## SeqSimulator模拟平台

系统概述

SeqSimulator具有以下五种功能：

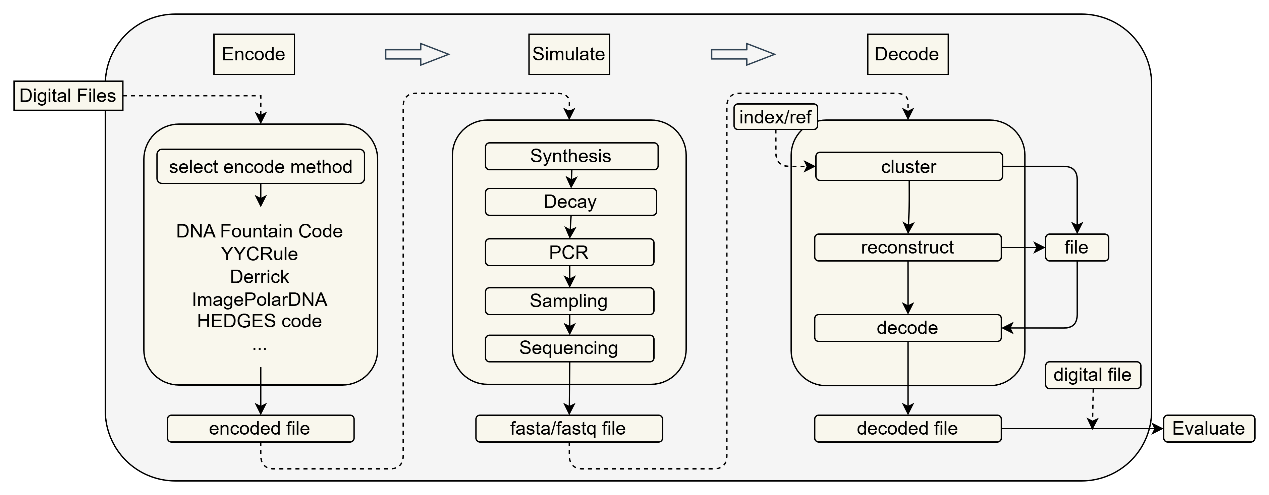
（1）支持五种DNA存储编解码方法：DNA Fountain、YinYang码、Hedges、Derrick、PolarCode。

（2）模拟DNA存储的编码、合成、衰老、PCR、采样、测序和解码等环节。

（3）提供解码过程中的聚类、重建及纠错过程。

（4）实现了DNA Fountain、YinYang码和PolarCode这三种编码方法的软解码，提升了它们的解码恢复率。

（5）用户可以直接使用任一模块进行模拟，还可以自定义编解码参数并选择不同的编码方法进行模拟评估。



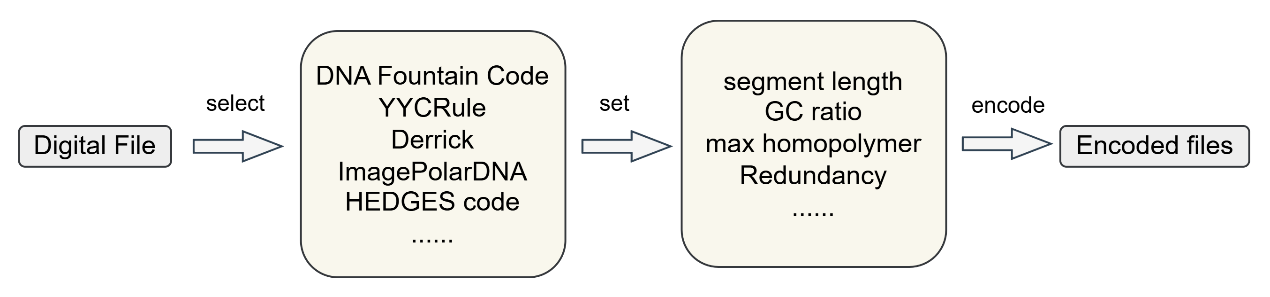
**SeqSimulator总体架构**

SeqSimulator平台的总体设计目标是为不同的DNA存储编解码方法提供一个高效、灵活、可扩展的模拟环境。该平台主要分为三个核心模块：编码模块、错误模拟模块（包括合成与测序）、解码模块，每个模块独立运行。编码模块负责将数字或其他格式的数据转换为DNA序列。模拟模块通过合成、衰老、PCR扩增、采样及测序过程模拟DNA存储的实际操作。解码模块对DNA序列进行解码，包括聚类、重建和纠错，最终恢复原始数据。通过对这些模块的有效组合与优化，平台能够支持各种DNA存储方案的开发和性能评估。

