**哈尔滨工业大学计算学部**

**读书/论文笔记**

课程名称：生物信息学

课程类型：选修

项目名称：基因组序列拼接 Velvet

班级：2103601

学号：2021112845

姓名：张智雄

|  |  |  |
| --- | --- | --- |
| **设计成绩** | **报告成绩** | **指导老师** |
|  |  | **刘博** |

1. **论文的主要研究问题描述**

这篇论文的主要研究问题是开发一种新的算法，用于处理高通量测序技术产生的非常短的DNA序列读段（25-50 bp）。这些短读段对于传统的基因组序列拼接方法来说是一个挑战，因为它们通常不足以跨越基因组中的重复区域，导致难以生成较长的连续序列（contigs）。为了解决这个问题，作者提出了一套名为“Velvet”的算法，专门针对de novo短读段拼接，使用de Bruijn图作为核心数据结构。

在传统的重叠布局一致性（OLC）方法中，每个读段被视为图中的一个节点，而检测到的重叠则作为节点间的弧。然而，由于短读段产生的大量节点和弧，使得OLC方法在处理短读段时变得非常低效，且容易在重复区域产生歧义。相比之下，de Bruijn图通过k-mers（k个核苷酸的序列）来表示数据，每个k-mer作为一个节点，节点通过有向弧相连，表示k-mers的顺序。这种方法在处理高冗余数据时更为高效，因为重复的序列在图中只出现一次，并且可以通过明确的链接来识别不同的起始和终止点。

Velvet算法包括两个主要的步骤：错误校正和重复解决。首先，错误校正算法通过合并序列来纠正测序过程中产生的误差。接着，重复解决算法通过分离共享局部重叠的路径来解决重复区域的问题。此外，作者还开发了一个名为“Breadcrumb”的模块，利用配对末端读段信息来解决重复区域，提高拼接的连贯性。

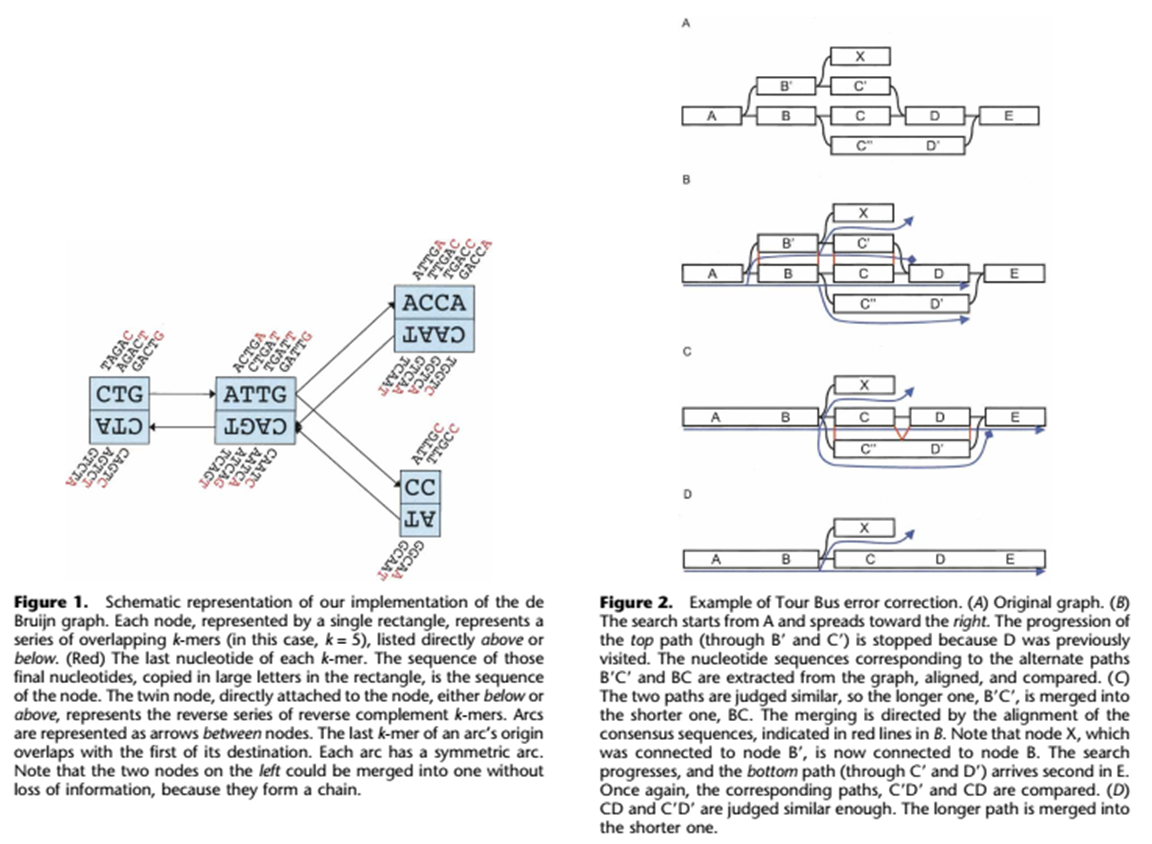
为了评估Velvet算法的性能，作者在模拟数据和真实数据上进行了测试。模拟数据包括大肠杆菌、酿酒酵母、秀丽隐杆线虫和人类基因组的5 Mb区域，使用35 bp长的读段进行测试。实验数据显示，Velvet能够在短读段数据上生成较长的contigs，并且在引入1%的错误率和模拟二倍体装配时，表现依然稳定。此外，作者还将Velvet与其他几种短读段拼接程序进行了比较，包括SSAKE和VCAKE，结果表明Velvet在内存使用、速度和生成的contigs大小方面具有优势。

