CGDNA:基于簇图的基因组序列集成拼接算法

徐 魁[1] 陈 科[1] 徐 君[2] 田佳林[1] 刘 浩[1] 王宇凡[1]
(天津工业大学计算机科学与软件学院 天津 300387)[1] (南开大学数学科学学院 天津 300071)[2]

摘 要 基因组测序的目的是获取一个生物体完整的DNA序列信息,而DNA信息是进行遗传学研究和疾病诊断的基础。通常而言,完整的基因组测序分为两个步骤:第一步通过实验手段测定DNA序列片段,第二步通过计算方法把DNA片段拼接为完整的基因组。尽管桑格测序技术成功解析了包括人类在内的多个基因组,但其由于成本过高,目前逐渐被新一代测序技术所取代。新一代测序技术的特点为高通量、高覆盖率、低成本,随之而来的缺点体现为短读长、更多类型的错误。这些特点也给基因拼接算法带来了更大的挑战。鉴于目前的数十种基因拼接算法中并没有一种算法显著优于其它算法,且一些分析表明不同算法的拼接结果具有互补性,提出了CGDNA算法框架,它把不同算法的拼接结果整合到一起,使得整合的结果超越任何单个算法的结果。提出了一种基于簇图的基因组序列集成拼接算法,它通过构建索引、读长映射、重叠群聚簇、构建簇图等步骤将重叠群拼接成更长的序列。实验结果表明,相对于目前最优的算法Velvet、ABySS、SOAPdenovo,CGDNA在N50与最长拼接序列这两项指标上的增长比例高达50%以上,并且达到了较高的覆盖度。当更多的基本算法集成到本算法时,性能可进一步提高。提出的方法大幅提高了基因拼接的长度,为下一步的遗传分析降低了难度,并加快了生物基因组研究的步伐。

关键词 基因组拼接,集成算法,簇图,索引,读长映射
中图法分类号 TP399 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.9.045

CGDNA:An Ensemble De Novo Genome Assembly Algorithm Based on Clustering Graph

XU Kui[1] CHEN Ke[1] XU Jun[2] TIAN Jia-lin[1] LIU Hao[1] WANG Yu-fan[1]
(School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China)[1]
(School of Mathematical Sciences, Nankai University, Tianjin 300071, China)[2]

Abstract The ultimate goal of genome sequencing is to determine the complete DNA sequence of an organism, which is the basis for genetic research and disease diagnosis. In general, genome sequencing can be divided into two steps: first, generating and determining the DNA fragments experimentally; second, assembling the fragments into full genome through computational method. Although the Sanger technology successfully resolves the human genome, it is replaced by the next generation of sequencing technology due to its high cost. The next generation of sequencing technology has the merits of high throughput, high coverage and low cost and accompanies with short reads and more errors as a by-product, which brings more challenge to the assembly algorithms. Since it is reported that the assembly results by different algorithms are complementary and none of the assembly algorithms consistently outperforms the remaining algorithms, this study aimed at integrating the assembly results produced by multiple algorithms. In this study, we proposed an algorithm based on clustering graph. Through building index, mapping of reads, clustering of contigs and building of clustering graph, the proposed algorithm outperforms any of the single algorithm. The experimental results demonstrate that by implementing the CGDNA algorithm, two standard metrics (the largest scaffold and scaffold N50) are increased by 50% when compared to the state-of-the-art algorithms, i.e., Velvet, ABySS, and SOAPdenovo. Moreover, the performance of CGDNA algorithm should be further improved when more base algorithms are added. The proposed algorithm largely improves the quality of assembly result, reduces the difficulty of genetic analysis and accelerates the genome research.

Keywords De novo genome assembly, Ensemble algorithm, Clustering graph, Indexing, Read mapping

---

到稿日期:2014-05-15 返修日期:2014-07-21 本文受国家自然科学基金(11201134),天津市自然科学基金一般项目(12JCYBJC31900),天津市高校中青年骨干创新人才培养计划资助。

徐 魁(1990—),男,硕士生,主要方向为计算生物学、机器学习,E-mail:kuixu.tj@gmail.com;陈 科(1982—),男,博士,副教授,主要研究方向为计算生物学、机器学习、计算机视觉;徐 君(1988—),男,博士生,主要研究方向为计算机视觉、机器学习、字典学习;田佳林(1990—),男,硕士生,主要研究方向为机器学习;刘 浩(1991—),男,硕士生,主要研究方向为计算机视觉、机器学习;王宇凡(1993—),男,硕士生,主要研究方向为数据挖掘。