模块功能设计

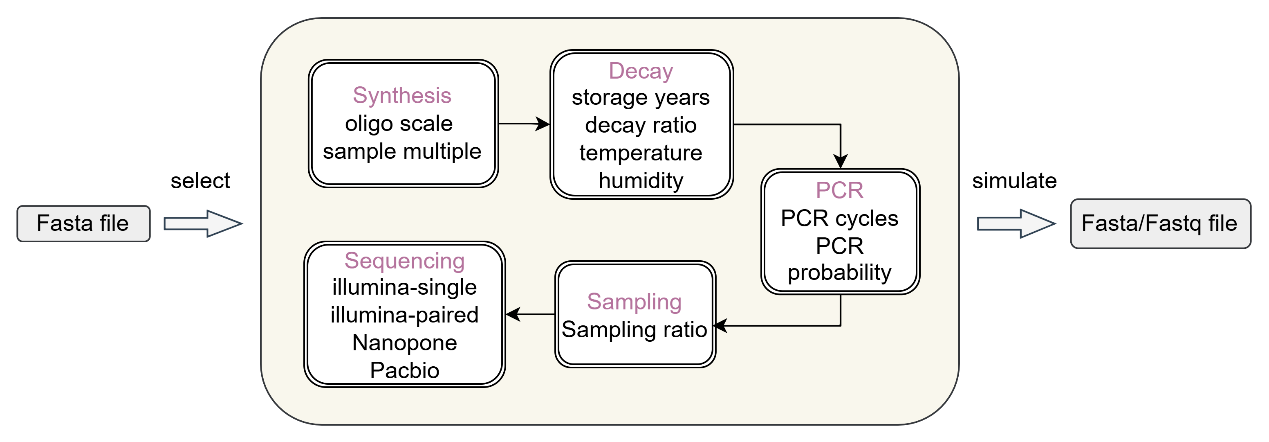
#### 编码模块

编码模块主要是将待存储文件的数字信息编码为对应的 DNA 序列，功能图如图所示。编码模块将文件转换为二进制比特流，然后借助特定的编码映射规则，把得到的二进制比特流转化为 DNA 碱基序列。编码模块实现了三种的软解码编码方案，分别为DNA Fountain、YinYang码、PolarCode，这些方法融合了特定的纠错码（如RS码），为用户提供了丰富且灵活的选择。



编码流程

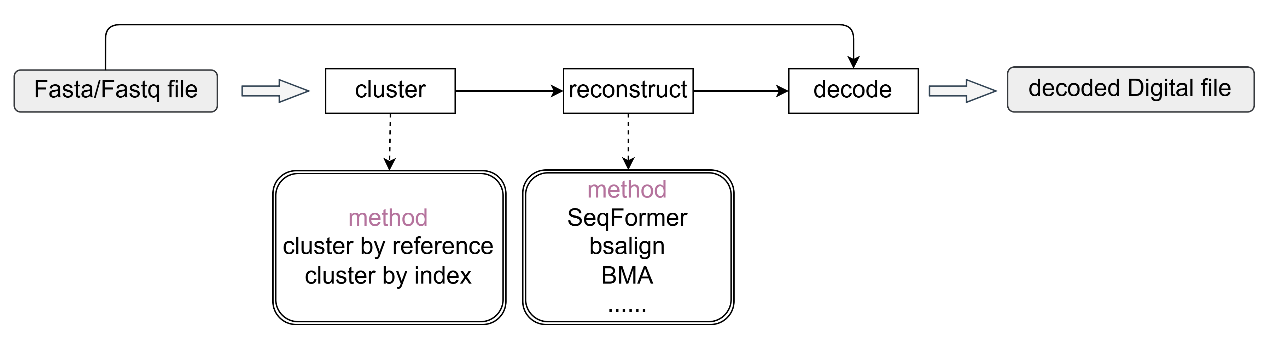
#### 模拟模块



模拟流程

模拟模块中包含了合成、存储（衰变、PCR扩增、采样）、测序五个环节的错误模拟，每个环节都包含了一些可设置的参数，用户可以任意选择多个环节进行错误模拟（合成环节一般必选）。其中，测序环节模拟了二代Illumina的单端及双端测序，三代的Nanopone及Pacbio测序。系统通过DNA 数据存储数字孪生工具[76] 模拟了DNA分子存储流程中潜在出现的多种误差情形，涵盖了DNA序列特有的丢失现象以及序列内部发生的错误，单独使用badread[77]软件模拟了三代测序错误，下面分别介绍每个步骤。

#### 解码模块



解码流程

解码模块负责从测序数据中恢复原始信息，包含了聚类、重建及解码。首先，通过聚类技术对测序结果进行整理，将重复的DNA序列聚类；其次，通过重建算法从这些DNA片段中重构consensus序列；最后，解码过程将重建后的DNA序列转换回原始数据的格式。聚类方法可以使用index/reference聚类，使用index是通过遍历用户提供的索引文件中的index，计算每条测序序列的前n位与索引的编辑距离，并把序列归类到距离最近的类中。重建方法中包含了多序列比对及按位多数对齐方法，并融合了SeqFormer序列重建模型，可以输出重建后的序列及碱基置信度。用户可以设置解码支持的最大的拷贝数量。另外，用户也可以不进行聚类，直接解码。

各个编解码方法性能对比及分析

#### 实验设计

在编解码模拟平台上，实现了多种主流编解码方法，包括了DNA Fountain、YinYang码、Hedges码、Derrick码和DNA PolarCode。本文对这些方法的性能进行了全面对比和分析。本文使用信息密度、存储成本、编码时间和物理密度来评估各个方法的性能，其中，信息密度计算公式如下：

 （1）

存储成本计算公式如下：

（2）

其中，单碱基合成成本为0.002$/nt，单碱基illumina测序成本为3.28e-6$/nt，单碱基Nanopone测序成本为2.365e-6$/nt。

物理密度计算公式如下：

 （3）

其中，DNA体积与编码后碱基数量关系如下：

 （4）

各个编码方法的默认参数如表默认参数设置如表1所示，实验的软硬件环境详细信息如表2所示。其中RS码放在原始比特序列后面，不考虑DNA存储过程中的引物。

**表1 默认参数设置**

|  |  |
| --- | --- |
| 具体参数 | 参数值 |
| 序列长度 | 260nt |
| RS码长度 | 4Byte（矩阵形式为255,32） |
| 最大均聚物 | 5nt |
| GC含量 | 0.4~0.6 |

