*xxxx*xxxxxx 在线优先出版日期：日 月 年 稿件类别

主题栏目

AMGC：基于自适应匹配的基因组压缩

Jia Wang 1、Yi Niu [1,2*]、Tianyi Xu [1], Mingming Ma 1、Dahua Gao [1] 和 Guangming Shi [1,2]

[1]西安电子科技大学人工智能学院，中国西安，710071，以及
[2] 鹏城实验室，中国深圳，518055。

*如需联系，请联系：

xxx

xxx

摘要

研究动机： 尽管第三代测序（TGS）技术取得了显著进展，下一代测序（NGS）技术在当前测序市场中仍占主导地位。这是因为与 TGS 相比，NGS 的错误率更低，分析软件也更丰富。NGS 技术会生成海量基因组数据，包括短读段、质量值和读段标识符。因此，高效压缩这类数据已成为迫切需求，并促使研究人员广泛开展 FASTQ 压缩器的设计研究。以往研究表明，质量值的无损压缩似乎已接近极限。但读段部分的压缩仍有很大提升空间。

研究结果： 通过研究测序过程的特点，我们提出了一种用于压缩 FASTQ 文件中读段的新算法，可集成到各种基因组压缩工具中。我们首先梳理了基于参考序列的算法流程，并确定了对存储结果影响显著的三个关键组成部分：读段在参考序列上的匹配位置（*refpos*）、读段中碱基的不匹配位置（*mispos*）以及匹配失败的读段（*unmapseq*）。为减小它们所占空间，我们详细分析了匹配位置和测序错误的分布，随后开发了 AMGC 的三个模块。实验结果表明，AMGC 优于当前最先进的方法，其压缩比平均提升了**81.23%**，相较于表现第二好的压缩器。

获取方式： https://github.com/wj-inf/AMGC联
系邮箱： niuyi@mail.xidian.edu.cn

1 引言

自下一代测序（NGS）技术出现以来（Mardis（2008年），我们见证了测序技术的快速发展。尽管第三代测序（TGS）技术已得到广泛发展（Schadt 等（2010年），但由于两个主要原因，NGS 技术在当前测序市场中仍占主导地位。首先，NGS 技术产生的测序数据错误率显著低于 TGS 技术。例如，在 HiSeq（NGS 技术）数据中，92% 的测序碱基的碱基质量分数 ⩾30，这意味着估计错误率低于 0.1%（Ma 等 (2019年）。TGS 长读段通常错误率较高：PacBio测序的错误率为 15%，Oxford Nanopore测序的错误率高达 40%（Wee 等 (2019年）。与 NGS 序列的组装相比，TGS 序列的高错误率使其组装看起来复杂得多、成本也高得多。NGS 数据也有更丰富的分析软件。SAM tools（Li 等 (2009年）、BWA-MEM 和 Bowtie2（Li(2013年)）已得到广泛应用，使 NGS 数据的下游分析更加便捷。因此，知名企业 Illumina 和

arXiv:2304.01031v1 [cs.IT] 3 Apr 2023

©

华大基因仍将 NGS 视为重要的测序业务，尤其是 Illumina 的 HiSeq测序仪系列。

下一代测序技术每天都能产生海量测序数据。例如，NovaSeq 6000 测序仪单次运行 44 小时即可生成高达 3 TB 的数据。然而，NGS 数据的增长给数据存储和传输带来了严峻挑战。如何有效存储和传输这些海量高通量基因组数据，已成为亟待解决的问题。基因组数据压缩技术是解决这一问题的重要途径。由于读段标识符具有明确的格式，NGS 数据压缩主要针对短读段和质量值。根据先前的研究（Niu 等人(2022年)），质量值的无损压缩似乎已接近极限。然而，读段部分的压缩仍有进一步提升的潜力。

在这项工作中，我们提出了一种用于读段压缩的算法 AMGC，其性能优于当前最先进的压缩器。AMGC 由三个关键模块组成，旨在解决当前基于参考序列的方法流程中的三种编码冗余。第一个模块专注于参考匹配位置（参考位置）的编码。通过进一步分析参考位置的分布，我们发现相邻读段更可能匹配到参考序列上彼此接近的位置，因此提出了基于位平面的差分参考位置编码模块。随后，我们研究了下一代测序过程中涉及的合成反应，发现短读段尾部更容易出现测序错误，导致测序错误呈均匀分布。为此，我们提出了自适应二进制错配位置编码模块。该模块使用二进制向量表示匹配错误，然后利用高阶上下文对该向量进行编码。最后，由于前子段的质量优于后子段，尾部连续的错配碱基可能导致整个读段匹配失败。因此，我们进一步提出了递归拆分匹配模块。匹配失败后，读段将被拆分为两个片段，并对这两个片段重新进行比对。这样，质量较高的前子段便可能成功匹配。

我们在从以下机构提供的标准测试数据集中选取的数据集上评估了 AMGC：AVS-G。实验结果表明，与性能第二好且随机存取存储器和时间开销相当的压缩器相比，AMGC 的平均增益达到 81.23%。

2 相关工作

下一代测序技术产生了大量冗余度高、易于压缩的短基因组读段。然而，gzip、bzip2 和 7z 等常用通用压缩工具无法在基因组数据上实现令人满意的性能，因为它们未能充分利用数据的生物学特性，例如重复片段和回文序列。因此，亟需专门针对基因组数据设计高效的压缩方法，这也促成了大量重要研究。