1 引言

高通量的全基因组序列拼接仍然是一个难题，尤其是基因组序列的从头拼接(De Novo Assembly)，它是一个 NP 难题[1]。基因组测序的目的是确定 DNA 的碱基序列，其通常可分为两个步骤：1)把 DNA 分割成一些碎片段，然后分别对这些碎片段进行测序；2)使用一些算法把这些碎片拼接成完整的基因组。因此，DNA 序列拼接是完成基因组测序的关键步骤之一。基于化学的第一代测序技术——桑格测序法得到的读长(碎片)长度大约在 500 至 1000 个碱基之间。这种较长的读长有利于基因拼接算法，并能降低拼接消耗的时间。然而由于桑格测序法非常耗时、耗材，并且需要昂贵的测序仪器和专业的专家来确保测序的可能性，这种方法目前基本被新一代测序技术所取代。包括 Illumina、Complete Genomics[2]、Helicos[3]、454 Life Sciences[4]、SOLID[5]、Ion Torrent[6]在内的多个公司设计和实现了新一代的技术，这些技术具有便宜、快捷、高通量等优点，但以牺牲读长的长度为代价。新一代测序技术所导致的海量的、短小的、包含错误的读长数据增加了基因序列拼接的难度。

基因序列的拼接问题通常被转化为图论的问题，目前的方法主要分为 3 类。第一类为基于重叠图的算法。这类方法通常包含 4 个步骤：删除重复的读长、构建重叠图、简化重叠图以及生成重叠群。此类方法主要包括 Edena[7]和 Newbler[8]。第二类方法基于贪心图。此类方法首先设定一个或多个种子，然后添加具有最好重叠度的读长并不断延伸。这类方法大幅降低了拼接图的复杂度，然而可能得不到最优解，主要包括 SSAKE[9]、SHARCGS[10]、VCAKE[11]、OligoZip[12]、Telescoper[13]等。第三类方法基于 de Bruijn 图，这也是目前最为流行的方法。此类方法首先把读长转化为一些长度为 k 的片段(k-mer)，然后使用 k-mer 构建图，这种数据结构压缩了读长的数据，且不用保存读长之间的连接关系，大幅降低了对内存的依赖。其包括 EULER-USR[14]、AllPaths LG[15]、ABySS[16]、Velvet[17]、SOAPdenovo[18]等。

尽管已有多种方法被提出，近来的两篇比较性研究[19,20]指出目前尚没有一种方法在不同的数据集上总是优于其它算法。此外，我们也注意到不同算法产生的拼接结果具有一定的互补性。为此，提出了一种基于簇图的算法，旨在整合来自不同算法的拼接结果。我们称该算法为 CGDNA(Clustering Graph based De Novo Assembly)。CGDNA 通过将原始短读长序列映射至重叠群，从而将较长的难以比对的重叠群序列用所映射的短读长集合来表示，这样就很容易根据短读长集合计算重叠群的相似度；然后将相似度达到阈值的重叠群聚到一个或多个簇中，并为每一个簇构建一个簇图，通过求解簇图的最长路径，最终得到更长的结果序列。CGDNA 在多个模块中均采用了自行设计的多线程框架实现加速。不同算法间的互补性使得本文提出的算法可以得到比其他任何算法更好的结果，这便是 CGDNA 算法最大的意义所在。

本文第 2 节给出问题的形式化描述；第 3 节介绍 CGDNA 算法的设计与实现；第 4 节介绍实验数据的处理方法和实验结果，并对其进行分析；第 5 节针对本文的模型和存在的问题进行讨论；最后给出结论。

2 问题形式化描述

对于一条给定的染色体，其基因序列可以表示为$\{S_i\}$ $(i=1,2,\cdots,n)$。一个读长为$\{S_i\}$的连续子串的长度通常为 100bp，譬如$\{S_i\}(i=k,k+1,\cdots,k+99)$。本文处理的数据为读长对，可以表示为$\{S_i\}$的两个子串，这两个读长之间的间隔在一定范围内浮动，平均间隔为 500。一个染色体通常可以测量很多次，因此可以生成许多读长对，读长对的起始位置并不固定。我们称所有读长对组成的集合为 I，该集合即为测序平台给出的数据。

在使用序列拼接方法后，许多读长对可拼接为较长的字符串，每种测序方法的拼接结果可以表示为$\{R_i^j\}(i=1,2,\cdots,m)$。其中 j 表示第 j 种拼接算法，m 表示该算法的拼接结果共生成 m 条字符串，这些拼接得到的字符串称为重叠群(contig)。