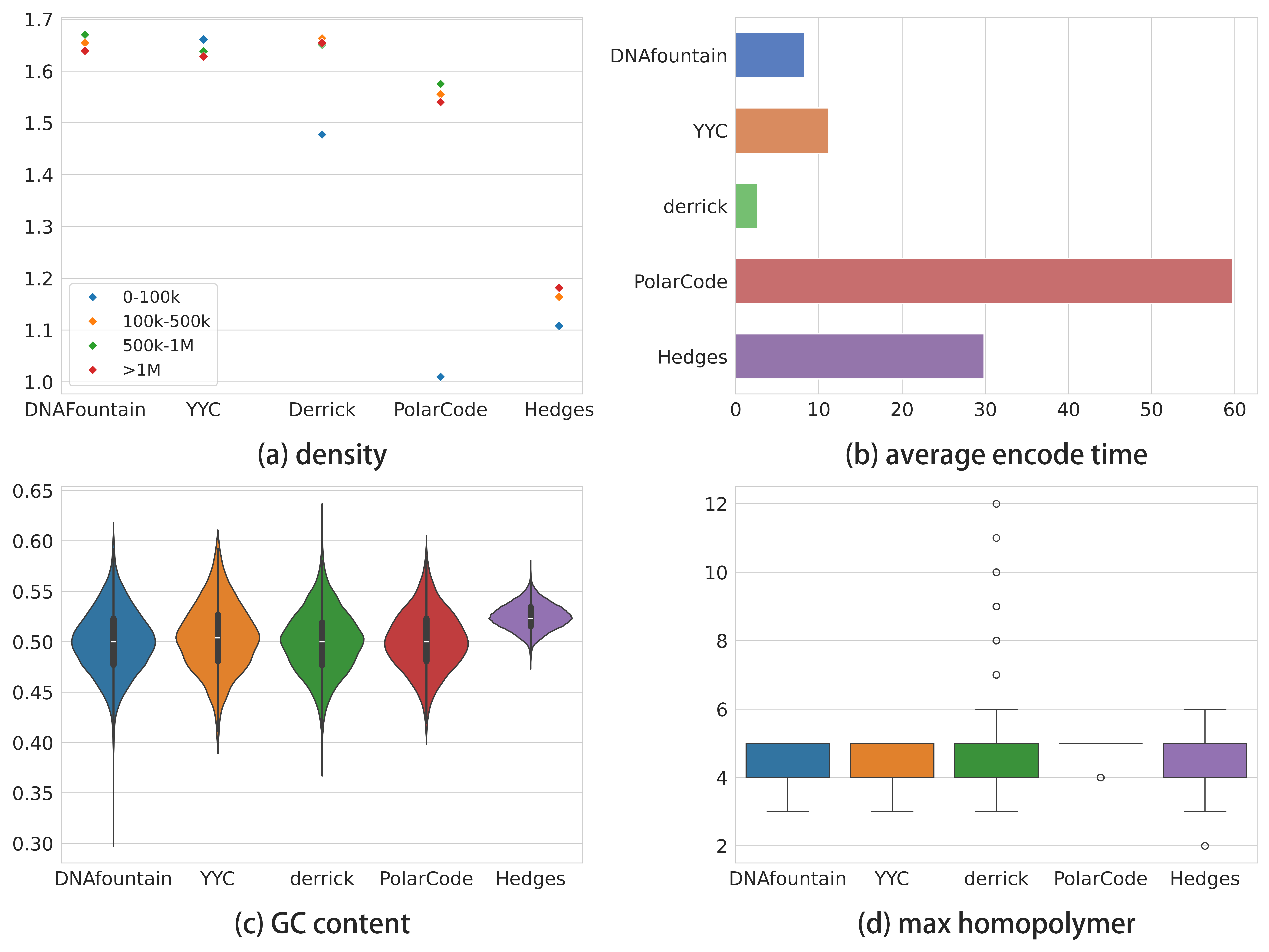
**表2 软硬件环境**

|  |  |
| --- | --- |
| 实验环境 | 具体参数 |
| CPU | Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz |
| GPU | NVIDIA Tesla P100-PCIE |
| 架构 | X86\_64 |
| 内存 | 512GB |
| 系统 | Ubuntu 16.04 |
| Python | 3.10.9 |

#### 实验结果

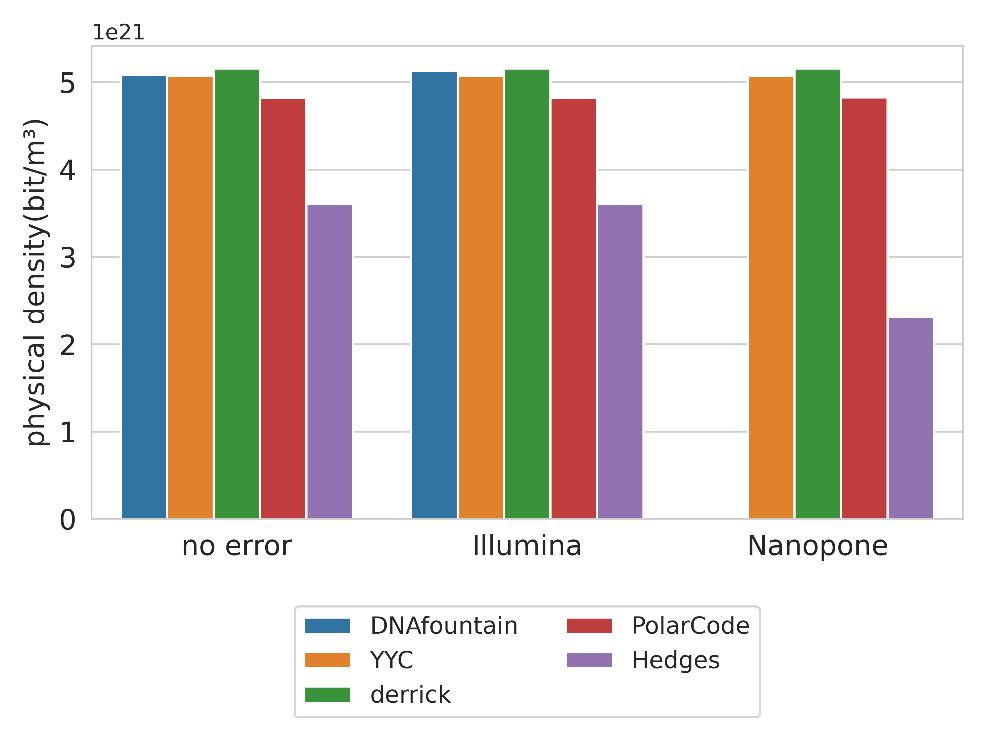
为了削减DNA存储技术的成本，转码策略通常需在确保数字信息完整性的前提下，力求最小化DNA序列的合成量，因此编码密度成为了评估转码方法性能的关键指标之一。基于表1默认参数设置，本文对不同文件大小区间内对多种数据编码方法的信息密度进行了对比，结果如图1(a)所示。DNAfountain和YYC方法在数据存储方面具有较好的性能，具有比较稳定的信息密度。另外三种方法因为是采用矩阵形式进行存储，需要满足一个矩阵大小才能编码，若二进制数据片段不足矩阵行数，会通过增加冗余序列填充矩阵，所以它们在存储小文件时具有劣势。当文件大于100KB时，信息密度才能真正反映它们的真实数据存储情况，从图中可以看出，Derrick相对其他方法表现出相对较高的信息密度。如图1(b)展示了不同方法在多个文件下的平均编码时间。从图中可以看出，使用极化码的编码时间远远大于其他方法。