针对原始 FASTQ 格式，已有多种专用压缩方法被提出（Bonfield 和 Mahoney(2013年)）（Deorowicz 和 Grabowski(2011年)）。根据是否需要额外的参考基因组，这些压缩技术大致可分为三类：高阶上下文压缩算法、基于组装的压缩算法和基于参考序列的压缩算法。高阶上下文压缩算法正逐渐被淘汰。

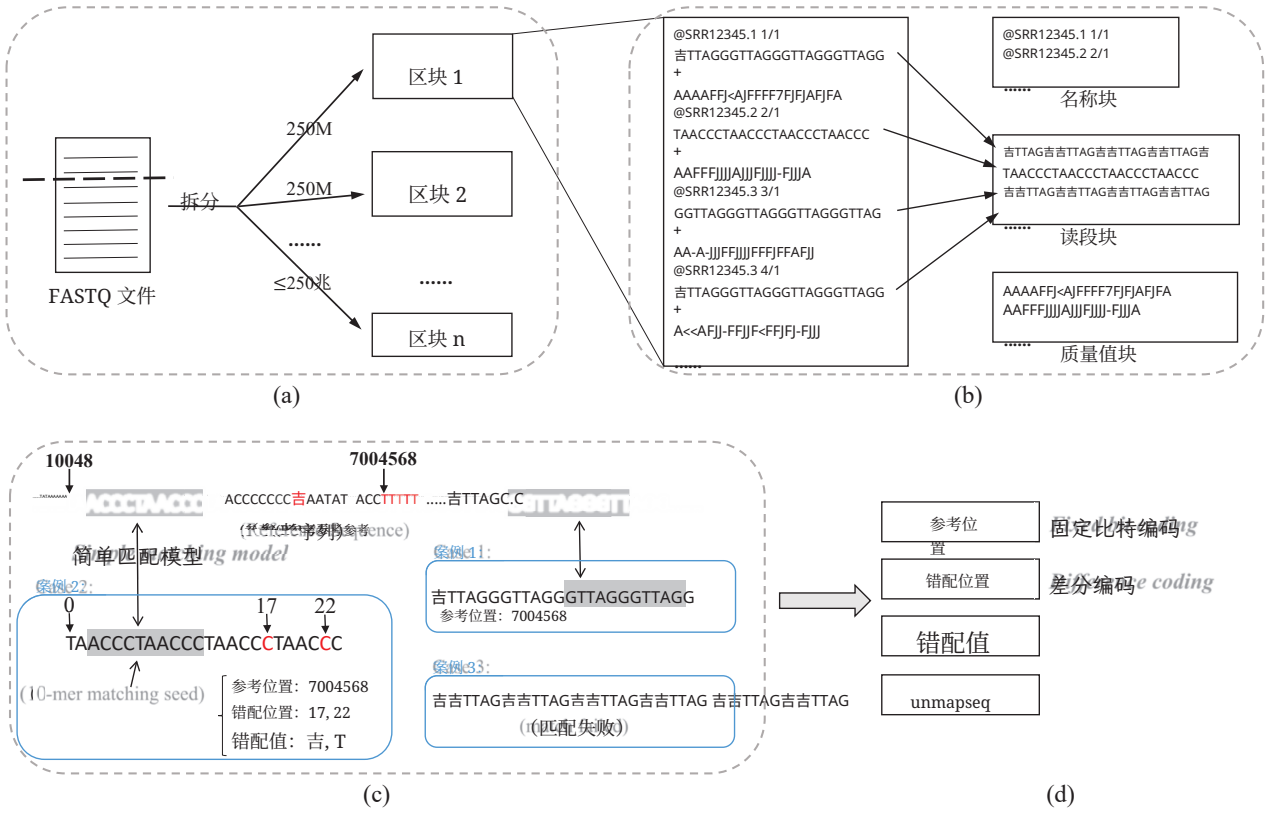
基于组装的方法利用 FASTQ 文件中读段之间的相似性，并通过组装算法将测得的短序列拼接成一条长序列。由于不需要参考基因组，这些方法可用于未测序物种和宏基因组学。Quip（Jones等(2012年)）使用布隆过滤器构建 de Bruijn 图进行拼接。拼接得到的重叠群序列被用作原始 FASTQ 文件中已测序读段的参考基因组。最后，仅记录每条已测序读段的比对位置和变异信息以进行编码。Leon（Benoit 等(2015年)）使用所有已测序读段构建 de Bruijn 图。与 Quip 方法类似，Leon 也使用布隆过滤器滤除低丰度 k-mer 节点，以减少分叉节点的数量。每条已测序读段表示为起始节点路径加上分叉节点信息。随后，分别采用树结构和零阶上下文模型对其进行编码。HARC（Chandak等(2018年)）大致按照读段在基因组中的位置重新排序。对重新排序后的读段进行编码，以消除相邻读段之间的冗余，并将所得参数存储在不同文件中。上述文件使用基于 Lempel-Ziv（Ziv 和 Lempel(1977年)）和 BWT（Burrows 和 Wheeler(1994年)）的通用压缩器进行压缩。PgRC（Kowalski 和 Grabowski (2020年)）构建伪基因组，其近似于读段的最短公共超串。然后，根据读段在伪基因组上的比对位置对其进行编码。