本文的问题可描述为：给定读长对的集合 I 以及不同拼接算法的结果$\{R_i^j\}$，能否综合不同算法的拼接结果，使得新结果超越任何单个算法的拼接结果。

3 CGDNA 算法

3.1 算法思想

CGDNA 算法的框架如图 1 所示。该算法可以分为如下几个步骤：

第一，数据预处理，包括生成高质量的读长对集合 I 和不同的拼接算法的结果。

第二，为拼接算法生成的重叠群建立索引。

第三，建立读长对的集合 I 与不同拼接算法的重叠群之间的映射，即生成一张索引表，表明每个重叠群都由哪些读长对拼接而来。

第四，重叠群聚簇，将来源于不同拼接算法的重叠群分组，其中组内的重叠群都是相关的，而组与组之间的重叠群无关，每个组称为一个簇。

第五，构建簇图，并生成每个簇的最长路径。每个簇内的重叠群是相关的，即表明它们是互补的，可进一步拼接。最终采用回溯法求解每个簇的最长路径并生成拼接结果。

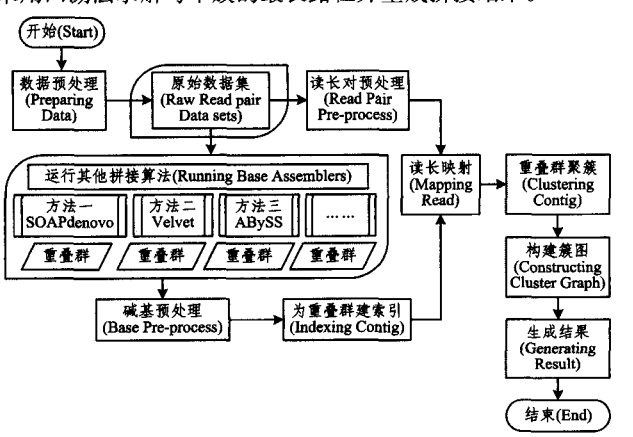

图 1 CGDNA 算法的框架

在下面的章节中，我们将对这 5 个步骤做详细的说明。

3.2 数据预处理

CGDNA 算法的输入包括两部分：1)原始的短读长对(Short Read Pair)数据，可以在美国国家生物技术信息中心(National Center of Biotechnology Information，NCBI)网站上

获得，这部分数据由实验测序得到；2)来自多个基本拼接算法生成的重叠群，即拼接算法应用到第一部分数据上生成的结果。具体的参数配置参见 4.1 节。本文实验中用到了 3 个拼接算法 ABySS、SOAPdenovo，Velvet，这 3 种算法代表了目前从头拼接的最高水平。ABySS 和 SOAPdenovo 的拼接结果包含两种：重叠群和 scaffolds，Velvet 的结果只含有重叠群，其中 scaffolds 是由重叠群进一步拼接而来。为便于描述，下文中重叠群用 contig 表示。

3.3 重叠群构建索引

重叠群索引构建即是把来自不同算法的重叠群序列综合到同一个索引结构中，这个索引结构将为下一步读长映射提供基础。对于每一个 contig，首先生成其对应的 k-mer 序列集合，实验中 k 设定为 30。索引的键为 k-mer 序列（“碱基空间”）转化的四进制整数；索引的值为该 k-mer 序列在重叠群集合中的位置，如图 2 所示。将索引写入索引文件，一方面减少了内存的开销，另一方面使得生成的过程文件可被重复使用。整个索引构建过程，采用多线程机制，进行多线程构建索引，不仅降低了构建索引的时间，也使得 CGDNA 算法整体的运行时间大幅度降低。

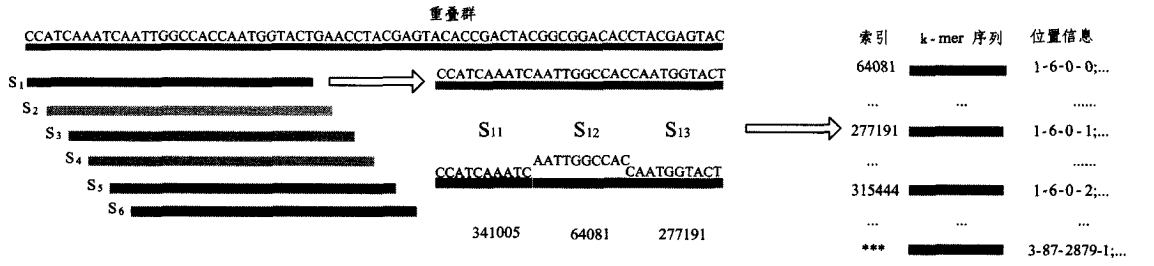

图 2 索引表

3.4 读长映射

在本步骤，我们将利用索引表将原始的短读长对映射到重叠群上。在此，我们使用带容错的匹配。对于一个给定的读长对，首先截取左读长的最右端的 30 个字符，查找它在索引表中对应的重叠群，此处容许两个字符的差异。同样截取右读长的最左端 30 个字符，查找索引表找到相应的重叠群，同样可容忍两个字符的差异，如图 3 所示。使用容错策略主要考虑到测序并不是百分之百准确。如果左读长与右读长映射的位置符合预期，则通常左读长与右读长的间隔为 400～600，判断该读长对正确映射到了 contig 上。在映射后，每个 contig 对应于读长对集 I 中的一个子集 $CR_{i,j}$。