最后，作者讨论了Velvet算法的计算复杂性和扩展性问题。指出图的构建是算法中最耗时和内存的部分，并且提出了可能的解决方案，如使用持久化数据结构来处理全基因组装配。作者认为，尽管基因组序列拼接问题尚未完全解决，但Velvet算法已经能够将高覆盖率的短读段转换成合理长度的contigs，并且通过结合配对读段信息，可以解决大部分重复区域，为基因组序列的de novo拼接提供了一个有效的工具。

1. **论文的主要方法**

**2.1 de Bruijn图的构建**

de Bruijn图的构建是Velvet算法中的关键步骤，它为后续的序列拼接提供了基础的数据结构。以下是de Bruijn图构建过程的详细描述：k-mer是长度为k的DNA序列片段。在构建de Bruijn图之前，首先需要确定k-mer的长度。这个长度受限于读段的长度，并允许有一定的重叠，以确保能够检测到读段之间的重叠区域。将所有的读段根据k-mer长度进行哈希处理。对于每个k-mer，记录它首次出现的读段的ID以及在该读段中的位置。同时，每个k-mer及其反转互补序列都会被记录，以确保图中能够表示正反两条链上的信息。在de Bruijn图中，每个节点代表一个k-mer序列。如果k是奇数，那么每个k-mer及其反转互补序列将被视为不同的节点。每个节点与其反转互补序列的节点配对，形成一个“块”。节点之间的连接（有向弧）表示k-mers之间的顺序关系。如果一个k-mer的最后一个k-1个核苷酸与另一个k-mer的第一个k-1个核苷酸相同，那么这两个节点之间就可以建立一个有向弧。通过哈希表中的信息，将每个读段重写为一系列原始k-mers的组合，以及它们与之前哈希的读段的重叠。这种新的读段表示方式被称为“路线图”。记录每个读段的原始k-mers被后续读段重叠的情况。每次读段与另一个读段开始或结束重叠时，该读段的原始k-mers序列就被切断。使用路线图，从图中的一个节点开始，沿着有向弧逐步构建路径，直到覆盖所有读段。每条路径都代表了一个可能的DNA序列。在图构建完成后，通常会简化它以减少内存使用和计算时间。简化的过程涉及合并只有单个出弧和入弧的节点，将链状的块合并成单个块。在进行错误校正之前，需要确保图中的每个节点都正确地表示了读段中的k-mers序列，并且所有的连接都反映了k-mers之间的正确顺序。通过上述步骤，de Bruijn图被构建完成，为下一步的错误校正和序列拼接打下了基础。Velvet算法利用这种图结构有效地处理了短读段数据，尤其是在处理重复序列和错误校正时，de Bruijn图的优势更为明显。



