Rapid and robust assembly and decoding of molecular tags with DNA-based nanopore signatures

Kathryn Doroschak, Karen Zhang, Melissa Queen, Aishwarya Mandyam, Karin Strauss & Jeff Nivala

摘要

分子标签是一种使用 DNA 或其他分子标记物理对象的方法。当 RFID 标签和 QR 码等方法不适用时，可以使用这种方法。目前不存在廉价、快速且可靠地解码并且可用于在最少资源环境中创建或读取标签的分子标记方法。为了解决这个问题，我们提出了 Porcupine，它是一种分子标记系统，具有基于 DNA 标签，可使用便携式纳米孔设备在几秒钟内读取信息。Porcupine 的数值位为 0 或 1，是由是否存在对应 DNA 分子片段决定。这些 DNA 分子片段称为分子比特 (molbits)。我们直接根据 DNA 分子比特通过纳米孔产生的电信号，对不同分子比特 molbits 进行分类。这可以避免进行 DNA 碱基测序。为了延长寿命、减少读出时间并使标签对环境污染具有鲁棒性，分子比特 molbits 在组装过程中为读出过程的准确性做好了准备，并且可以通过脱水使其更稳定。实验结果表明这是一个可扩展的、实时的、高精度的分子标记系统，其中包括一种生成高区分度的分子比特的方法。

关键词：分子标签；Porcupine；molbits

1 引言

在生活和生产中，我们经常会给物品贴标签。例如包装中的 UPC 条形码、用于轻松关联数字信息与印刷材料的 QR 码，以及用于库存跟踪的射频识别 (RFID) 标签[1]。但是，这些标签不能应用于太小、太灵活或太多的对象，或者用于肉眼不可见代码的场景，例如防伪。分子标签通过其纳米级足迹和难以伪造来解决这些缺点。然而，现有的在分子中编码数字信息的方法；包括二氧化硅封装的 DNA 示踪剂[2] 、嵌入 3D 打印材料中的 DNA[3]、微生物条形码[4] 或空间分离的标记肽[5] ；需要访问专门的实验室和设备来制作新标签——这使得它们在需要大量标签的应用中不切实际。在大多数情况下，该协议会阻止实时用例，例如，在基于 PCR 或基于 SHERLOCK 的检测的情况下，最好的情况下需要数十分钟[6]。一个理想的分子标记系统应该是廉价和可靠的，具有快速读出和用户控制的端到端编码和解码，对实验室设备的依赖最小。

分子生物学家也一直对标记生物分子感兴趣，例如，通过在每个测序片段上附加一个样本特异性短 DNA 标签，在单次测序运行中对样本进行多重分析。特别是对于纳米孔测序，直接原始信号技术提高了解引用多重条形码的效率。DeepBinner 对 Oxford Nanopore Technologies 的 96 个商用条形码中的 12 个原始信号进行分类，改进了他们之前基于序列的工具 Porechop[7]。同样，DeePlexiCon 对四种手工设计的 DNA 条形码进行了分类，以便在直接 RNA 测序中进行多路复用，这些条形码以前在商业上是不可获得的[8]。虽然这些工具能够提高解复用读取的产量，或者在 RNA 的情况下，使复用成为可能，但如果条形码旨在优化原始信号可分离性，它们的性能可能会更好。还开发了其他工具用于后期测序管道中的原始信号处理，例如基因组比对[9]。

Porcupine 以基于纳米孔的 DNA 测序技术和原始信号处理工具的最新进展为基础，解决了先前基于 DNA 的标记方法的局限性。便携式实时纳米孔测序[10] 的发展，以及简化预定义 DNA 序列[11] 模块化组装的新方法，为在低资源环境中快速写入和按需读出创造了更多机会。