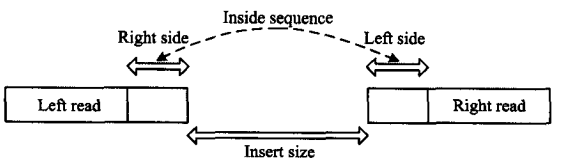

图 3 读长对的内部序列

3.5 重叠群聚簇

在此步骤中，我们将实现两个目标：第一，哪些 contig 之间是有关联的，即可进一步拼接；第二，相关联的 contig 的精确匹配。使用上一步映射得到的读长对的集合 $CR_{i,j}$ 作为每个 contig 的特征，通过 $CR_{i,j}$ 集合之间的关系来寻找 contig 之间的关系。简言之，如果两个 contig 使用大量的相同的读长对，则这两个 contig 的拼接内容是关联的。

不同 contig 之间的相关性计算方法如下：给定两个 contig，它们的序列长度表示为 $length(C_{i,j})$ 与 $length(C_{i',j'})$，它们映射得到的读长对集合表示为 $CR_{i,j}$ 与 $CR_{i',j'}$，则两个 contig 之间的相关度定义为

$$\lambda=\frac{|CR_{i,j}\cap CR_{i',j'}|}{\min\{length(C_{i,j}),length(C_{i',j'})\}} \tag{1}$$

当 λ 超过一个给定阈值 θ，就表示两个 contig 是相关的、潜在的、可拼接的，并把它们定义为 contig 对（Contig Pair）。我们采用单链接的方式对所有 contig 进行聚类，相关的 contig 被聚到一个簇中，而不同簇中的 contig 都是不相关的。

Contig 之间的精确匹配计算方法如下：采用动态规划策略计算 contig 对中的映射区域的最长公共子序列，求解最长公共子序列过程而建立的递归关系如式(2)所示。计算结果可以得到两个 contig 之间的位置匹配信息，包括匹配上的子序列所在区域长度和起始位置。在此过程中，用 $c[i][j]$ 记录重叠群对中 $Col_{i,j,1}$ 和 $Col_{i,j,2}$ 的最长公共子序列的长度。式(2)中分别用 x 和 y 表示 $Col_{i,j,1}$ 和 $Col_{i,j,2}$。

$$c[i][j]=\begin{cases}0, & \text{if } i=j=0\\ c[i-1][j-1], & \text{if } i,j>0;x[i]=y[i]\\ \max\{c[i-1][j-1]-1,0\}, & \text{if } i,j>0;x[i]\neq y[j];x[i]\parallel y[j]=N\\ \max\{c[i-1][j-1]-5,0\}, & \text{if } i,j>0;x[i]\neq y[j]\neq N\end{cases} \tag{2}$$

在该过程中再次进行了重叠群对的筛选，移除匹配的子序列的长度小于某一阈值的 contig 对。在本步骤中，我们找到了相关的 contig 以及它们之间的精确匹配，并把相关的 contig 聚类到一些簇中。

3.6 构建簇图

在本步骤，我们将一个簇中 contig 之间的关系转化为图论的问题。簇图 G 记作 $G=(V,E)$，其中 V 中的顶点表示 contig 或 contig 的子串，有向边表示顶点之间的连接关系。对于一个重叠群对的重叠区域，其公共子序列表示为 $\{Z_i\mid i=1,2,\cdots,k\}$，非公共的部分在重叠群 A 与 B 中可分别表示为 $\{X_i\mid i=1,2,\cdots,k+1\}$ 与 $\{Y_i\mid i=1,2,\cdots,k+1\}$，这个重叠群对的比对关系在图中产生 $\{X_i\}$，$\{Y_i\}$，$\{Z_i\}$ 共 $3k+2$ 个顶点。如图 4 所示，该例子中两个重叠群在重叠区域有一段公共子序列，因此共生成 $3*1+2=5$ 个顶点。该图为重叠群的最小子簇图。然而，大多数重叠群对会有多个匹配的重叠区域，即会产生多个最小子簇图。多个最小子簇图合并成簇图。不断地把新的重叠群对添加到生成的图中，直到所有重叠群对全部处理完成，得到一个簇的簇图。