基于参考序列的方法会将测序读段比对到外部参考序列，以识别它们之间的相似性。编码的不是原始目标序列，而仅是比对位置和变异信息。通过利用测序基因组与参考基因组之间的相似性，这些方法能够实现更好的性能。以人类基因组为例，超过 99% 的碱基对都能与参考基因组匹配（Lander 等人（2001年））。在 LW-FQZip（Zhang 等人（2015年））中，读段先由轻量级比对模型进行预处理，随后使用 LZMA 等通用工具压缩映射结果。CRAM（Fritz等人（2011年））等算法采用标准的基于参考序列的方案来压缩 SAM/BAM 数据。该方案依赖经过高度优化的 SAM 字段，并支持多种统计模型。使用这些模型可以更高效地从参考序列中提取相关信息，从而降低压缩率。GTZ（Xing 等人（2017年））是工业应用中广受青睐的基于参考序列的算法。在压缩效果方面，它显著优于此前提出的其他基于参考序列的算法。

然而，目前所有算法都存在三个缺点。一个广泛采用的假设是参考位置在参考序列中均匀分布，因此使用固定比特编码压缩参考位置。另一个广泛采用的假设是错配位置（错配位置）也均匀分布，因此采用差分编码错配位置。最后，这些算法没有考虑连续错配的影响，而是设计了一个简单的匹配模型。该模型为整个读段的错配数设定阈值。然而，进一步的数据分析表明，参考位置和错配位置的分布都并不均匀。这意味着此前使用的三个模型都有改进空间。基于这些发现，AMGC设计了三个模块，分别针对这三项缺点进行改进。

3 洞察

在详细介绍所提出的 AMGC 算法的三个模块之前，我们先揭示 NGS 测序过程的两个重要特性。这有助于读者理解这三个关键模块的技术动机。在第 3.1 节中，我们首先以整数映射 k-mer 索引方法（可通过 http://www.ysunlab.org/kic.jsp 获取）为例，回顾基于参考序列的算法流程。然后展示每个匹配结果对最终压缩后大小的影响。在第 3.2 节和第 3.3 节中，我们简要回顾 NGS 测序过程及合成反应，随后详细分析参考位置 和错配位置的分布。



3.1 基于参考序列的方法的流程

符号说明：令“*refpos*”表示每条读段在参考序列上的匹配位置。令“*mispos*”表示读段上错配碱基的位置。令“*misvalue*”表示参考序列上与每个错配位置对应的正确碱基。令“*unmapseq*”表示未能匹配的读段集合。

首先，将 FASTQ 文件拆分为多个 250MB 的内存块（如图 1(a) 所示）。这是必要的，因为 FASTQ 文件通常过大，无法一次性载入随机存取存储器。拆分后也更便于进行多线程处理。如图 1(b) 所示，每个 250MB 的块会进一步划分为“名称块”“读段块”和“质量值块”。每个 250MB 的块各不相同，因此需要针对不同块中的读段采用自适应编码方案。

我们关注的是“读段块”。如图 1(c) 所示，使用 10-mer整数模型将每条读段逐一与参考序列进行比对。一些读段能够完全匹配（案例 1），我们只需记录其 *refpos*。另一些读段也能成功匹配，但存在少量不同的碱基（案例 2），因此需要记录其 *refpos*、*mispos*和 *misvalue*。对于其余读段（案例 3），它们被归入 *unmapseq*部分；由于这些读段与参考序列中的所有子串之间的间隔都很大，匹配因此失败。具体来说，通过 k-mer 确定参考位置后，算法会向两端扩展，并比较各对应位置上的碱基。如果不同碱基的数量超过某个阈值，就会判定本次比对失败。如果多个 k-mer 映射均失败，则判定该读段匹配失败。匹配过程结束后，读段块会被分为图 1(d) 中的四个部分，接下来只需进行压缩。

图 2 展示了匹配后读段块中各部分所占的比例。如图所示，“参考位置”、“*unmapseq*”和“错配位置”各部分

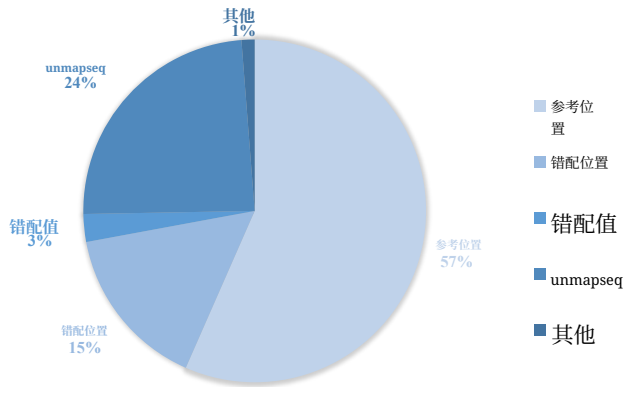


图 2. 匹配后读段块中各部分所占的比例。“其他”类别与合并碱基的处理有关。所用数据来自 AVS-G 处理的 SRR6691666_1.fastq 文件中的前 50 兆数据。

是决定压缩数据大小的关键因素。为此，AMGC 开发了三个模块以减小压缩后的数据大小。基于位平面的差分参考位置 编码模块显著缩小了“参考位置”部分。递归拆分匹配模块增加了成功匹配的读段数量，从而间接缩小“*unmapseq*”部分。此外，自适应二进制错配位置编码模块为“错配位置”提供了一种新的编码方法。