较长的单核苷酸均聚物运行通常会导致DNA合成和测序过程中出现错误， 这会影响DNA中存储数据的解码。在DNA数据存储的随机存取和DNA测序步骤中，区域GC含量对聚合酶链式反应扩增有相当大的影响。故DNA存储中会避免出现极端的GC含量和单碱基重复。针对这两个指标进行了测试，调查了不同编解码方法的序列特征。如图1(c)所示，这几种编解码方法的GC含量都集中在0.4~0.6， DNA Fountain和Derrick还是存在少部分序列在40% 以下或者 60% 以上GC含量的情况。Hedges方法生成的DNA序列最为稳定。另外，如图1(d)所示的单碱基重复可以看出，这五种方法序列最大均聚物都集中在3-5之间，Derrick存在一些比较长的单碱基重复数，PolarCode均聚物分布最为集中。



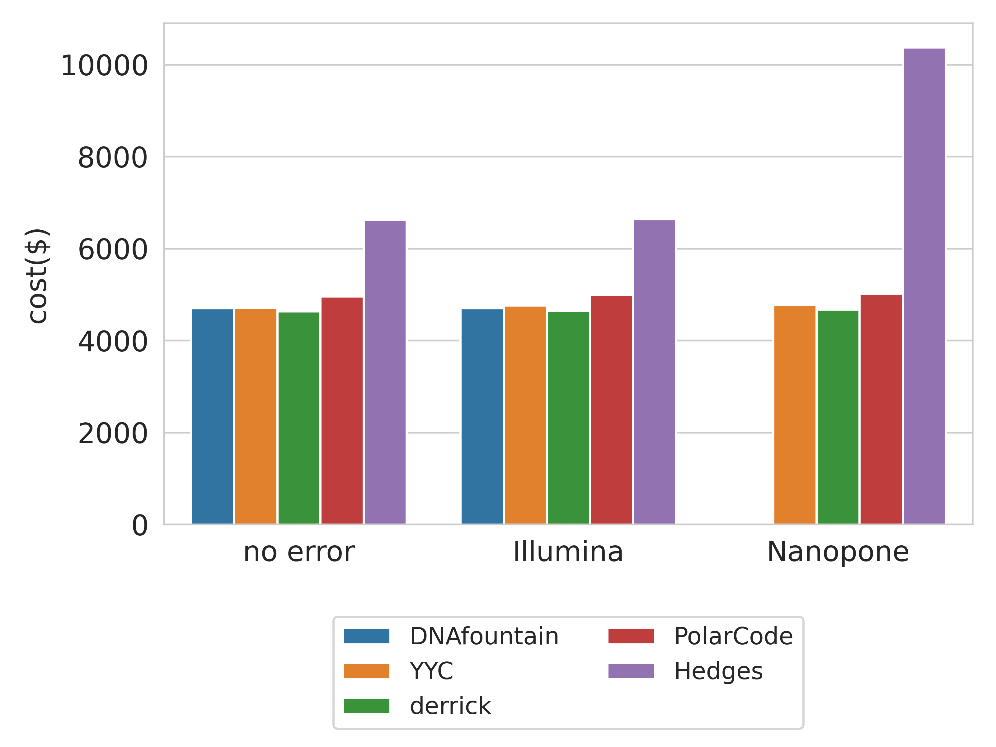
**图1各种编解码方法性能对比**

如图2所示，展示了各个编解码方法在同一文件下的物理密度对比。可以看出，Hedges的物理密度在三种情况下都最低。

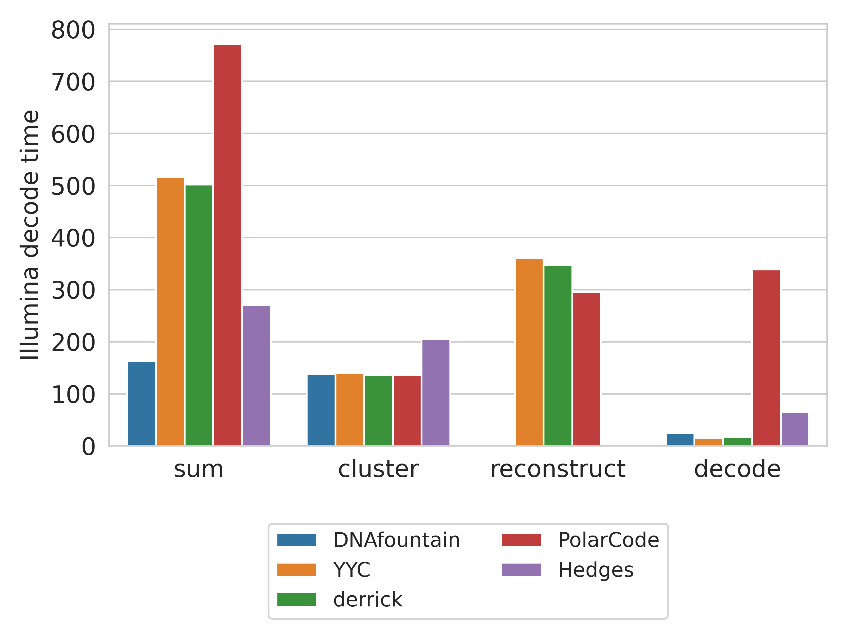


**图2 物理密度对比**

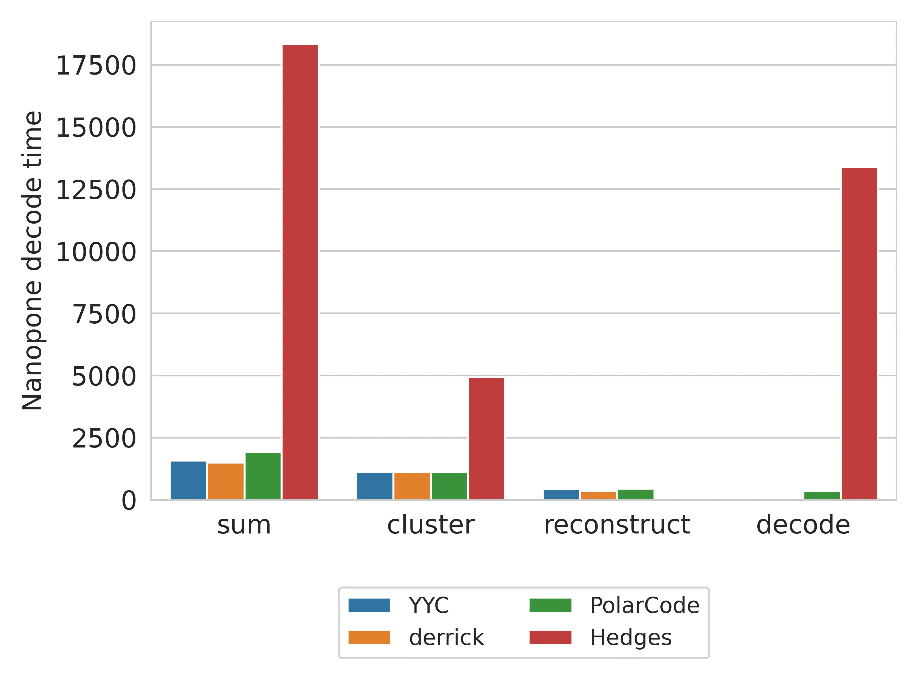