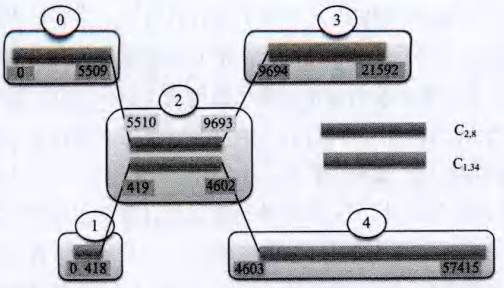

图 4　重叠群的最小子簇图

我们采用回溯法搜索簇图的最长路径，每个簇图的最长路径即为 CGDNA 拼接得到的 scaffold(Sd_i)。最终所有簇图生成的 scaffold 的集合{ Sd_1, Sd_2, …, $Sd_{C\text{-}Count}$ }即为算法生成的结果。

构建簇图算法的伪代码如下。

算法　构建簇图

输入：CPairs：collection of contig-pairs in a cluster icc∈$CC^{C\text{-}Cout}$

初始化：set gid=0，itype=0，and initialize Cluster Graph as CG

1. for i in SPairs
2.　　for j in SPairs[i]. Region
3.　　SGi,j ←getSubGraph(SPairs[i]:Region[j]);
4.　　itype ←getMergeType(SGi,j;CG);
5.　　ConstructGraph(gid;itype;CG)
6.　　gid ← gid+4
7.　　end. j ←j+1
8. end. i ←i+1

输出：Final structure of Cluster Graph icc CGicc=CG

4　数据处理和实验结果

本节将介绍 CGDNA 算法的运行环境、数据集准备、数据预处理等具体步骤，并将实验结果和基本拼接算法（ABySS、SOAPdenovo、Velvet 等）进行比较和分析。

在分子生物学、生物化学、遗传学和生物技术等研究领域中，大肠杆菌 K-12 是一种最普遍的生物模型，早在第一代测序技术时就利用桑格双脱氧法测序技术得到了其参考基因序列，本文用到的大肠杆菌 K-12 MG1655（NCBI SRA accession ERR022075）数据就是基于第二代测序技术在 Illumina Genome Analyzer IIx 平台上重测序得到的读长对序列，读长长度为 100bp，测试深度为 990X，左右读长间隔平均为 500bp，其基因组的总长度大约是 4639675bp，错误率是 0.2%左右。它是一个在基因组从头测序实验中使用非常广泛的数据集，可直接在美国国家生物技术信息中心（National Center of Biotechnology Information）官网下载。

4.1　数据集和数据预处理

基因序列数据的预处理包括处理从 NCBI 官方网站得到的数据源文件及生成本文算法的输入数据（contig），下面做简要介绍。

4.1.1　数据处理一

使用 SRA Toolkit 工具包将下载得到的 ERR022075. sra 文件转化成常用的 fasta 格式的文件。经过转换得到的 fasta 数据文件将会作为本文算法的两种输入数据文件之一，另一种数据文件是数据处理二中运行其他算法得到的 contigs。

4.1.2　数据处理二

将数据处理一中得到的结果文件（fastq 和 fasta）作为输入，配置 Velvet、ABySS、SOAPdenovo 相关的参数（参数选择可参考各个基本拼接算法的使用文档），以 Velvet、ABySS、SOAPdenovo 作为拼接的基本算法，得到拼接的产生文件 contigs。contigs 将会作为本文算法的输入。

测试多组参数，如表 1 所列。根据最长 Contig 指标，选取各个算法的最优结果作为 CGDNA 输入。最优的结果在表 1 中用粗体表示。

表 1　实验参数和结果

实验编号	算法名称	参数选择	k-mer	内存(GB)	耗时(h)	Contig N50	最长 Contig	总长度	覆盖率
#Aby1	ABySS	maxk=64	31	5	3.5	113989	266089	4593976	99.02%
#Aby2	ABySS		50	8	4	**115397**	**266089**	4639233	**99.99%**
#Vel1	Velvet	ins_length=500	31	18	3	15622	66436	4539522	97.84%
#Vel2	Velvet	cov_cutoff=33	57	20	3	**58896**	**166049**	4556708	**98.21%**
#Vel3	Velvet	min_contig=200	61	18	3	15622	66436	4539522	97.84%
#Soa1	SOAPdenovo	avg_ins=500	31	8	3	102	676	15315129	330.09%
#Soa2	SOAPdenovo	reverse_seq=1	51	18	4	176	5752	7532955	162.36%
#Soa3	SOAPdenovo	asm_flags=3	61	18	4	1532	9507	5705309	122.97%
#Soa4	SOAPdenovo	rank=1	77	15	2	**14074**	**58258**	4753931	**102.46%**

