基因拼接算法设计

金熙森
复旦大学, 计算机科学与技术学院
15307130053@fudan.edu.cn

1 简介

基因组测序是生物信息学一个重要的任务。有趣的是，它的性能不仅与生物学上技术有关，也与我们设计的算法有着密切的联系。目前的测序有两种流行的方式- Illumina Paired End 测序与 Pacbio single end 测序。基于他们出现的时间，我们分别将他们称作“二代基因测序”与“三代基因测序”技术。根据本次课程项目给出的数据，他们分别有如下的特点：

- 二代测序序列：长度较短（100bp），可以通过 Mate pair 信息大致估计双端序列的距离，复制次数多，错误率低。
- 三代测序序列：长度较长（1000bp），复制次数少，错误率较高。

我们很容易看到，二代测序序列与三代测序序列各有优点，且缺点互补。如果将二代测序序列与三代测序序列共同用于测序基因序列，想必会比只使用单个测序序列效果要优秀。然而，我们侧重于二代测序序列，因为尽管它比较短，但是错误率很低，这对我们的前期工作有着莫大的好处。

本次项目中，我以循序渐进的过程对这一问题进行了探索。首先，我探索了二代序列的检错纠错方式；其后，我引入三代序列，使用二代序列与三代长序列共同完成基因拼接的任务。

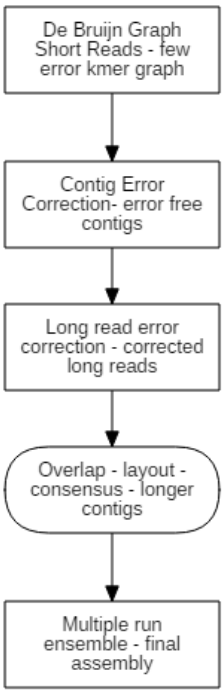

图 1. 所涉及的算法的流程图与各个流程以及各个流程的输出。圆角步骤引用了开源代码。

2 数据结构

尽管基因拼接的算法大多工程量较大，人们仍很容易想到一些直观的做法。例如，我们可以使用贪心算法，在测序的序列中不断查找与当前串重叠最大的串进行拼接。然而，这种算法典型地需要序列数目平方级以上的复杂度，因此对于未经压缩的几十万规模的二代测序序列压力很大；同时，这种算法不易进行纠错检错操作，我们可能需要重复地扫描整个数据集。为此，我使用一种建模子序列重叠的数据结构——De bruijn 图存储序列数据，并在该数据结构上设计算法。

De bruijn 图 $G(V, E)$ 的每个节点 u 存储一个 kmer（长度为 k 的字符串）。$(u, v) \in E$ 当且仅当 u.seq 的最后 $k-1$ 个字符与 v.seq 的前 $k-1$ 个字符相同。k 的大小可以设置。项目初期，我使用了双向 *De bruijn* 图用于显式建模 DNA 序列的双链互补特性。算法使用 python 语言实现，代码实现基于以下规则。记 s.cmpl 或 cmpl(s) 为 s 的反向互补序列。项目后期使用了单向 *De bruijn* 图。两种图在附带的代码中都有实现。

- Vertex 类，即 De bruijn 图的节点。它永远存储 s 和 s.cmpl 中字符串比较意义上较小的一个。因此，s 和

s.cmpl 构成一个等价类，记为 $\hat{s}$，下同。Vertex 全体通过 python 字典（底层实现为散列表）实现，键值为 $\hat{s}$。它维护 out_edge 表，记录出边，以及自身存放序列的方向。它额外有一个 cov(erage) 属性，用于记录被重复添加进图的次数。

- Edge 类，即 De bruijn 图的边，存储连接的两个定点对应 kmer 的等价类。双向 de bruijn 图通过属性 ori(entation) 存储入顶点与出顶点序列的方向。

3 算法

3.1 Reliable Short Contigs

首先介绍仅仅使用 De bruijn Graph 与 short reads 的算法。将所有的 short reads 打断为 kmers，通过上一章的算法读入 De bruijn Graph。读入每一个 reads 时，也同时将它的反向互补序列输入该图。这样，我们就得到了描述 kmers 之间重叠状况的表示。某些算法在 De bruijn 图中寻找欧拉回路来表示获得 contigs. 然而，经过实验后发现，该算法作为一个 baseline 得到的序列较短，在存在错误且复制错误较多的数据集更是如此。因此，为了构建算法的一个 strong baseline，我通过深度优先遍历的到一系列的 contigs：对 G 进行深度优先遍历，为每一个路径/或连通分量输出全部的 contigs. 通过这种算法，我们可以在 data1 得到大约 5000 的 NGA50. 这种算法是不可靠的，但是在指标意义上的确能取到一定质量的结果；然而，在 data2 与 data3 中，由于 short reads 中引入了读取片段，这种做法的成绩会直线下降，只能获得 400-500 左右的 NGA50.