2 相关工作

Porcupine 是一种分子标记系统，它的使用基于合成的 DNA 分子片段（分子比特）和纳米孔的电流信号。Porcupine 通过存在或不存在预定的 96 个 DNA 片段来编码相应数值位，存在即为 1，不存在即为 0。我们把这个 DNA 分子片段称之为分子比特（molbits，由 Cafferty 等人创造[5]），如图 1a[1]所示。尽管通常认为 DNA 的读写成本很高，但 Porcupine 通过预合成 DNA 分子片段降低了成本，然后可以任意混合以创建新的分子标签。用户可使用便携式低成本测序设备（Oxford Nanopore Technologies 的 MinION；图 1b[1]）快速读出分子标签。通常，原始纳米孔信号必须首先转换回 DNA 序列，这个过程称为 basecalling，其计算量很大。但我们直接从纳米孔电信号对分子比特进行分类，而避免了 basecalling。这可以大大提高效率和准确性。对纳米孔电信号分类通常也用于 DNA 和 RNA 样本的多路分解，这也使用了 DNA 条形码[7][8]；Porcupine 通过特别设计分子片段的 DNA 碱基序列来产生独特的电流特征，并且大大增加了条形码的数量。纠错码也添加到标签中以减少解码错误，就像电子信息传输系统一样[12]。分子比特 Molbits 在设计时就准备好用于读出（测序），并且可以通过脱水来变得更稳定。脱水可以延长分子标签保质期、减少解码时间并减少环境中 DNA 对其可能的污染。因此，创建标签需要一套预先准备好的分子比特 Molbits、测序适配器套件及其相应的要求（例如，离心机、热循环仪和旋转混合器）和脱水方法；阅读分子标签只需要标签、无核酸酶水和一个 MinION 设备。最终，它是一个高度准确的实时标记系统，其中包括生成高区分度的分子比特 molbits 的方法。这些分子比特 molbits 以及我们用来开发它们的方法是可扩展的；它们既可以在 Porcupine 中用于标记物理对象，也可以用于其他分子级标记需求，例如用于纳米孔测序的样品多路复用。

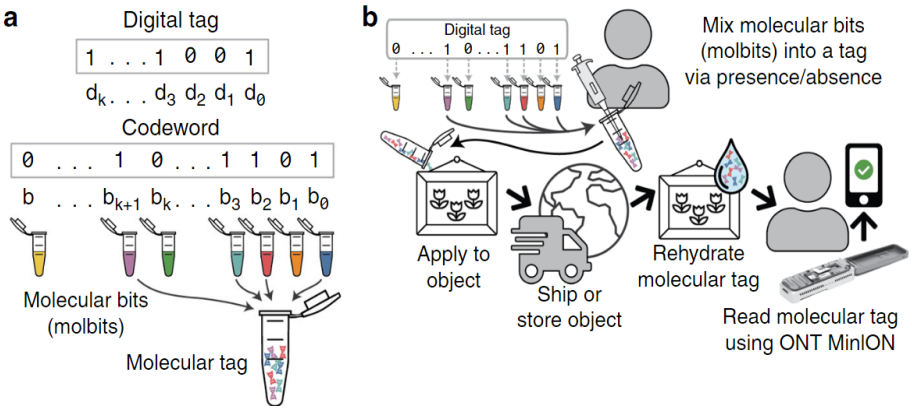

图 1: 利用 Porcupine 创建一个分子标签

3 本文方法

3.1 本文方法概述

3.1.1 Molbit 定义和组装

为了开发 Porcupine，首先将单个 molbit 定义为 DNA 链，它结合了独特的条形码序列（40 nt）和从预先确定的一组序列长度中选择的更长的 DNA 片段（图 2a[1]）。为了使 molbits 的组装简单和模块化，这里将它们设计为与 Golden Gate Assembly 兼容，Golden Gate Assembly 是一种方便且可扩展的单锅 DNA 组装方法，通过结合短的单链悬垂[11]结合了 TypeIIS 限制酶和连接酶。为了提高分类准确性并减少计算时间，我们进一步优化它们以避免碱基调用。因此，对于条形码区域，目标是生成大量序列，这些序列可以生成独特的离子电流特征（“波浪线”）以促进明确的分类。

3.1.2 设计高区分度的 molbits

为了对任意 DNA 序列的预测离子电流特征进行建模，我们使用了“Scrappie squiggler”，这是一种通过卷积模型将碱基序列转换为离子电流的工具。例如，为了展示 Scrappie 准确模拟真实纳米孔波浪线的能力，我们手工设计了一个 DNA 序列，在波浪线空间中显示为字母“UW”（图 2b[1]），与模拟波浪线具有高度视觉相似性（噪音除外）. Scrappie 的输出还让我们使用动态时间扭曲 (DTW) 作为距离度量来定量计算两个序列的信号相似性。我们在旨在使条形码尽可能可分离的进化模型中使用这种方法（图 2c[1]）。