如图3所示，展示了各个编解码方法在同一文件下的DNA存储成本对比。可以看出，Hedges的成本最高，其他几种方法成本相差不大。如图4所示，展示了各个编解码方法在同一文件下的Illumina测序解码时间对比。可以看出，使用极化码解码时间最高。如图5所示，展示了各个编解码方法在同一文件下的Nanopone测序解码时间对比。可以看出，使用Hedges解码时间最高。



**图3 成本对比**



**图4 Illumina测序解码时间**

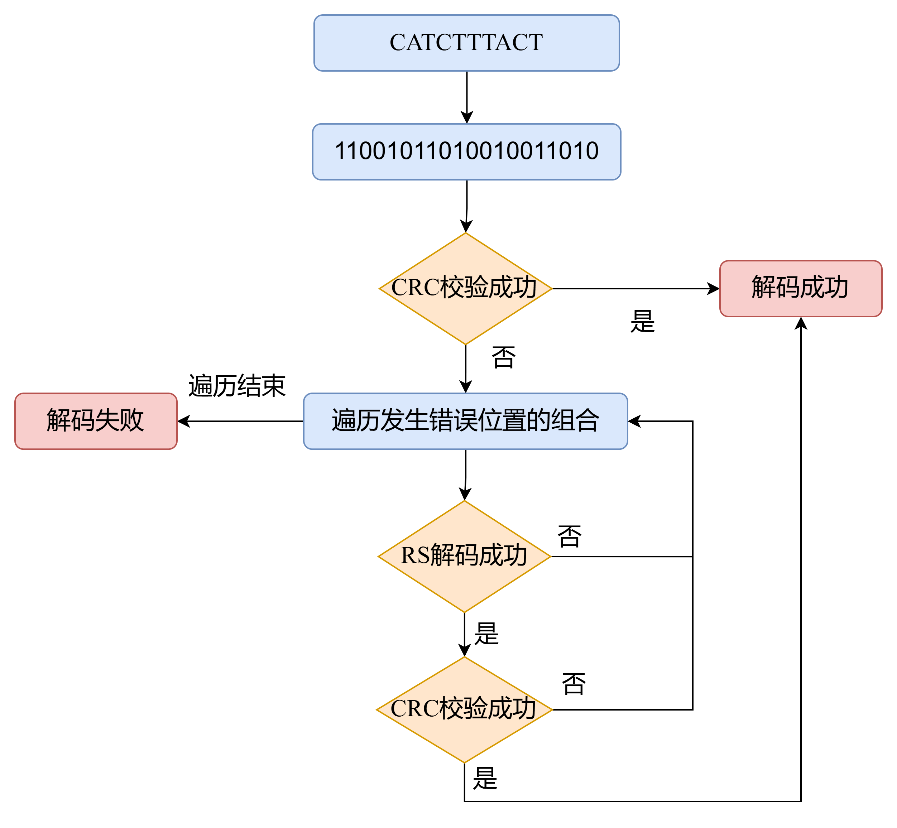


**图5 Nanopone测序解码时间**

## 软解码方法

### 基于RS的软解码

软解码算法的核心思想是对得到的一致性序列中出现的低置信度碱基进行迭代处理以提高解码恢复率，对于一个经过RS编码和CRC校验的DNA序列，获取每个碱基的置信度，低置信度的碱基将成为重点处理对象，选择前n个低置信度的位置进行软解码。RS软解码方法流程如图4-1所示。首先把DNA序列根据映射规则转换为二进制数据，使用CRC校验数据的完整性，如果通过校验，说明此数据没有发生错误，解码成功，可以直接返回数据original\_data。如果CRC校验失败，对当前序列的所有碱基置信度进行排序，得到前n个置信度最小的位置，得到种组合形式，这里的k是RS码的个数。遍历这个组合，把数据及错误位置传入RS码进行解码，若RS解码成功且CRC校验成功，则认为数据已纠正，可以返回解码后的数据，否则遍历下一个错误位置，直到所有位置遍历结束，则认为解码失败。