4.2　实验结果

本文采用将 ABySS、SOAPdenovo、Velvet 3 种算法的结果进行融合的策略，利用其大量存在重复区域的特性，对 3 种基本拼接算法生成的 contig 进行重新拼接，得到更长的序列 scaffold，本文提出的算法结果相较于以上 3 种算法都有较大幅度的提高。

表 2 列出了所有的实验结果，实验 #CGDNA1 和 #CGDNA1 是由本文算法将 ABySS、SOAPdenovo、Velvet 的实验 #Aby2、#Vel2、#Soa4 产生的 contig 进行融合得到的，实验 #CGDNA2 是在 #CGDNA1 的基础上优化了聚簇过程中的相关度的阈值。实验 #CGDNA3 使用 SOAPdenovo 程序，并将在实验 #Soa1 的参数下生成的 contig 作为输入。

表 2　Scaffold 及各项指标的比较

实验编号	Scaffold N50	最长 scaffold	总长度	覆盖率
#Aby2	134011	266089	4639293	99.99%
#Vel2	58896	166049	4556708	98.21%
#Soa4	125196	267369	4628146	99.75%
#CGDNA1	196852	392298	4639068	99.99%
#CGDNA2	268965	458120	4639365	99.99%
#CGDNA3	213406	400460	11899013	256.46%

分析实验结果，可以得到以下结论：

（1）使用基本算法中实验结果较好的 contig，CGDNA 能得到更优的结果。

（2）ABySS 和 SOAPdenovo 算法的结果非常的接近，它们产生的 contig 和 scaffold 具有高度的相似性。

(3)Velvet 产生的结果与 ABySS、SOAPdenovo 有显著差异，这种情况可推测 Velvet 和其他两个算法的结果具有较高的互补性，因此将 Velvet 和其他两个算法进行融合可得到更高质量的拼接结果。

(4)从 #CGDNA 的结果和图 5 可以看出，融合之后的结果提高非常显著。图 6 显示融合之后最长的 scaffold 和 scaffold N50 都相对提高超过 50%，相对于 Velvet，最长的 scaffold 增长率达到了 141.2%，scaffold N50 增长率达到了 262.3%。图 7 示出了 3 种方法融合之后的显著效果，簇图 1 和簇图 9 的增长率分别达到了 80.2%和 97.7%。

(5)图 8 与图 9 显示在 SOAPdenovo 和 Velvet 融合后，根据映射的信息聚成了 65 个簇，每个簇都会构建各自的簇图，通过搜索簇图的最长路径得到扩展之后的序列 scaffold，图 8 与图 9 示出了每个簇图产生的 scaffold 相对于簇内最长 contig 的增长率，即每个簇图在融合后相对于融合前的增长率。我们注意到簇图 6 的增长率高达 55.6%。

(6)实验结果显示，CGDNA 利用多个算法之间的互补性有效地提高了基因组拼接的质量。多个算法在不同的参数、不同的数据集上均表现出不同的优势，CGDNA 的思想很好地利用了各个基本算法的优势从而达到了提高基因组拼接质量的目的。据此可以推测，倘若更多的算法加入进来，将获得更优的结果，更快地得到完整的基因组序列将成为可能。