为了生成一组 96 个正交 molbit 条形码序列，使用 96 个随机或预置起始序列初始化了进化模型。作者通过在随机位置同时突变两个相邻的核苷酸，以随机顺序独立地扰乱每个序列。如果突变序列未能提高其自身与所有其他序列之间的最小和平均 DTW 相似性，将逆转突变并再次尝试相同序列。我们还限制了序列的序列相似性和自由能，以避免标记歧义和二级结构。使用这种方法，我们从一组起始序列开始，该序列的最小 DTW 相似度为 2.9，平均值为 4.2 ± 0.4，经过 31 轮进化后达到最终最小值 4.2，平均值为 5.8 ± 0.8（图 2d[1]），代表最小值和平均值均提高约 40%。

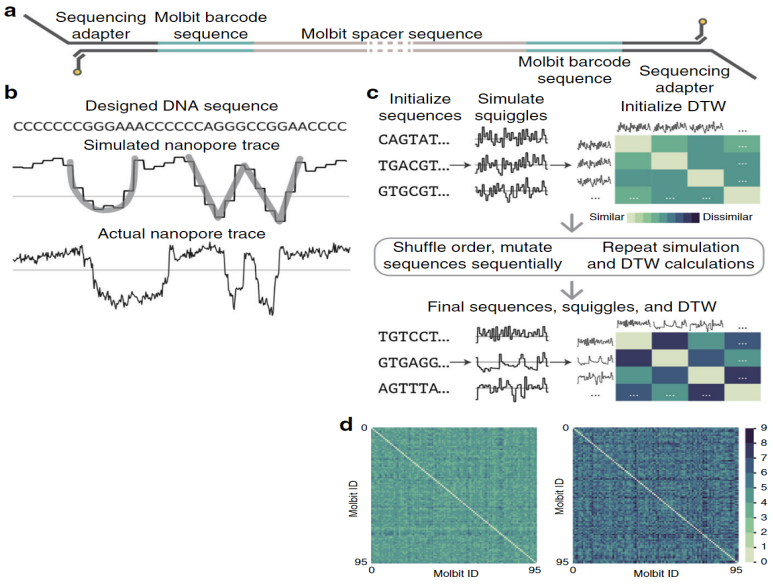

图 2: 分子比特 molbits 的设计

3.1.3 直接利用纳米孔电信号对 molbits 进行识别

在设计 molbits 并从理论上了解它们的可区分后，论文作者开发了一个卷积神经网络 (CNN) 模型，以直接从真实的原始纳米孔数据中对它们进行准确分类。直接信号分类优于基于序列的方法，主要是因为分类本质上是一个比全序列解码更简单的问题，在全序列解码中，问题可以简化为区分 96 个不同的信号，而不是再现产生任意信号的确切的潜在核苷酸系列。这使得更简单的模型架构成为可能，这些架构通常具有较低的计算和训练数据要求。在我们的模型中，molbits 不需要被分割或以其他方式从原始数据中分离出来，相反，CNN 只使用每个 molbits 的第一部分，因为 molbits 条形码位于链的开头。我们通过将 96 个 molbit 条形码分成六个序列集来收集训练数据，每个 molbit 出现一次；优化集合以获得最大序列可分离性以改进标记；并在 MinION 上运行每个集合。我们使用传统的碱基调用方法和修改后的半局部 Smith-Waterman 序列比对[13]为每个 molbit 读数分配标签，仅使用高置信度比对。对于测试数据，我们将 96 个 molbits 分成两个序列集，每个 molbits 出现一次，并按照训练所述收集数据。该模型经过 108 次迭代训练，与序列衍生标签相比，最终训练、验证和测试准确率分别为 99.9%、97.7% 和 96.9%。然而，在现实世界的解码中，所有的 reads 都是分类的，而不仅仅是那些通过 basecalling 和序列比对的 reads。论文作者发现 CNN 始终能够自信地对更大部分的读数（测试集中 97% 的读数）进行分类，而不是碱基调用加比对（测试集中 75% 的读数），揭示了我们认为的大部分读数无法轻易验证。论文作者推断，如果每个 molbit 的出现在碱基识别和 CNN 之间成比例，则 CNN 可能不会进行虚假识别，但可能在原始信号数据上表现更好。两种方法之间的读数计数相关性非常好，揭示了直接信号分类的另一个优势，比单独测序多使用了约 30% 的数据。因此，“准确率”仅反映模型准确率，并不一定衡量每个 molbit 的错误率或错误解码标签的总体机会。