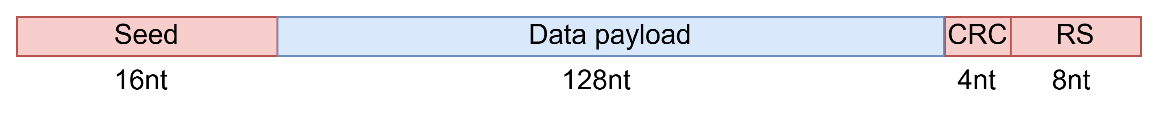


**图1 RS软解码方法流程**

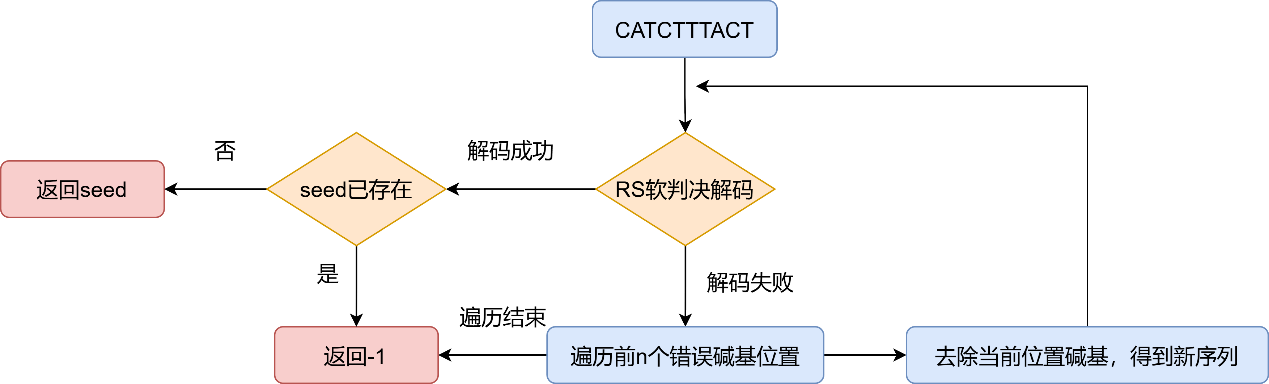
|  |
| --- |
| 算法: 基于RS的软解码算法 |
| Input: 一致性序列Consensus，一致性序列所对应的碱基置信度，置信度尝试数量n  Output: 解码后的数据decode\_data   1. 初始化decode\_data  [] 2. 将Consensus根据某种转码方法转换为数据data 3. 去掉rs位和校验位，得到original\_data及crc检验码crc\_code 4. **if** crc校验成功 **then** 5. **return** original\_data 6. 计算当前read前n个置信度最低的位置pos 7. 生成所有可能的错误位置组合(0~rs) comb 8. **for** p in comb **do** 9. decode\_data  RSDecode(data, p) 10. 根据decode\_data得到original\_data 11. **if** crc校验成功 **then** 12. return decode\_data 13. **if** rs解码失败或crc校验失败 **then** 14. 尝试下一个组合 15. **return** decode\_data |

### DNAFountain+RS的软解码

在DNA存储技术中，编码方案的效率和鲁棒性是决定存储系统性能的关键因素。DNA Fountain编码是一种基于喷泉码原理的DNA数据存储方案，它通过将原始数据分割成多个数据包，并为每个数据包生成唯一的标识符（ID），然后对这些数据包进行冗余编码，生成多个喷泉块。这些喷泉块被随机地嵌入到DNA序列中，以提高数据的恢复能力。在DNA Fountain实现基于RS的软解码结构如图2所示。



**图2基于DNA Fountain码实现的软解码结构**



**图3 DNAFountain软解码流程**

基于RS解码的原理，只使用RS码进行解码只能解决一些替换错误，在已知错误位置的情况下，尝试手动删除一些错误位置，基于DNAFountain的软解码流程如图3所示。对于一条待解码的DNA序列，先使用RS软解码进行解码，如果解码成功，检查seed是否已存在，若已存在则丢弃这个seed，否则返回seed。如果RS软解码方法失败，则遍历前n个最小置信度的错误碱基位置，删除这个位置的碱基，得到一条新的DNA序列，对新序列重新使用RS软解码方法，直到遍历结束。在DNAFountain解码时使用软解码相对于使用硬解码来说，能够得到更多有效的液滴，也就是说在解码时有更多种子可用于还原为原数据，从而能够提升解码恢复率。

|  |
| --- |
| 算法4-3: 基于DNAFountain的软解码算法 |
| Input: 一致性序列consensus，碱基置信度phreds，置信度尝试数量s，原序列长度orilen  Output: 用于DNAFountain解码的种子   1. 初始化seed  -1 2. 使用RS软解码方法得到decode\_data 3. **if** RS软解码方法失败 **then** 4. **if** len(consensus) > orilen **then** 5. phreds按从小到大排序，得到排序后的indices 6. **for** si in indices[:s] **do** 7. 去除consensus位置为si的碱基，得到新序列consensus' 8. 使用consensus'进行RS软解码方法得到decode\_data 9. **if** RS软解码方法成功 **then** 10. seed  compute\_seed(decode\_data) 11. **break** 12. **else** 13. seedcompute\_seed(decode\_data) 14. **if** seed != -1 **and** seed已存在 **then** 15. seed = -1 16. **return** seed |