**2.2图的简化**

在de Bruijn图中，每个节点代表一个k-mer序列，而节点之间的连接（有向弧）表示k-mers之间的顺序关系。当一个读段结束时，它在图中形成一个“链”，这是一系列线性连接的块。这些链状结构增加了内存的使用量，并且延长了计算时间，因为每个块都需要单独处理。如果一个节点A只有一个出弧，且指向的节点B也只有一个入弧，那么这两个节点（以及它们的双胞胎节点）可以合并成单个节点，而不会丢失任何信息。这个过程是迭代进行的，图中的链状结构被逐步合并，直到无法进一步简化为止。在合并节点时，相关的信息（如读段ID、覆盖度和序列信息）被适当地转移到合并后的节点上。随着节点的合并，图中的路径也会相应地调整，以确保读段在图中的正确映射。简化后的图更加紧凑，减少了内存占用和处理时间，同时保留了足够的信息用于后续的错误校正和序列拼接步骤。

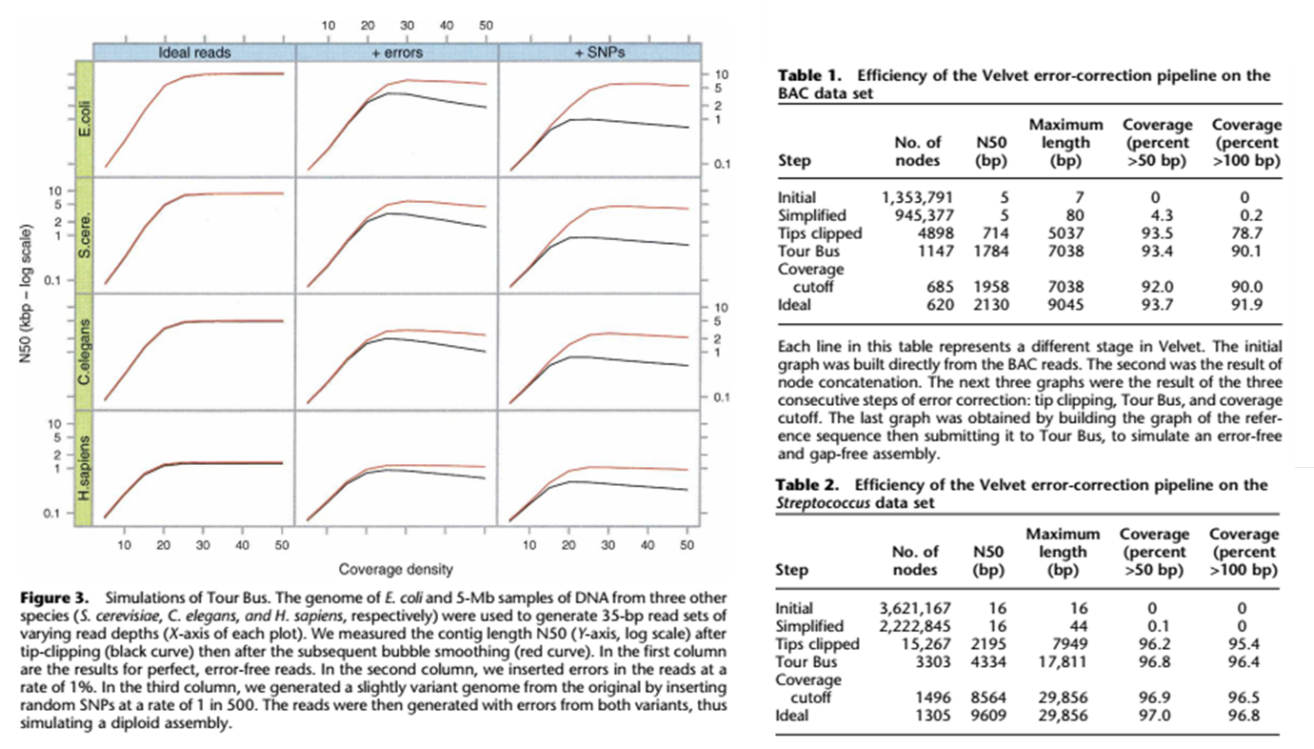
通过减少节点和弧的数量，简化操作减少了内存的使用，这对于处理大规模基因组数据集尤其重要。简化图减少了算法在后续步骤中需要处理的数据量，从而加快了处理速度。尽管进行了简化，但图中的序列信息和覆盖度信息仍然被保留，这对于维持拼接结果的准确性至关重要。在某些情况下，过度简化可能会导致丢失对拼接有用的信息，尤其是在基因组中存在复杂重复区域时。简化过程可能受到k-mer长度等参数的影响，需要仔细调整以避免损失关键信息。