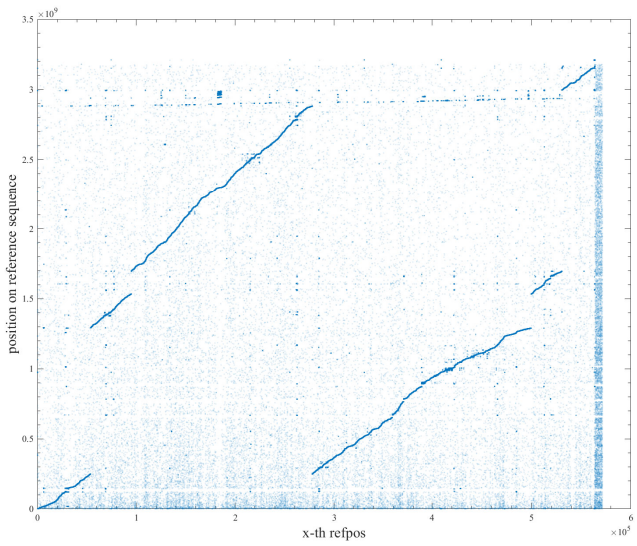


图 3. FASTQ 文件中所有读段的参考位置散点图。横轴表示各个读段，纵轴表示它们在参考文件 hg38.fa 中的位置。

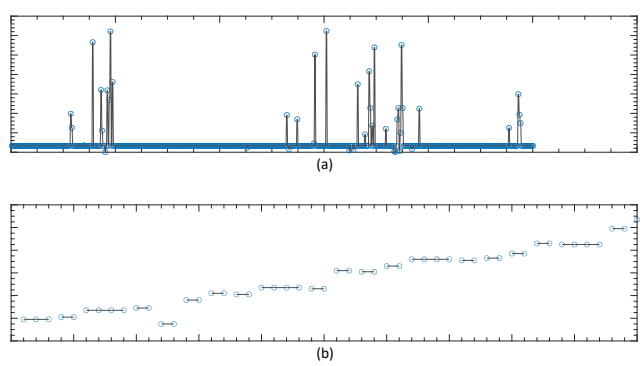


图 4.FASTQ 文件中局部读段的参考位置散点图

3.2 参考位置分布

例如，在 NGS 中，提取的 DNA 样本需要在样本制备阶段被切割成小片段，随后通过聚合酶链式反应扩增样本，并将其聚集成簇，然后开始合成测序阶段。与荧光标记脱氧核糖核苷酸配对的另一条链会同时发出不同颜色的光信号。测序仪捕获这些光信号，根据不同的光信号区分不同的碱基，并将测序结果写入 FASTQ 文件等文本文件中，从而完成测序过程。关键在于，随着相邻的小 DNA 片段被测序，FASTQ 文件中位置相近的读段通常会匹配到位置相近的区域，从而使比对位置信息呈现较强的局部相关性。此外，由于测序是按顺序进行的，FASTQ 文件中读段的参考位置 在局部也呈线性增长。

如图 3所示，*SRR6178157* 中所有读段的参考位置散点图如图所示。可以观察到，参考位置是一种具有局部增长趋势并伴有一些噪声的一维信号。因此，我们可以利用这一增长特性来开发参考位置编码模块。

为了更清楚地观察，我们放大了图 3。如图4所示，每个蓝色散点代表一个参考位置，黑线用于辅助观察

。图 4(a) 展示了连续 1000 个参考位置，可以观察到信号呈线性增长，同时在一定距离处出现一些噪声。图*4*中的连续 50 个参考位置 4(b) 中，黑线段表示连续出现的相同参考位置。连续相同的参考位置可能由聚合酶链式反应过程导致。这些参考位置信号的分布特征构成了 AMGC 位平面差分参考位置编码模块的设计依据。

3.3 错配位置的分布

测序错误可能会导致生成的读段与参考碱基进行比较时出现偏差。为了设计更有效的
压缩算法，分析错误来源至关重要。在合成和测序过程的早期阶段，反应可能不稳定，但 DNA 酶的质量良好。然而，随着测序的进行，反应逐渐稳定，而酶活性和特异性则逐步下降，导致累积错误随时间增加。这一趋势使碱基测序的错误率先略微下降，待稳定后再上升。关于这一现象的进一步分析，参见（Minoche 等人(2011年)）。

图 5. 散点图中读段错配位置的分布。横轴表示各个位置，纵轴表示错误数量。两条不同的曲线“正常”和“反向”分别表示顺序比较和反向比较。我们还显示了“反向 2”，即“反向”的反转。

图 5 以散点图形式展示了 *143003* 条读段在 *FASTQ* 文件“*SRR6178157.fastq*”中的 错配位置分布。结果清晰地呈现了上述分析所得的趋势。为应对这一特征，我们设计了自适应二进制错配位置编码模块。我们还仔细考量了这些错配碱基对匹配过程的影响，并提出了递归拆分匹配模块。

4 方法

在阐述这三个模块的技术动机后，本节将深入介绍其实现细节。第 4.1 节介绍 refpos 处理的完整流程，包括使用去重、差分和中值滤波来拟合 refpos 的观测数据特征。第 4.2 节介绍自适应二进制错配位置编码模块，该模块使用二进制向量表示匹配错误，再利用高阶上下文对该向量进行编码。第 4.3 节介绍递归拆分匹配模块，以便在未映射读段匹配中实现高质量子片段匹配。