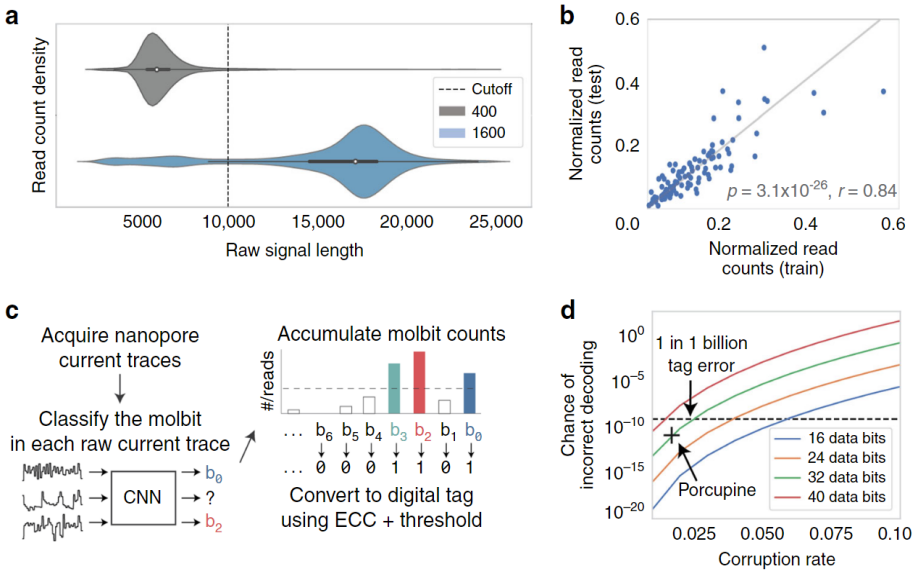

图 3: 分子比特 molbit 的识别和标签的解码结果

3.1.4 通过在有纠错能力的分子标签中编码数据

论文作者接下来组成实际的分子标签。我们在二进制标签中为每个 molbit 分配了一个独特的位置，允许每个 1 或 0 代表在单个标签排序（解码）运行过程中特定 molbit 的存在或不存在。为了确定是否存在，我们对每个 molbit 使用了 CNN 分类的读取计数。理想情况下，0 位将具有零读取，而 1 位

将具有非零读取。然而，有两个因素使我们设置此阈值以确定比特存在变得复杂：(1) 实验中本应不存在的 molbit，但其计数往往非零；(2) 实验中存在的 molbit 计数会显着变化。幸运的是，当我们比较训练和测试数据中 molbit 计数的比率时，这些变化是一致的（图 3b[1]）。我们通过基于这些比率的固定向量缩放所有读取计数来解释这种变化。阈值和缩放读取计数将我们的每比特错误率从 2.9 ± 1.8% 降低到 1.7 ± 1.6%，降低了 42%。

由于可靠的标记系统应该有非常低的错误解码机会（例如，十亿分之一），我们决定通过将纠错码 (ECC) 作为我们标签设计的一部分来进一步降低我们的整体标签解码错误率（图 3c[1]). 在这些标签中编码信息的最简单的非 ECC 方法是数字位和摩尔位之间的原始 1:1 映射；然而，使用这种方法，即使是单个位错误也会使标签无法恢复（即产生不正确的解码）。在我们的系统中，使用读取计数存在或不存在的阈值将位设置为 1 或 0，这意味着高于该阈值的任何 0 位都将翻转为 1，反之亦然。尽管每位错误率相对较高，但 ECC 通过为数字消息保留较少的位数并通过将此消息投影到具有更大可分离性的更大空间来创建代码字，从而降低了不可恢复标签的可能性。这允许在消息被错误解码之前翻转更多位。为了对数字消息进行编码，我们只需将消息乘以随机数的二进制矩阵，称为随机生成矩阵（补充方法）。为 ECC 保留的位数取决于应用程序的错误容限和每位错误率（图 3d[1]）。随着错误率的增加，错误解码的机会呈指数增加。因此，必须仔细选择消息的位数。我们选择了 32 位的消息大小，在 1.7% 的错误率下产生 1.6×10^{-11} 的错误解码机会，并允许约 42 亿个唯一标签，正确解码保证在 9 位错误或以下。