**2.3 错误校正**

在测序过程中可能会产生不同类型的错误，包括替换、插入和缺失。Velvet算法主要关注替换错误，因为这是短读段测序技术中最常见的错误类型。在进行错误校正之前，首先使用选定的k-mer长度构建de Bruijn图。图中的每个节点代表一个k-mer，节点之间的连接（弧）表示k-mers的顺序关系。Velvet的错误校正包括三个主要步骤：去除尖端（tips）、去除气泡（bubbles）和处理错误连接：

* + **去除尖端：**尖端是图中未连接的末端链，可能是由于测序错误或真实的序列变异造成的。Velvet通过设置长度阈值（通常是2k）来识别和去除这些尖端，以避免移除真实的序列。
  + **去除气泡：**气泡是图中的简单循环，可能由测序错误或真实的序列多态性造成。Velvet使用“Tour Bus算法”来识别和合并相似的路径，从而去除气泡。
  + **处理错误连接：**错误连接是图中不形成循环的异常连接，可能是由测序错误或克隆错误造成的。Velvet通过设置覆盖度阈值来识别和移除这些错误连接。
  + **Tour Bus算法：**该算法通过广度优先搜索（BFS）遍历图，并在遇到重复节点时回溯，以找到两个相似路径的共同祖先。然后，算法会比较这两条路径的序列，并在它们足够相似时合并它们。
  + **覆盖度阈值：**在Tour Bus算法之后，Velvet使用用户定义的覆盖度阈值来移除剩余的低覆盖度节点，这些节点可能是由于错误造成的。

通过校正测序过程中产生的错误，Velvet能够提高最终拼接序列的准确性。Tour Bus算法能够有效处理图中的气泡结构，即使在存在序列多态性的情况下也能进行校正。错误校正步骤能够适应不同的测序错误率和基因组复杂性。错误校正的性能在很大程度上依赖于参数（如k-mer长度和覆盖度阈值）的选择，这可能需要用户的经验和优化。错误校正步骤可能难以区分真实的生物学变异（如SNPs）和测序错误，这可能导致一些真实的序列变异被错误地校正。错误校正步骤，特别是Tour Bus算法，可能需要较多的计算资源，尤其是在处理大规模基因组数据集时。



**2.4 Breadcrumb模块**

Breadcrumb模块的主要目的是利用配对末端（Paired-end）信息来解决de Bruijn图中由于重复序列造成的复杂结构，从而提高基因组序列的拼接质量。

Breadcrumb模块首先定义一个长度阈值，该阈值通常比大多数插入片段（inserts）的长度要长。所有长度超过这个阈值的节点被称为“长节点”（long nodes）。使用配对末端信息，Breadcrumb模块尝试将长节点两两配对。即使没有关于配对的唯一性信息，该步骤也尝试找到可以连接的节点对。对于每个长节点，Breadcrumb标记所有包含该节点配对末端读段的节点。如果存在一个对应的长节点，它会标记所有与该对应节点相连的节点。Breadcrumb尝试从每个被标记的节点出发，探索可能的路径，直到遇到没有标记的节点或存在多个可能的路径为止。如果一个长节点通过少于5个配对末端读段与多个长节点相连，这些连接被认为是不可靠的，并将被丢弃。Breadcrumb在探索路径时考虑到了错误的发生，它使用Tour Bus算法中的错误校正信息，忽略被认为是不可靠的读段。Breadcrumb能够通过配对末端信息将两个长节点之间的序列连接起来，形成所谓的序列连接的超连续序列（Sequence Connected Super Contigs，SCSCs）。这些SCSCs提供了比传统间隙超连续序列（gapped supercontigs）更多的信息。

Breadcrumb能够有效地处理基因组中的重复区域，提高了拼接的连贯性和准确性。通过使用配对末端读段，Breadcrumb能够跨越de Bruijn图中的复杂结构，连接分离的序列。模块设计中包含了对测序错误的容忍，使得即使在存在错误的情况下也能进行有效的拼接。Breadcrumb的性能可能受到插入片段长度分布和覆盖度等参数的影响，需要适当的调整。对于非常复杂或高度重复的区域，即使使用Breadcrumb模块，也可能难以完全解决拼接问题。处理大量的配对末端信息和探索可能的路径可能会消耗较多的计算资源。