4.1 基于位平面的差分参考位置 编码

现有方法未考虑参考位置的分布。我们采用差分操作来处理参考位置的显著局部相关性。差分过程中使用中值滤波，以考虑

噪声点。中值滤波差分公式为:

$$d(x) = p(x) \; - \; mid\{p(x-1), p(x-2), p(x-3)\}$$

其中， $p(x)$ 表示第 x 个参考位置， $d(x)$ 表示差分后的第 x 个参考位置，“mid”表示三个数的中位数，$p(0)$、 $p(-1)$ 和 $p(-2)$ 均为零。我们首先消除参考位置 中的连续重复项，以便让相同的参考位置 连续显示。

这项技术的输入是匹配后的“参考位置”部分。作为示例，我们使用人类基因组和 hg38.fa，其中每个参考位置占 32 位。该技术的步骤如下:

- 步骤 1：移除参考位置中的连续重复项。我们的分析发现，一些连续的参考位置 有许多相同的情况。这可能是因为两个相同的读段被合并，因此在对参考位置进行初始处理时移除了这些重复项。我们将重复项的描述记录在“PosEqual”文件中，其中 1 表示与前一个参考位置相同，0 表示不同。我们使用二进制算术编码器对“PosEqual”进行编码。

- 步骤 2：对处理后的参考位置进行中值滤波差分。在第 3 节中，我们观察到，由于 DNA 链顺序测序，参考位置呈线性增长。为了处理这个一维信号，我们采用差分处理。由于存在噪声，差分结果受到干扰，因此我们使用当前参考位置之前三个参考位置 的中值作为减数，以滤除噪声。

以 32位编码的<style id='1'>参考位置增量

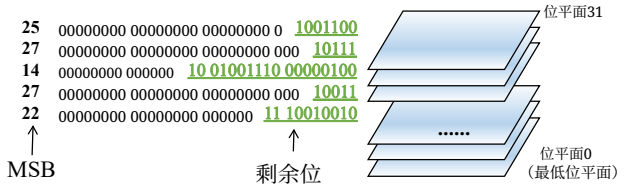


图 6. 参考位置差分的编码算法。右图展示了什么是位平面。左图展示了如何将参考位置差分拆分为“最高有效位”（MSB）和“剩余位”（RestBit）

- 步骤 3：将差分结果编码为位平面。我们将差分结果表示为正负符号和 32 位平面，并将其拆分为“最高有效位”（MSB）和“剩余位”（RestBit）两部分（图 6）。符号部分将输入简单的算术编码模型。对于“最高有效位”部分，我们使用 32值算术编码器直接编码。对于“左侧位”部分，编码时将 MSB 和每个位所在的位平面用作上下文。

4.2 自适应二进制错配位置 编码

考虑到读段上的错配位置 分布并不均匀，我们采用高阶上下文编码来捕捉错配趋势的相关性。该技术的输入是“

错配位置”匹配后的部分。我们以 151-mer 读段为例。

- 步骤 1：使用二进制向量表示匹配错误。根据 Insight 节中的分析，读段中每个碱基的错误概率并不均匀，尾部的错误概率更高。因此，我们将各个位置分开，然后用二进制信号对错误位置进行编码，如图 7所示。

- 步骤 2：使用高阶上下文编码。为了捕捉错配趋势的相关性，AMGC 不会仅以位置作为上下文来编码每个位，而是

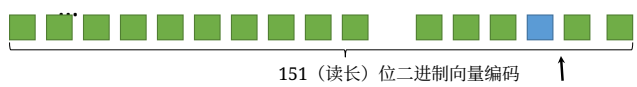


图 7. 错配位置的二进制信号

将当前位置前 10 个碱基中的错误数作为上下文。对当前编码位进行概率建模的方式:

$$p\left(x_i \middle| \sum_{k=i-10}^{i-1} x_k\right)$$

4.3 递归分割匹配

该技术用于匹配过程中。以整数映射的 k-mer 索引方法为例，比对工具会通过读段上的短 k-mer 将读段与参考序列进行匹配，然后将 k-mer 扩展至整个读段。如果错配碱基数超过阈值，扩展就会停止，匹配也会失败。然而，正如第 3.3 节所分析的，测序错误更容易发生在读段末端。这意味着读段前端的高质量片段也会被归入*unmapseq* 部分，造成参考序列的浪费。

为解决这一问题，我们的递归分割匹配技术会将匹配失败的读段切分成两个等长片段，再将它们重新交给比对工具，直到片段短到无法继续切分。该切分过程会递归进行，直到子片段短于阈值。

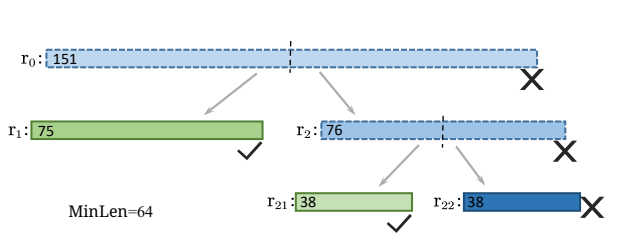


图 8. 输入读段及其子片段的匹配示例。每个读段都只会从中间切分成两个等长部分。“right”或“wrong”分别表示匹配成功或失败。“MinLen”是停止递归分割的阈值。