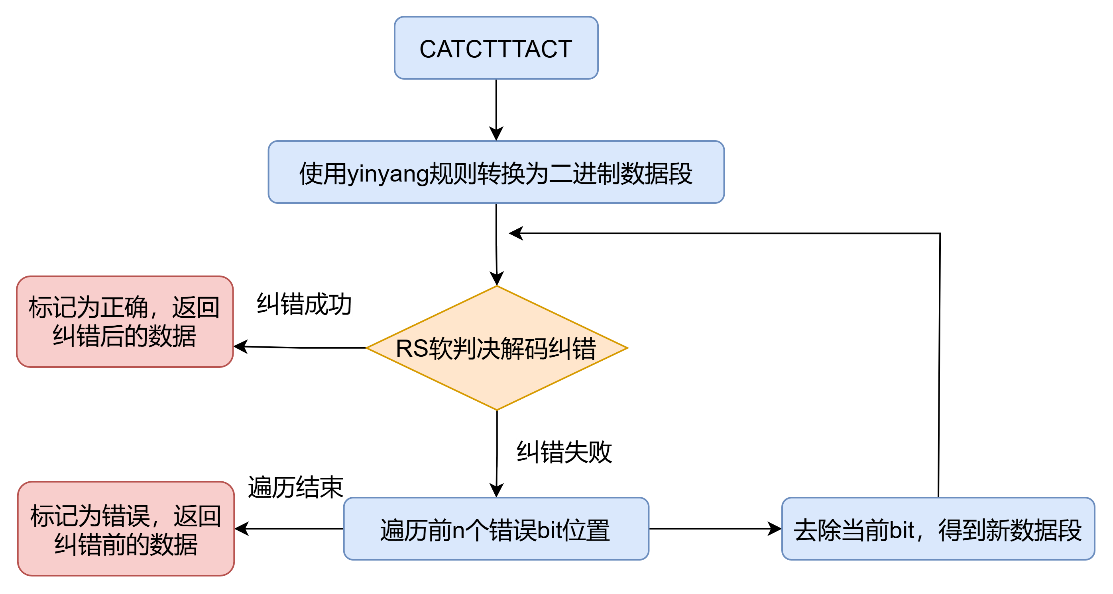
### YYC+RS的软解码

Yin-Yang码通过采用两套独立且互补的规则——“阴”规则与“阳”规则，分别对两条二进制信息流实施“一对一”映射转换，将两条原本独立的信息无缝融合为一串具有DNA序列特性的编码字符串，在YYC中实现基于RS的软解码结构如图4所示。



**图4基于YYC实现的软解码结构**

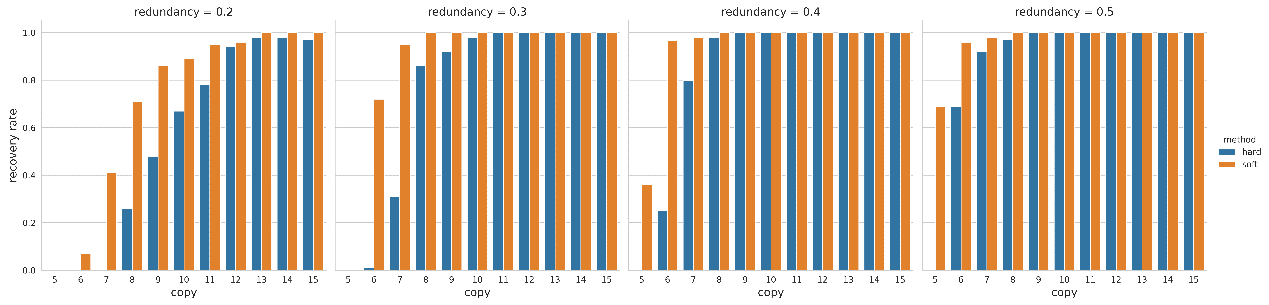
因为YYC是对两条二进制片段编码成一条序列的，故一条一致性序列解码会得到两个二进制片段，这条一致性序列中的碱基置信度也会适用于这两个二进制片段，可以为它们提供不可靠的位置，从而进行软解码方法。在YYC仅使用RS进行软解码方法也只能解决一些替换错误，基于YYC软解码完整解码流程如图4-6所示。对于一条DNA序列，首先按照yinyang规则转换为二进制数据段，然后使用RS软解码方法纠错，如果纠错成功则标记为正确，返回纠错后的数据，如果纠错失败，则遍历前n个置信度较低的bit位置，每次都去除当前位置的bit，得到新数据段，重新使用RS解码纠错，若遍历结束还未纠错成功，就标记为错误，返回纠错前的数据。



**图5 YYC软解码流程**

|  |
| --- |
| 算法4-4: 基于YYC的软解码算法 |
| Input: 一致性序列consensus，碱基置信度phreds，置信度尝试数量s，原序列长度orilen  Output: 解码后的数据rdata及其纠错结果type   1. 初始化type False, rdata none 2. 根据yinyang规则把consensus转换为binstring 3. 使用RS软解码把解码纠错得到decode\_data 4. **if** RS软解码方法失败 **then** 5. **if** len(binstring) > orilen **then** 6. phreds按从小到大排序，得到排序后的indices 7. **for** si in indices[:s] **do** 8. 去除binstring位置为si的bit，得到新的binstring' 9. 使用binstring' 进行RS软解码方法纠错得到decode\_data 10. **if** RS软解码方法成功 **then** 11. rdatadecode\_data 12. type=True 13. **break** 14. **else** 15. rdata  decode\_data 16. type=True 17. **return** rdata, type |

软解码实验对比

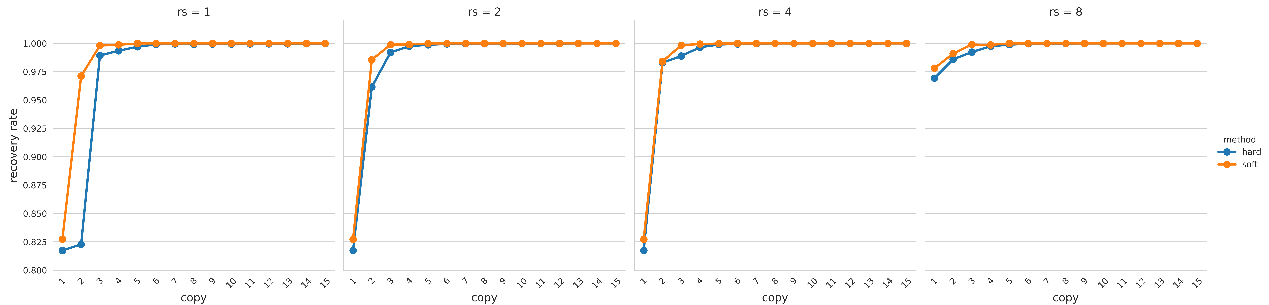


**图5 喷泉码恢复率对比**

**表3 red=0.5时 软/硬解码对比**

|  |  |  |  |
| --- | --- | --- | --- |
| Copy | Hard | Soft | 提升 |
| 5 | 0% | 68.99% | 68.99% |
| 6 | 69% | 96% | 27% |
| 7 | 92% | 98% | 6% |
| 8 | 97% | 100% | 3% |
| 9 | 100% | 100% | 0% |