3.1.5 端到端编码和解码

接下来，作为标记系统的原理验证，论文作者演示了“分子信息系统实验室”的缩写“MISL”的端到端标记编码和解码（图 4a[1]）。论文作者首先使用 ASCII 将 MISL 编码为二进制，每个字符使用 8 位，总共 32 位，然后将该位向量乘以生成矩阵以生成 96 位代码字。然后如前所述制备分子标签，对实验室效率进行了一项修改（参见方法部分）。一旦组装好分子标签，就可以使用 ONT MinION 对其进行测序和读出。然后，使用训练有素的 CNN 分类器从原始数据中识别出 molbits，为每个 molbit 累积一个计数，并重新调整这些计数（如上所述）以适应系统的读取计数差异。然后，通过使用滑动读取计数阈值对计数进行二值化以确定存在与否，然后在每个读取计数阈值处找到二值化计数与最近的有效代码字之间的距离，从而按照针对 ECC 所述对标签进行解码。当编辑距离足够低以保证唯一的正确解码时（此 ECC 的距离为 9），分子标签解码完成。最早的正确解码发生在样品加载后不到 7 秒（观察到 109 个 molbit 分子链），证明使用便携式测序仪仅需几秒钟即可对 32 位消息进行可靠的编码和解码。

4 复现细节

4.1 与已有开源代码对比

这里使用了论文作者的源代码进行复现。考虑到更高效地处理较多 fast5 文件，我将解码程序 decoder 的线程数量由原作者默认的 64 个，提高到 128 个（执行的代码指令如图 4）。

```
(pytorch) wangJL@ubuntu:/data02/wangjinlong/Porcupine/ecc$ make clean
rm -f decoder.o decoder
(pytorch) wangJL@ubuntu:/data02/wangjinlong/Porcupine/ecc$ make THREAD_POW=7
gcc -c -o decoder.o decoder.c -O3 -Wall -pthread -DTHREAD_POW=7
gcc -o decoder decoder.o -O3 -Wall -pthread -DTHREAD_POW=7
(pytorch) wangJL@ubuntu:/data02/wangjinlong/Porcupine/ecc$
```

图 4: 将 decoder 程序线程数改为 128

4.2 实验环境搭建

安装 anaconda，以及与服务器对应版本的 pytorch 11.0 + CUDA 11.0；实现在本地电脑浏览器连接服务器的 jupyter notebook，并运行代码。

4.3 创新点

为提高程序处理较多 fast5 文件时的效率，我将解码程序的线程数由原作者默认的 64 个，提高到 128 个后。实验结果表明，在处理较多 fast5 文件时，这样的改动确实能提升效率，平均可减少约 15% 的时间。如表 1 所示：

实验编号	Fast5 文件数（个）	线程数	挂钟时间（s）
1	15	64	318
		128	**281**
2	29	64	616
		128	**557**
3	73	64	1296
		128	**966**

表 1

5 实验结果分析

用过运行代码，将 fast5 文件中的电信号信息最终还原为 32 位二进制数据，并且解码出来的二进制数据符合预期。如图 5所示：

```
In [12]: best_msg, best_d = decode_run(f5_dir, model_file, decoder, possible_labels, molbit_data_file, cnn_label_file,
                                       overwrite=False, conf_thresh=0.95, batch_size=500, n_workers_cnn=30,
                                       scaling_factors=scaling_factors, decoding_guarantee=9)

Reading in fast5 data from path/to/fast5/data_01_test3.
Searching dir: path/to/fast5/data_01_test3
Beginning data extraction (15 fast5 files).
Saving to file (60000 reads): path/to/reads/molbit_extracted_data.hdf5
/tmp/ipykernel_9729/696812445.py:62: UserWarning: swmr=True only affects read ('r') mode. For swmr write mode, set f.swmr_mode = True after opening the file.
  with h5py.File(molbit_data_file, "w", swmr=True) as f:
/tmp/ipykernel_9729/696812445.py:64: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  f.create_dataset("data", shape=signal_data_stacked.shape, dtype=np.float, data=signal_data_stacked)
Reading in pretrained CNN.
Beginning classification.
Saving classifications to test_labels.csv.
Beginning decoding.
Decoded: 01001101010010010101001101001100, 9

In [21]: # The true message can gbe given as a bit string (just set true_msg directly) or as a list of molbits
molbits_in_set = [1,4,5,7,9,12,15,17,19,22,23,25,28,29]  # etc.
true_msg = "".join([str(x) for x in get_read_counts(molbits_in_set, possible_labels=possible_labels)])[:32]

In [22]: print(best_msg)
print(true_msg)
print("".join(["0" if s1 == s2 else "1" for s1, s2 in zip(best_msg, true_msg)]))
print(sum([0 if s1 == s2 else 1 for s1, s2 in zip(best_msg, true_msg)]))

01001101010010010101001101001100
01001101010010010101001101001100
00000000000000000000000000000000
0
```