如图 8 所示，输入读段 (r0) 的长度为 151。匹配失败后，我们将其拆分为两个等长部分（r1 长度为 75，r2 长度为 76）。由于测序错误更可能出现在读段末端，r1 匹配成功，而 r2 匹配失败。随后，我们将 r2 分为 r21（长度为 38）和 r22（长度为 38）。r21 匹配成功，r22 匹配失败，但 r22 太短，无法继续拆分。r0 的整体匹配到此结束；通过更充分地利用参考序列，我们成功减少了未比对碱基数和 *unmapseq* 的大小。

5 结果与讨论

我们使用 AVS-G 对所提出的算法 AMGC 进行评估，因为它是我们能够获取的软件；随后，我们在多个真实数据集上对其进行了测试。下文中，“AMGC”既指该算法，也指部署了我们算法的压缩器 AVS-G。我们将 AMGC 的性能与多种现有算法进行了比较，并针对 GTZ、Leon、HARC、PgRC 和 gzip 测试了其性能。GTZ 是一种广泛使用的

表1. 用于压缩的数据集

数据集	原始大小	RUN	物种名称	技术	平台	单端/双端	读长
SRR4017489	16.5吉字节	SRR4017489	智人	WCS	Illumina HiSeq 2000	PE	101,101
SRR6691666	316.1吉字节	SRR6691666	智人	WGS	TruSeq + Illumina HiSeq X	PE	151,151
ERR753370	12.7 吉字节	ERR753370	智人	完成 Illumina HiScanSQ 测序		PE	101,101
SRR1238539	74.2 吉字节	SRR1238539	智人	WGS	Ion Torrent Proton	SE	177
SRR6178157	37.0 GB	SRR6178157	智人	WXS	Ion Torrent Proton	SE	135
SRR3479107	5.9 吉字节	SRR3479107	小鼠	MBD 测序	AB 5500xl GeneticAnalyzer 双端测序+单端测序 30,30		
SRR3479107	21.9 吉字节						
SRR5572323	29.8 吉字节	SRR5572323	小鼠	FAIRE 测序	Illumina HiSeq 2000	SE	76
SRR6240776	8.5 吉字节	SRR6240776	拟南芥 ATAC测序		Illumina HiSeq 4000	双端测序+单端测序	50, 50
SRR6240776	14.9 吉字节						

注： _1 表示仅使用了两个双端FASTQ文件中的一个。单端测序和双端测序的运行共享部分信息。

表2. 压缩器的性能

数据集	原始大小	长度	压缩比							Gain
			GTZ	LEON	HARC	PgRC	gzip		AVS-G	AMGC	
SRR4017489 _1	16.5吉字节	101,101	81.82	23.40	30.08	77.47	19.30	51.64	147.82	80.66%
SRR6691666 _1	316.1吉字节	151,151	73.51	24.70	60.79	81.05	16.81	53.56	98.78	21.88%
ERR753370 _1	12.7吉字节	101,101	73.89	22.22	-	-	15.37	53.45	121.20	64.03%
SRR1238539	74.2吉字节	∼177	19.53	8.83	-	-	11.78	10.78	21.82	11.73%
SRR6178157	37.0吉字节	∼135	17.62	11.56	-	-	11.21	14.29	19.47	10.50%
SRR3479107	5.9吉字节	30,30	15.28	13.63	13.87	18.42	7.51	25.64	77.96	204.06%
SRR3479107 _1	21.9吉字节	30	14.60	13.69	15.64	24.80	8.42	25.26	72.81	188.24%
SRR5572323	29.8吉字节	76	45.92	11.46	14.90	24.83	9.93	34.66	68.14	48.39%
SRR6240776	8.5吉字节	∼50	17.36	16.50	-	-	6.07	19.08	21.56	13.00%
SRR6240776 _1	14.9吉字节	∼50,∼50	29.55	29.07	-	-	15.60	43.67	117.82	169.80%
相较于表现第二好的压缩器的平均增益						rs:				81.23%

注： 对于读长不同的数据，HARC 和 PgRC 压缩失败。压缩比通过总大小/压缩后大小计算。鉴于标识符所占比例极小，实际读段压缩比约为所给压缩比的一半。

压缩器。它的参考压缩模式具有很高的压缩比和压缩效率。我们仅测试 GTZ 的参考压缩模式，不打包所用的参考部分（-r -n）。Leon、HARC 和 PgRC 是三种具有代表性的基于组装的方法，它们采用不同的方式拼接参考基因。gzip 是经典的通用压缩器。测试在一台配备第11代 Intel Core i7-11700F 2.50 GHz 处理器和 128 GB 随机存取存储器的机器上进行。

5.1 数据集

用于评估的数据集选自 AVS-G 提供的标准测试数据集，列于表 1。这些数据集包括五个人类、两个拟南芥和三个小鼠数据集，并包含原始大小、读长、所用测序平台及测序方法等信息。

我们比较的主要对象是 GTZ、AVS-G 和 AMGC。由于我们的算法只专注于压缩读段部分，因此在测试前对 FASTQ 文件进行了预处理。我们简化了标识符部分，并将质量值设为相同的值，从而使测试结果聚焦于读段部分。由于 GTZ 中不同部分的处理是集成的，因此我们在压缩输出中包含标识符和质量值部分，但这不会影响 GTZ、AVS-G 和 AMGC 之间的比较。需要注意的是，实验的设计方式有利于基于组装的算法。因为格式化后的标识符和质量值仍会对 GTZ、AVS-G 和 AMGC 产生轻微影响，而基于组装的算法会将它们舍弃。