1. **论文的主要实验结果**

**3.1 模拟数据测试**

作者使用模拟数据对Velvet算法进行了测试，选择了四种不同物种的基因组：大肠杆菌（E. coli）、酿酒酵母（S. cerevisiae）、秀丽隐杆线虫（C. elegans）和人类（H. sapiens）。这些测试包括了不同覆盖度（5x到50x）下，35 bp长度的读段，并且考虑了1%的突变错误率。实验结果表明，在没有错误的情况下，随着覆盖度的增加，N50值开始迅速增长，并在达到一定水平后趋于稳定，这个稳定值与基因组的复杂性和重复性有关。当读段中引入1%的错误率时，Velvet算法仍然能够有效地进行拼接，尽管最大N50值有所下降。这显示了Velvet在错误存在的情况下仍保持了较好的性能。在模拟二倍体装配的情况下，即使在参考基因组中随机加入SNPs，Velvet算法的性能也未受到显著影响，这表明算法对基因组的自然变异具有一定的容忍度。

**3.2 真实数据测试**

使用Solexa测序技术获得的真实人类BAC克隆数据进行了测试。该BAC克隆的长度为173,428 bp，平均覆盖度为970x。通过Velvet算法进行错误校正后，使用Tour Bus算法合并了相似的路径，有效地处理了重复序列。实验结果显示，与使用已知完成序列构建的理想de Bruijn图相比，Velvet生成的图在结构上保持了高度的一致性，中位节点长度和最大节点长度都与完成的BAC图相当。

图形用户界面, 应用程序

描述已自动生成

在Streptococcus suis P1/7的2 Mb基因组上进行的测试中，Velvet算法处理了270万个36 bp的读段，平均覆盖度为48x。测试结果没有产生错误的装配，基因组的96.5%被覆盖，且序列一致性达到了99.996%。

**3.3 覆盖度对N50的影响**

通过构建不同覆盖度的子集读段的de Bruijn图，研究了覆盖度对N50值的影响。实验结果表明，随着覆盖度的增加，N50值呈指数增长，但最终会趋于一个平台期，这个平台期可能与基因组的重复结构有关。

**3.4 Breadcrumb模块测试**

Breadcrumb模块是为了解决de Bruijn图中的重复区域而设计的。通过使用模拟数据集进行测试，作者评估了Breadcrumb模块的性能。实验结果显示，当插入序列（insert length）足够长以跨越图中的障碍时，N50值会增加。然而，如果插入序列过长，可能会导致scaffolding（支架）问题，从而降低装配的质量。Breadcrumb模块在处理重复区域时，通过配对末端信息来连接长节点，有效地提高了装配的连贯性。尽管如此，一些复杂或稀疏的重复结构仍然对装配构成了挑战。

1. **论文方法的优缺点分析**

Velvet算法特别设计用于处理新一代测序技术产生的非常短的读段（25-50 bp），适用于高覆盖率的数据集。通过使用de Bruijn图，Velvet能够以紧凑的形式表示基因组数据，这使得即使是大规模的数据集也能够被有效管理。Velvet算法包括一个错误校正步骤，可以识别并校正测序过程中产生的错误，提高了拼接的准确性。通过Breadcrumb模块，Velvet能够利用配对末端信息来解决基因组中的重复区域，增强了拼接的连贯性。Velvet在模拟数据和真实数据上进行了测试，证明了其方法的有效性和鲁棒性。Velvet的代码是开源的，并且可以在GNU公共许可下免费使用，这增加了其在研究社区中的可访问性。与其他短读段拼接程序相比，Velvet在内存使用、速度和生成的contigs大小方面表现出优势。

Velvet的性能对参数k（k-mer长度）非常敏感，需要用户进行适当的调整，这可能需要一定的实验和优化。尽管Velvet在处理短读段方面表现出色，但在处理大规模基因组数据时，图构建阶段可能会消耗大量内存和计算资源。于非常复杂或长串联重复的区域，即使是Breadcrumb模块也可能无法完全解决，这可能限制了算法在某些复杂基因组中的应用。错误校正步骤可能无法完全区分真实的生物学变异（如SNPs）和测序错误，这可能需要后续的后处理步骤来区分。随着基因组数据量的增加，Velvet算法需要进一步优化以处理更大的数据集，尤其是在内存和计算效率方面。在某些特定的应用案例中，如非常长的重复序列或结构变异丰富的基因组，Velvet可能需要与其他技术或算法结合使用以获得最佳结果。由于基因组学是一个快速发展的领域，新的测序技术和算法不断涌现，Velvet算法可能需要不断的更新和改进以保持其竞争力。