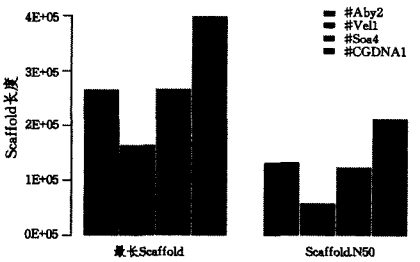

图 5 CGDNA 在 ScafflodN50 和最长 Scafflod 长度两项指标上与 ABySS、SOAPdenovo、Velvet 的比较

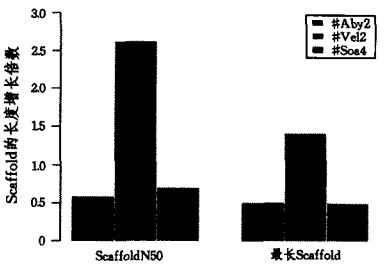

图 6 CGDNA 在 ScafflodN50 和最长 Scafflod 两项指标上相对于 ABySS、SOAPdenovo、Velvet 的增长比率

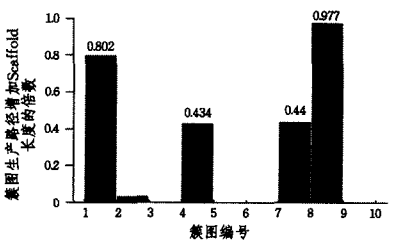

图 7 在 ABySS、SOAPdenovo 和 Velvet 集成后产生的每个簇图生成 scaffold 的增长比率

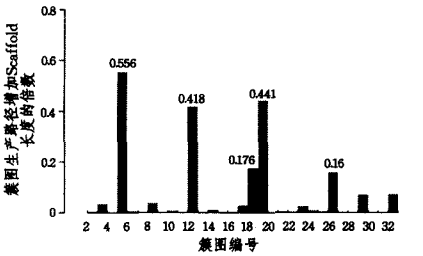

图 8 在 SOAPdenovo 和 Velvet 集成后产生的每个簇图生成 scaffold 的增长倍数

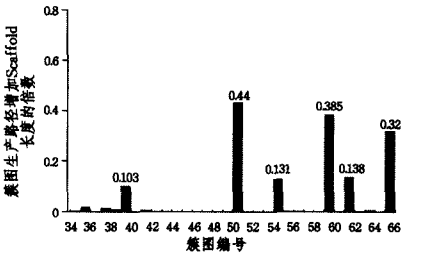

图 9 在 SOAPdenovo 和 Velvet 集成后产生的每个簇图生成 scaffold 的增长倍数

5 讨论

针对新一代测序技术产生的序列具有高通量、短读长的特点，基因组序列拼接可用的数据信息受限，外加上不同生物的基因结构的复杂程度和重复度各异，当前的拼接软件分别都在符合各自特点的生物数据上得到了比较理想的结果，然而对于同一数据，不同算法的结果相差甚远。本文提出了一种集成多个拼接算法的方法，其基于不同拼接方法结果的互补性。实验结果显示它大幅度增加了拼接序列的长度。但是，由于序列拼接问题难度较大，仍存在部分问题需要进一步分析和解决。

下一步研究工作主要包括以下几点：

(1)原始数据中存在的错误问题

在基因测序产生序列片段的过程中，由于光谱等因素，导致难以避免的碱基识别错误，也导致原始读长数据有一定的错误率，一般在 0.2%左右。虽然算法在使用读长进行映射的过程中，对可能会错的读长进行了一些处理，但是对于映射仍然还会有一定的影响，尤其是对于读长中存在大量碱基缺失的情况，带来了一定的映射缺失、映射错误、簇图分支等问题。如何高质量地对读长进行错误纠正，直接影响着基因序列拼接的质量。后续的工作中将尝试对多个不同测序深度的大肠杆菌 K-12 MG1655 基因组数据集进行实验，并将一些纠错算法(例如 ECHO)加入到数据预处理中，从而增加数据的质量。

(2)序列比对内存消耗过剩问题

文中在寻找潜在的、可拼接的 contig 对时，采用了动态规划的方法求解 contig 对中两个 contig 中读长映射区域序列的最长公共子序列，算法能得到全局的最优解，但是当两条序列长度过长时，算法将因内存受限而终止。遇到这种情况时，本文通过线程等待和重启的方式，将当前内存受限的序列进行排队，等到有足够的空置内存时再重启线程，完成比对工作。这种方式虽然能暂时性地解决内存受限问题，但是一旦受主

(下转第 245 页)

timization for NO_x Emission Model of power Plant Boilers Based on Gravitational Search Algorithm [J]. Journal of Chinese Society of Power Engineering, 2013, 33(2): 100-106

[10] 李沛,段海滨. 基于改进万有引力搜索算法的无人机航路规划[J]. 中国科学,2012,42(10):1130-1136
Li P, Duan H B. Path planning of unmanned aerial vehicle based on improved gravitational search algorithm[J]. Sci China Tech Sci, 2012, 42(10): 1130-1136

[11] Jiang S, Ji Z, Shen Y. A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints [J]. Electrical Power and Energy Systems, 2014, 55: 628-644

[12] 李春龙,戴娟,潘丰. 引力搜索算法中粒子记忆性改进的研究[J]. 计算机应用,2012,32(10):2732-2735
Li C-L, Dai J, Pan F. Analysis on improvement of particle memory in gravitational search algorithm [J]. Journal of Computer Applications, 2012, 32(10): 2732-2735

[13] 徐星,李元香,姜大志,等. 一种基于分子动理论的改进粒子群优化算法[J]. 系统仿真学报,2009,21(7):1904-1907
Xu X, Li Y X, Jiang D Z, et al. Improved Particle Swarm Optimization Algorithm Based on Theory of Molecular Motion [J]. Journal of System Simulation, 2009, 21(7): 1904-1907

[14] Rashedi E. Fuzzy Gravitational Search Algorithm [C]// 2012 2nd International Conference on Computer and Knowledge Engineering(ICCKE). 2012: 18-19