5.2 压缩大小和压缩比

表 2 展示了不同压缩工具的压缩比。我们用红色标出表现最佳的压缩器，用蓝色标出表现次佳的压缩器。从表中可以看出，GTZ 和 AVS-G 在不同数据上各有优势，目前是表现最好的两种压缩器。PgRC 是基于组装的压缩器中表现最佳的。在压缩数据 SRR 6691666_1时，它取得了次佳表现。不过，AMGC 的表现胜过所有其他压缩器，平均压缩比比表现次佳的压缩器高 81.23%。

5.3 时间和内存使用情况

表3 和表4 展示了压缩和解压缩所需的时间和内存使用情况。所有压缩器均使用十六个线程运行。对于压缩整个 FASTQ 文件的压缩器，质量值对压缩比的影响较小，但 FASTQ 文件预处理仍会占用相应的内存。

基于参考序列的压缩器（如 AVS-G 和 GTZ）在压缩和解压缩时表现出更短的耗时。基于组装的

表3. 压缩器的时间开销

数据集	原始大小	编码时间							解码时间						
		GTZ	LEON	HARC	PgRC	gzip	AVS-G	AMGC	GTZ	LEON	HARC	PgRC	gzip	AVS-G	AMGC
SRR4017489	16.5 吉字节	2m7s	3分47秒	5分56秒	5分46秒	4m8s	3m5s	3分19秒	23s	1分20秒	41s	17s	39s	1分18秒	1分22秒
SRR6691666	316.1 吉字节	21分18秒	126分9秒	149分9秒	294分5秒	65分51秒	36分49秒	41分33秒	21分33秒	45分23秒	32分6秒	13分2秒	35分37秒	32分52秒	32分59秒
ERR753370	12.7 吉字节	2分34秒	3分20秒	-	-	3分42秒	1分40秒	1分47秒	2m5s	1m1s	-	-	31s	16s	17s
SRR1238539	74.2 吉字节	9分31秒	41分24秒	-	-	20分10秒	4分55秒	17分47秒	13分6秒	12分53秒	-	-	4分18秒	5分33秒	5分31秒
SRR6178157	37.0 GB	5分39秒	11分58秒	-	-	9分42秒	4m0s	7m5s	6分30秒	4m3s	-	-	1分40秒	1分22秒	1分25秒
SRR3479107	5.9 吉字节	56s	53s	13分7秒	3分44秒	3分19秒	1分55秒	1分15秒	11s	35s	22s	19s	17s	10s	14s
SRR3479107	21.9 吉字节	2m0s	2分47秒	78分15秒	11分30秒	7分57秒	1分50秒	2m2s	42s	3分20秒	42s	57s	1m2s	34s	44s
SRR5572323	29.8 吉字节	3分24秒	8分33秒	16分47秒	21分22秒	10分45秒	3分22秒	3m8s	1m4s	3分28秒	2分15秒	1分19秒	1分21秒	1分52秒	1分54秒
SRR6240776	8.5 吉字节	1分10秒	1分55秒	-	-	5分44秒	47s	33s	1分20秒	59s	-	-	26s	14s	16s
SRR6240776	14.9 吉字节	1分24秒	2分26秒	-	-	3分14秒	1分31秒	1分10秒	1分53秒	1分44秒	-	-	36s	16s	19s

注： 对于读长不同的数据，HARC 和 PgRC 压缩失败。

表4. 压缩器的内存使用量

数据集	原始大小	编码内存						解码内存					
		GTZ	LEON	HARC	PgRC	gzip	AVS-G	AMGC	GTZ	LEON	HARC	PgRC	gzip	AVS-G	AMGC
SRR4017489	16.5 吉字节	4.0G	4.8G	3.3G	3.4吉	1.7兆	9.5G	23.4吉	9.0G	1.4G	0.4G	2.2吉	1.4兆	5.4G	8.4G
SRR6691666	316.1 吉字节	6.4G	7.9G	47.8吉	72.3吉	1.6兆	13.9吉	34.7吉	9.6G	13.2吉	1.9吉	24.7吉	1.5兆	5.5G	23.9吉
ERR753370	12.7 吉字节	5.7G	0.9G	-	-	1.6兆	8.9G	21.1吉	17.4吉	1.4G	-	-	1.4兆	4.0G	9.4G
SRR1238539	74.2 吉字节	9.3G	6.1G	-	-	1.8兆	11.7吉	23.0吉	19.4吉	8.0G	-	-	1.4兆	3.7G	11.8吉
SRR6178157	37.0 GB	8.9G	2.5G	-	-	1.7兆	8.8G	22.6吉	18.6吉	3.5G	-	-	1.4兆	3.9G	12.0吉
SRR3479107	5.9 吉字节	3.3G	0.5G	0.9G	3.3吉	1.6兆	17.0吉	18.2吉	7.1G	0.6G	0.3G	1.7吉	1.5兆	3.2G	9.4G
SRR3479107	21.9 吉字节	4.4G	1.7G	6.5吉	11.6吉	1.8兆	9.5G	17.4吉	9.4G	6.1G	0.5G	4.3吉	1.5兆	4.4G	13.0吉
SRR5572323	29.8 吉字节	6.9G	2.2G	9.2吉	10.0吉	1.7兆	12.4吉	29.9吉	8.6G	2.7G	0.9G	4.5吉	1.4兆	5.2G	16.3吉
SRR6240776	8.5 吉字节	7.6G	0.5G	-	-	1.6兆	6.6G	12.0吉	12.4吉	1.2G	-	-	1.3兆	2.7G	8.7G
SRR6240776	14.9 吉字节	6.1G	0.7G	-	-	1.7兆	3.1G	14.4吉	16.1吉	0.4G	-	-	1.4M	3.3G	10.8G