有了一个比较的基线后，我们正式开始介绍算法。图 1给出了算法的大致流程图。首先，读入 kmers 并通过一定算法在 de bruijn 图中提取尽可能最可靠的序列。长度考虑是第二位的，这是因为我们在后续的过程中需要对长序列进行纠错，而长序列的长度本身长度已经足够长。因此，我们的任务转化为了如在 de bruijn 图中找到能保证无错误，或尽可能最小化错误的序列。有很多针对 bubble 类型错误（中间段错误），tip 类型错误（尾端错误）有许多流行的纠错算法，这些大多针对在 de bruijn 图中寻找回路等目的而设计。但是，由于我们的算法目的较为特殊，我们可以使用以下的算法进行几乎无错误的可靠序列：

1) 每次选定一个未访问过且度数不为 2 的点，进行 DFS。
2) 如果该点出度为 1，则继续遍历下一个节点
3) 如果该点出度为 2 或以上，对该点的出边以 coverage 为键值进行排序。如果最大 coverage 边的出度不为 1 且倒数第二个边的出度为 1，那么认为 coverage 最大的边为正确的边，其他边为错误边。沿着正确边继续遍历。
4) 如果上述条件不满足，或者该节点之前以访问过，输出路径上的的节点作为 contigs。

算法的第 3 步基于同一位置不会出现两次错误的假定。这是合理的，因为短序列的错误概率为 1% 到 2%，且复制次数为 10x 或 20x。这种情况下，同一点发生同一类错误的概率是很小的（尽管同一点发生不限定同类的错误的概率并不小）。但是，以这种方式，我们能保证输出很可靠的 contig 序列。在 data1 中，由于 short reads 没有错误，因此能得到若干 9000 长度的 contig. 在 data2 中，由于涉及错误，会产生大约 300 个长度为 100 以上的 contig，以及若干个长度为千级别的 contig。

算法中的第三步在不同的数据集上实现策略略微有些差异。例如对于 data1，由于不涉及错误，可以适当减弱退出循环的条件。

3.2 Pre-error correction with Aligning

仅仅通过短序列几乎不能得到很好的拼接效果，这是因为 data2, data3, data4 中 short reads 的复制倍数在逐渐减少。我们需要使用错误较多但是较长的 long reads 数据，将上一步得到的 “尽可能可靠的” 序列排列在上面。本工作中没有使用到 paired end 之间距离的信息，但是我相信长序列在一定程度上也可以以起到脚手架（scaffold）的作用。

我们通过对 “可靠序列” 与 “长序列” 进行排列，对长序列进行纠错。可以使用动态规划设计这一系列算法。然而，尽管长序列的长度固定（1000bp），可靠序列的长度的变化范围较大（50bp-1000bp）。可靠序列与长序列有若干种不同的重叠方式，可能是全部包含，或者在前缀包含。令 $dp[0...i][o...i]$ 为代价矩阵。定义状态转移方程为：

$$dp[i,j] = min \begin{cases} dp[i,j-1] + s(,x[j-1]) \\ dp[i-1,j] + s(x[i-1],) \\ dp[i-1,j-1] + s(x[i-1],x[j-1]) \end{cases} \tag{1}$$

其中 s 对两个字符的差异度进行打分。本次工作中简单认为不同的字符差异度为 1，相同字符差异度为 0. 空格与任何字符的差异度为 1. 尽管在现实中，不同的字符对间的差异度值应该不同，本次工作中不做更多假设。

我们可以通过观察较短序列末尾对应的 dp 矩阵的最小值得到关于两个序列重复度的信息。如果要测试两个序列是否存在后缀-前缀近似匹配的情况，可以通过限制 dp 矩阵的第一列的值总为 0 的方式实现。本节的算法总是在可靠序列与长序列之间进行。通过设定阈值，当识别出可靠序列与长序列存在包含关系时，通过可靠序列的片段对长序列对应片段进行替换的方式进行纠错。由于算法第一步的设计使得此时的输入序列足够可靠，这种方式是合理的（否则，我们可能需要通过多个 contigs 进行 consensus 来进行纠错）。当识别出