(上接第 239 页)

机内存的限制,算法将无法执行。有的学者为了解决这个问题,采用了局部比对的方法,但这种方法很难得到全局的最优解。因此,这个问题仍面临着更大的挑战,需要更进一步的深入探讨和研究。

(3)探索方法之间的互补性

将进一步使用更多的基因数据和拼接算法做实验,后期将加入 Meraculous、IDBA-UD 等先进的算法,并试图从实验中找出拼接算法之间互补性的规律和结论,为选用哪几种拼接方法进行融合提供指导和推荐。

(4) 对 CGDNA 进行加速

在运行耗时上,CGDNA 优势并不明显。主要的耗时操作为构建索引和读长映射两个步骤,对其进行加速的工作迫在眉睫,目前正在使用 Hadoop 构建大规模并行框架来优化 CGDNA,以减少 CGDNA 的运行时间并提高运行效率。

结束语 基因序列拼接是全基因组测序的重要一环,其中基因组从头拼接难度远远大于重测序,也更具挑战性。在深入分析当前基因测序方法的特性以及拼接数据高通量、短序列的特点之后,结合当前序列拼接问题的研究现状,本文提出通过构建簇图的方式解决当前基因序列拼接的问题。

本文提出的 CGDNA 算法根据多个算法生成的重叠群之间的相关性,寻找潜在可拼接的重叠群,然后通过序列比对验证匹配区域的确切位置,从而确定可继续拼接的重叠群集合。通过构建簇图,将较难的序列拼接问题转化为在簇图上寻找路径的问题。

实验结果表明,本文提出的 CGDNA 算法可以显著提高单个拼接算法的性能,在最长的 scaffold 和 scaffold N50 这两项指标上提高幅度均超过 50%。由于 CGDNA 算法可兼容任意多个拼接算法,我们相信使用更多的基本拼接算法将进一步提高 CGDNA 算法的性能。

参 考 文 献

[1] Medvedev P, Georgiou K, Myers G, et al. Computability of Models for Sequence Assmbly[M]// Algorithms in Bioinformatics. Springer-Verlag Berlin Heidelberg, 2007: 289-301

[2] Drmanac R, et al. Human genome sequencing using unchained base reads on self-assembling DNA nanoarrays [J]. Science, 2010, 327(5961): 78-81

[3] Harris T D, et al. Single-molecule DNA sequencing of a viral genome[J]. Science, 2008, 320(5872): 106-109

[4] Margulies M, et al. Genome sequencing inmicrofabricated high-density picolitre reactors[J]. Nature, 2005, 437(7075): 376-380

[5] McKernan K J, et al. Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding[J]. Genome Res, 2009, 19(9): 1527-1541

[6] Medvedev P, et al. Paired de Bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers[M]// Research in Computational Molecular Biology. Springer, 2011: 238-251

[7] Hernandez D, Francois P, Farinelli L, et al. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer[J]. Genome Res., 2008, 18(5): 802-809

[8] Miller J R, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data[J]. Genomics, 2010, 95(6): 315-327

[9] Warren R L, Sutton G G, Jones S J, et al. Assembling millions of short DNA sequences using SSAKE[J]. Bioinformatics, 2007, 23(4): 500-501

[10] Dohm J C, Lottaz C, Borodina T, et al. A fast and highly accurate short-read assembly algorithm for de novo genomic sequencing[J]. Genome Res, 2007, 17(11): 1697-1706

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

[12] http://linux1.softberry.com/berry.phtml?topic=OligoZip

[13] Bresler M, Sheehan S, Chan A H, et al. Telescoper: De novo Assembly of Highly Repetitive Regions[J]. Bioinformatics, 2012, 28(18): 311-317

[14] Chaisson M J P, et al. De novo fragment assembly with short mate-paired reads: does the read length matter? [J]. Genome Res, 2009, 19(2): 336-346

[15] MacCallum I, et al. ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads[J]. Genome Biol, 2009, 10(10): R103

[16] Simpson J T, et al. ABySS: a parallel assembler for short-read sequence data[J]. Genome Res, 2009, 19(6): 1117-1123

[17] Zerbino D R, Birney E. Velvet: algorithms for de novo short-read assembly using de Bruijn graphs[J]. Genome Res, 2008, 18(5): 821-829

[18] Li R, et al. De novo assembly of human genomes with massively parallel shortread sequencing[J]. Genome Res, 2010, 20(2): 265-272

[19] Earl D A, et al. Assemblathon 1: a competitive assessment of de novo short-read assembly methods[J]. Genome Res, 2011, 21(12): 2224-2241

[20] Miller J, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data[J]. Genomics, 2010, 95(6): 315-27