yang. Generating a genome assembly with PCAP. Current Protocols in Bioinformatics, 2005, 11: 3.

[18] D. Hernandez, P. Francois, L. Farinelli, M. Steras and J. Schrenzel. *De novo* bacterial genome sequencing: Millions of very short read assembled on a desktop computer. Genome Research, 2008, 18(5): 802-809.

[19] F. Paul, B. Ewan. Sense from sequence reads: Methods for alignment and assembly. Nature Methods, 2009, 6(11): S6-S12.

[20] S. Batzoglou, D. B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J. P. Mesirov and E. S. Lander. ARACHNE:


 

A whole-genome shotgun assembler. Genome Research, 2002, 12(1): 177-189.

[21] R. L. Warren, G. G. Sutton, S. J. M. Jones and R. A. Holt. Assembling millions of short DNA sequences using SSAKE. Bioinformatics. 2007, 23(4): 500-501.

[22] J. C. Dohm, C. Lottaz, T. Borodina and H. Himmelbauer. SHARCGS, a fast and highly accurate short-read assembly algorithm for denovo genomic sequencing. Genome Research, 2007, 17(11): 1697-1706.

[23] W. R. Jeck, J. A. Reinhardt, D. A. Baltrus, M. T. Hickenbotham, V. Magrini, et al. Extending assembly of short DNA sequences to handle error. Bioinformatics, 2007, 23(21): 2942-2944.

[24] D. W. Bryant Jr., W. K. Wong and T. C. Mockler. QSRA: A quality-value guided *de novo* short read assembler. BMC Bioinformatics, 2009, 10: 69.

[25] G. G. Sutton, O. White, M. D. Adams and A. R. Kerlavage. TIGR assembler: A new tool for assembling large shotgun sequencing projects. Genome Science and Technology, 1995, 1(1): 9-19.

[26] M. J. P. Chaisson, P. A. Pevzner. Short read fragment assembly of bacterial genomes. Genome Research, 2007, 18(2): 324-330.

[27] J. Butler, I. MacCallum, M. Kleber, I. Shlyakhter, M. K. Belmonte, et al. ALLPATHS: *De novo* assembly of whole-genome shotgun microreads. Genome Research, 2008, 18(5): 810-820.

[28] D. R. Zerbino, E. Birney. Velvet: Algorithms for *de novo* short read assembly using de Bruijn graphs. Genome Research, 2008, 18(5): 821-829.

[29] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, et al. ABySS: A parallel assembler for short read sequence data [J]. Genome Research, 2009, 19(6): 1117-1123.

[30] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, H. Yang and J. Wang. *De novo* assembly of human genomes with massively parallel short read sequencing, Genome Research, 2009, 20(2): 265-272.

[31] S. Steven, L. Phillippy, M. Adam and Z. Aleksey. GAGE: A critical evaluation of genome assemblies and assembly. Genome Research, 2012, 22(3): 557-567.

[32] S. Batzoglou. Algorithmic challenges in mammalian genome sequence assembly. In: M. Dunn, L. Jorde, P. Little and S. Subramaniam, Eds., Encyclopedia of genomics, proteomics and bioinformatics. Hoboken: John Wiley and Sons, 2005.

[33] M. Pop. McGraw-Hill 2006 yearbook of science and technology. New York: McGraw-Hill, 2005.

[34] G. Sutton, I. Dew. Shotgun fragment assembly. In: I. Rigoutsos and G. Stephanopoulos, Eds., Systems biology: Genomics. New York: Oxford University Press, 2007: 79-117.

[35] B. Ewing, L. Hillier, M. C. Wendl and P. Green. Base-calling of automated sequencer traces using phred. I. Accuracy assessment. Genome Research, 1998, 8(3): 175-185.

[36] E. S. Lander, M. S. Waterman. Genomic mapping by fingerprinting random clones: A mathematical analysis. Genomics, 1988, 2(3): 231-239.

[37] M. C. Wendl, R. H. Waterston. Generalized gap model for bacterial artificial chromosome clone fingerprint mapping and shotgun sequencing. Genome Research, 2002, 12(12): 1943-1949.

[38] M. Pop. Genome assembly reborn: Recent computational challenges. Briefings in Bioinformatics, 2009, 10(4): 354-366.

[39] M. Pop, S. L. Salzberg. Bioinformatics challenges of new sequencing technology. Trends in Genetics, 2008, 24(3): 142-149.

[40] M. Sahli, T. Shibuya. Arapan-S: A fast and highly accurate whole genome assembly software for viruses and small genomes. BMC Research Notes, 2012, 5: 243.

[41] P. E. C. Compeau, P. A. Pevzner and G. Tesler. How to apply de Bruijn graphs to genome assembly. Nature Biotechnology, 2011, 29: 987-991.

[42] H. Fleischner. Eulerian graphs and related topics. London: Elsevier Science, 1990.

[43] P. A. Pevzner, H. X. Tang and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences of the USA, 2001, 98(17): 9748-9753.

[44] P. A. Pevzner, H. Tang. Fragment assembly with double-barreled data. Bioinformatics, 2001, 17(1): S225-S233.

[45] D. Earl, K. Bradnam, J. St. John, et al. Assemblathon 1: A competitive assessment of *de novo* short read assembly. Genome Research, 2011, 21(12): 2224-2241.

 