表 1
算法执行到不同步骤时的表现。作为示例 kmer 取 51。排行数据截至 2016-6-24 20:40

	Stage	Rank	Genome Frac.	Dupl. Ratio	NGA50	Misassem.	Mis. Per100
data1	Reliable Contig	24	89.23	1.9018	4050.2	2.0	0
	After OLC	24	89.23	1.9246	4050.2	2.0	0
	Ensemble	7	99.838	4.2454	9242.0	5.0	0
data2	Reliable Contig	21	91.528	1.766	381.8	0.0	0
	After OLC	17	97.884	2.5212	4563.2	0.0	0
	Ensemble	1	99.568	8.9286	9775.4	2.0	0
data3	Reliable Contig	21	72.036	1.4224	193.4	0.0	0
	After OLC	12	80.66	2.1852	4704.8	0.0	0
	Ensemble	5	95.934	5.715	7635.2	1.0	0
data4	Reliable Contig	17	69.5714	1.4772	289.6	1.0	0
	After OLC	13	81.1926	2.189	3190.6	7.0	0
	Ensemble	5	96.8118	4.0308	11382.2	34.0	0

后缀-前缀近似匹配时，将长序列与可靠序列进行拼接。最后，我们将得到一系列的纠正好的长序列与 contigs 片段。这部分算法是很具有启发式的。在本项目初期，还没有使用 OLC 算法时，算法到此步终止。事实证明这一步骤起到了很大的作用。然而，在加入 OLC 步骤后，该部分带来的提升不大。

3.3 Overlap-Layout-Consensus on Long Contigs

现在，我们得到了 contigs 与初步纠正后的长序列。这些序列大多数有着较长的长度，而且数量较少（data1 到 data3 仅几百条量级）。在算法开始时，经过分析后我们并没有直接使用 OLC 算法，原因是对它的性能较为担忧。然而，算法执行到下载我们已经获取列数量较少，但是长度较长的序列，这正是 OLC（0verlap-Layout-Consensus）算法所擅长的。我对长序列与可靠序列使用 OLC 算法拼接。由于之前已经进行了大量的探索，这里我决定直接使用开源的 Overlap Layout Consensus 代码（并不是商业软件）。简单来讲，算法首先构建一个重叠图数据结构，并对图进行简化、按序输出，最后调用 consensus 进行最后一步的纠错。

3.4 Ensemble: 多次运行累积长片段

不同的 kmer 大小对算法的结果对 reliable contigs 的长度以及 genome fraction 由较大的影响。当 kmer 大小较小时，可以取得较大的 genome fraction；当 kmer 大小较小时，由于不同基因 kmer 覆盖率减小，生成的无歧义的 contigs 更多，对 OLC 步产生的序列长度由直接影响。我们可以通过一定的策略将这些优势组合起来。这个策略是通用的，同一套参数适用于 data1 到 data4 的全部数据。

1) 设定 Kmer 大小为 17，运行算法，得到拼接的序列，得到集合 S。
2) 在上一步基础上，设定 Kmer 大小为 51-77 之内的奇数值，运行算法。将长度超过一定值[1]，或者长度超过当前为止的 S 内最长序列长度减去 1000 的序列加入 S。得到的 S 为最终的拼接结果。

1. data1,data4:9000bp; data2,data3:8000bp

这会得到非常显著的表现提升。在某些数据集可以得到几千量级的表现提升。

4 实验结果

表 1给出了算法运行到不同阶段得到的结果的评测指标。作为示例，kmer 取 51，但这对于 Reliable contig 和 OLC 步骤并不是最好参数（实际上，仅靠前两部也能得到比较好的结果）。

为了证明 Reliable Contigs 步骤是有用的，我在 OLC 步骤中将 reliable contigs 替换为 raw reads 原始数据。如果这样做，OLC 步骤后即使在比较简单的 data2 上也只能得到大约 400 左右的 NGA50. 这证明 Reliable Contigs 步骤是有用的。

此外，值得一提的是，项目初期使用的仅依靠 short reads 的使用 DFS 或寻找欧拉回路的方法在 data1 和 data2 也能得到比较好的效果，分别可以得到 7000,3000 左右的 NGA50。然而这些算法在 data3 中就显得很乏力了。

5 声明

1) 如前文所述，本项目在 OLC 部分使用了开源代码：https://github.com/yechengxi/DBG2OLC
2) 本项目参考了 [1] [2] [3] 等论文。

参考文献

[1] S. Koren, M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis, *et al.*, "Hybrid error correction and de novo assembly of single-molecule sequencing reads," *Nature biotechnology*, vol. 30, no. 7, p. 693, 2012.

[2] 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*, vol. 20, no. 2, pp. 265–272, 2010.

[3] C. Ye, C. M. Hill, S. Wu, J. Ruan, and Z. S. Ma, "Dbg2olc: efficient assembly of large genomes using long erroneous reads of the third generation sequencing technologies," *Scientific reports*, vol. 6, p. 31900, 2016.