图 5: 实验结果

6 总结与展望

总之，Porcupine 提供了一种基于合成 DNA 序列存在与否的分子标记方法，这些序列可生成独特的纳米孔原始电流信号。通过直接识别纳米孔原始电流的片段并保持序列较短，我们降低了最终用户的合成成本，并产生可见的独特的纳米孔电流信号，从而实现高精度解码。快速的解码时间意味着我们的系统可以使用更新的技术进行解码，例如 Flongle，一种由 ONT 生产的更便宜的一次性设备，它只有 MinION 孔数的约四分之一，用时约 1-3 分钟。此外，标签可以在标签创建时准备好用于测序，然后通过脱水来稳定保存，进一步减少读出时间，但代价是通过测序准备增加“写入”时间。在对新制备的且脱水的标签（脱水后第 0 周和第 4 周）进行测序时，正确的标签解码和测序产量、Q 分数和碱基检出序列长度的最小变化支持了准备工作似乎具有最小的风险。

将来，可以通过增加更多的插入长度、扩展独特区域的长度以允许 molbits 之间的更多变化或通过串行组合条形码区域来获取更多的信息位。此外，用于 molbit 设计的生成模型可能是下一个方向，特别是如果需要大量的 molbits，因为进化模型计算随所需位数呈指数增长。

通过本次论文复现，我学习了如何给服务器搭建环境（安装 anaconda，pytorch，CUDA 等），还学会了如何利用本地电脑的浏览器运行服务器的 jupyter notebook，并学习了如何运行和调试作者原代码，知道了可以通过增加线程数来提高对较多 fast5 文件的处理效率。另外，我对原代码中的解码程序 decoder 还有许多地方没有看明白。在后面的学习中，我将尝试更好的理解该解码程序的算法原理和代码实现，以及进一步复现本论文的重要工作之一：生成 96 个电信号差异较大的（高区分度的）DNA 分子片段（分子比特）。

参考文献

[1] DOROSCHAK K. Rapid and robust assembly and decoding of molecular tags with DNA-based nanopore signatures[J]. Nature Communication, 2020.

[2] Mikutis. Silica-encapsulated DNA-based tracers for aquifer characterization[J]. Environ. Sci. Technol, 2018, 52: 12142-12152.

[3] Koch. A DNA-of-things storage architecture to create materials with embedded memory[J]. Nat. Biotechnol., 2020, 38: 39-43.

[4] Qian. Barcoded microbial system for high-resolution object provenance[J]. Science, 2020, 368: 1135-1140.

[5] Cafferty. Storage of information using small organic molecules[J]. ACS Cent. Sci., 2019, 5: 911-916.

[6] Kellner. nucleic acid detection with CRISPR nucleases[J]. Nat. Protoc., 2019, 14: 2986-3012.

[7] WICK L M, Judd. Deepbinner: demultiplexing barcoded Oxford nanopore reads with deep convolutional neural networks[J]. PLoS Comput. Biol, 2018, 14: e1006583.

[8] SMITH M A. Barcoding and demultiplexing Oxford nanopore native RNA sequencing reads with deep residual learning[J]. Genome Res, 2020, 30: 1345-1353.

[9] Han. Novel algorithms for efficient subsequence searching and mapping in nanopore raw signals towards targeted sequencing[J]. Bioinformatics, 2020, 36: 1333-1343.

[10] Jain. The Oxford nanopore MinION: delivery of nanopore sequencing to the genomics community[J]. Genome Biol., 2016, 17: 239.

[11] ENGLER C M. Combinatorial DNA assembly using golden gate cloning[J]. Methods Mol. Biol., 2013, 1073: 141-156.

[12] ENGLER C M. Multicomponent molecular memory[J]. Nature Communication, 2020, 11: 1-8.

[13] SMITH T F W. Identification of common molecular subsequences[J]. J. Mol. Biol., 1981, 147: 195-197.