注释： 对于读长不同的数据，HARC 和 PgRC 压缩失败。

压缩器在压缩时需要更多时间将所有输入读段组装成一个序列，因此耗时更长。与 AVS-G 相比，AMGC 的时间复杂度相似，这意味着我们的算法没有显著增加所需时间。

一个特殊示例是来自 Ion Torrent Proton 平台的 SRR1238539，其中 AMGC 的压缩耗时比 AVS-G 更长。这是因为该平台的读段可能比普通 NGS读段更长。对于较长的读段，AMGC 的递归拆分匹配模块需要进行更多次比对，因此编码时间更长。不过，我们的算法在其他数据集上表现稳定。与 GTZ 相比，AMGC 的压缩性能相近，但解压缩性能更好。

在内存消耗方面，GTZ 和 Leon 的效率较高，而 gzip 则具有显著优势。处理大规模数据时，HARC 和 PgRC 会占用大量内存。AMGC 的总体内存使用量较高。不过，随着内存技术的发展，更大容量的内存已变得普遍可用，因此内存用量不再是衡量基因组压缩算法优劣的重要因素，压缩比才是。

5.4 讨论

尽管基于组装的压缩算法不需要外部参考序列，但在组装得到参考序列后，仍需进行读段匹配。因此，我们的算法适用于这两类方法。

总体而言，与内存和时间开销相近的次优压缩器相比，AMGC 的压缩增益达到 81.23%。这是无损压缩领域的一项重要成果。

6 结论与未来工作

我们介绍了 AMGC，这是一种包含三个模块、基于读段匹配来提升读段压缩率的新算法。该算法可集成到各种基因组压缩工具中。AMGC 的表现优于当前最先进的方法，与表现次优的压缩器相比，增益达到 81.23%。

未来的工作可通过开发更合适的 k-mer 映射工具，并将空间组学信息纳入映射结果的编码中，将 AMGC 扩展至空间组学基因组数据的压缩。

资金支持

本研究获得了国家自然科学基金（NSFC 61875157）和华大基因深圳开放课题的资助，地址为中国深圳 518000。

参考文献

Benoit, G. 等(2015). 基于概率 de Bruijn 图的高通量测序数据无参考压缩。《BMC生物信息学》，**16**(1), 1–14. Bonfield, J. K. and Mahoney, M. V. (2013). FASTQ 和 SAM 格式测序数据的压缩。《PLOS ONE》，**8**(3), e59190. Burrows, M. and Wheeler, D. (1994). 一种块排序无损数据压缩算法。载于《Digital SRC 研究报告》。CiteSeer. Chandak, S. 等(2018). 通过基于哈希的重排序压缩基因组测序读段：算法与分析。《生物信息学》，**34**(4), 558–567.

Deorowicz, S. and Grabowski, S. (2011). FASTQ 格式基因组序列的压缩。《生物信息学》，**27**(6), 860–862. Fritz, M. H.-Y. 等(2011). 使用基于参考序列的压缩高效存储高通量 DNA 测序数据。《基因组研究》，**21**(5), 734–740. Jones, D. C.等(2012). 借助高效的从头组装压缩新一代测序读段。《核酸研究》，**40**(22), e171–e171. Kowalski, T. M. and Grabowski, S. (2020). PgRC：基于伪基因组的读段压缩器。《生物信息学》，**36**(7), 2082–2089. Lander, E. 等(2001). 人类基因组的初步测序与分析。《自然》，**409**. Li, H. (2013). 使用 BWA-MEM 比对测序读段、克隆序列和组装重叠群。arXiv 预印本 arXiv:1303.3997。Li, H. 等(2009). 序列比对/映射（SAM）格式与 SAMtools。《生物信息学》，**25**(16), 2078–2079. Ma, X. 等(2019). 深度新一代测序数据错误谱分析。《基因组生物学》，**20**, 1–15.

Mardis, E. R. (2008年). 下一代 DNA测序方法. 《基因组学与人类遗传学年度评论》,**9**, 387–402. Minoche, A. E. 等人 (2011年). 对 Illumina HiSeq 和 Genome Analyzer 系统生成的基因组高通量测序数据的评估. 《基因组生物学》,**12**(11), 1–15. Niu, Y. 等人 (2022年). ACO：基于自适应编码顺序的无损质量分数压缩. 《*BMC*生物信息学》,**23**(1), 1–14. Schadt, E. E. 等人 (2010年). 第三代测序的一扇窗口.《人类分子遗传学》,**19**(R2), R227–R240. Wee, Y. 等人 (2019年). 基于第三代测序的基因组组装与分析生物信息学工具. 《功能基因组学简报》,**18**(1), 1–12. Xing, Y. 等人 (2017年). GTZ：一种针对 FASTQ 文件优化的快速压缩与云端传输工具. 《*BMC*生物信息学》,**18**(16), 233–242. Zhang, Y.等人 (2015年). FASTQ 数据的轻量级基于参考序列的压缩. 《*BMC*生物信息学》,**16**(1), 1–8. Ziv, J. and Lempel, A. (1977年). 一种用于顺序数据压缩的通用算法. 《*IEEE*信息论汇刊》, **23**(3), 337–343.