实验引入了SeqFormer模型的序列及碱基置信度以实现多种编解码方法的软解码从而增强解码恢复率。为了说明软解码在DNAFountain上的有效性，本文研究了不同拷贝数量对解码恢复率的影响。拷贝数量被定义为在获取一致性序列时的最大序列数量，这一参数直接影响接收到的序列数据的准确率及其碱基置信度的可靠程度，从而进一步影响解码过程中的错误纠正能力，使用RS(36,1)及2字节的crc校验码的解码恢复率对比如图5所示。随着拷贝数量的增加，硬解码逐渐接近软解码的性能，但软解码仍稳定在更高恢复率。另外，随着冗余大小的增加，软解码和硬解码之间的差距逐渐缩小。这是因为DNAFountain编码的冗余度代表了数据传输时所附加的冗余数据量，直接影响数据恢复的容错能力。在较低冗余度时，误差的容忍度较低，软解码方法可能提供更大的性能提升。在冗余较大时，使用硬解码时只利用序列信息便可得到足够多解码成功的数据来恢复文件，这时使用软解码的用处就相对更小，在错误率较小时也是同样的道理。从表3中可得，低拷贝数量（5~7）时软解码提升更加显著，最高达68.99%。



**图7 YYC不同RS大小下软解码与硬解码对比**

**表4 rs=1时恢复率对比**

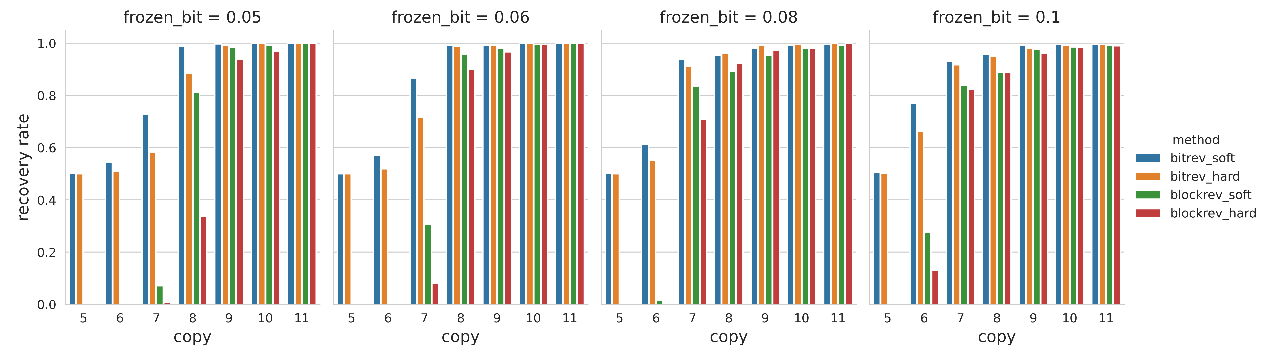
|  |  |  |  |
| --- | --- | --- | --- |
| Copy | Hard恢复率(%) | Soft恢复率(%) | 提升(%) |
| 2 | 82.28 | 97.06 | 14.78 |
| 3 | 98.93 | 99.77 | 0.84 |
| 4 | 99.35 | 99.88 | 0.53 |
| 5 | 99.7 | 99.99 | 0.29 |
| >=6 | ≥99.90 | 100 | 0.1 |

**表5 平均提升结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 指标 | 软解码 | 硬解码 | 平均提升(%) |
| bit恢复率 | 97.52 | 99.82 | 2.3 |
| block恢复率 | 97.54 | 99.83 | 2.29 |
| 100%恢复次数 | 68.2 | 96.1 | 27.9 |

使用YYC在不同纠错码(RS(36,1), RS(36,2) RS(36,4) RS(36,8))大小下进行软解码的实验结果对比如图7所示。从图中可以看出，使用软解码能提升yyc纠错能力，rs越大，纠错能力相差越小。第一是因为使用碱基置信度信息可以给RS码提供不准确的位置，从而提高文件恢复恢复率，第二是因为通过碱基置信度信息估计不准确的比特位置，从而去进行删减，这可以解码RS码无法解决的插入问题。从表4和表5可以看出，使用软解码可以明显提高文件恢复次数，平均提升了27.9%。另外，在低拷贝时恢复率最高可提升14.78%。

本文还使用极化码对比了在不同冻结比特率下，使用软解码与硬解码的比特恢复率和数据块恢复率。在极化码中，冻结比特是指在编码过程中被设置为固定值（通常为0或1）的比特。这些比特的值在编码和解码过程中不会改变或参与信息传输。在10%冻结比特的情况下，编码过程中用于传输信息的比特将更少，但可能带来解码性能的提高。如图8展示了这几个冻结比特下不同拷贝数量的恢复性能。看图可知，软解码在低冗余时优势明显，高冗余时差距缩小，但软解码仍更稳定。根据表6数据可知，低拷贝时提升显著，软解码平均比硬解码提升约2.6%。



**图8 极化码不同冻结比特下软解码与硬解码对比**

**表6 平均恢复率对比**

|  |  |  |  |
| --- | --- | --- | --- |
| 指标 | Hard均值 | Soft均值 | 平均提升 |
| bitrev\_5 | 0.5 | 0.503 | 0.30% |
| bitrev\_6 | 0.56 | 0.624 | 6.40% |
| bitrev\_8 | 0.911 | 0.94 | 2.90% |
| bitrev\_10 | 0.95 | 0.957 | 0.70% |
| 综合均值 | 0.73 | 0.756 | 2.60% |