[MECAT2][https://github.com/xiaochuanle/MECAT2]是MECAT的升级版本, 用于单分子测序reads的校正, 组装，比对软件.

包含一下四个模块:

- mecat2pw, 快速准确pairwise比对
- mecat2ref, 快速准确参考基因比对
- mecat2cns, 基于pairwise重叠校正噪音reads
- fsa, 基于组装工具的string graph

We present a concept and formalism, the string graph, which represents all that is inferable about a DNA sequence from a collection of shotgun sequencing reads collected from it. We give time and space efficient algorithms for constructing a string graph given the collection of overlaps between the reads and, in particular, present a novel linear expected time algorithm for transitive reduction in this context.

MECAT2不支持Nanopore raw reads, 针对Nanopore raw reads请使用MECAT

Quick Start

1. 下载数据

```
$ mkdir -p ${MECAT_PATH}/ecoli
$ cd ${MECAT_PATH}/ecoli
$ wget http://gembox.cbcb.umd.edu/mhap/raw/ecoli_filtered.fastq.gz
$ gunzip ecoli_filtered.fastq.gz
```

2. 创建配置模版文件

   `mecat.pl config ecoli_config_file.txt`, 包含一下内容:

```
PROJECT=
RAWREADS=
GENOME_SIZE=
THREADS=4
MIN_READ_LENGTH=500
CNS_OVLP_OPTIONS=""
CNS_OPTIONS="-r 0.6 -a 1000 -c 4 -l 2000"
TRIM_OVLP_OPTIONS="-B"
ASM_OVLP_OPTIONS="-n 100 -z 10 -b 2000 -e 0.5 -j 1 -u 0 -a 400"
FSA_OL_FILTER_OPTIONS="--max_overhang=-1 --min_identity=-1"
FSA_ASSEMBLE_OPTIONS=""
USE_GRID=false
CLEANUP=0
CNS_OUTPUT_COVERAGE=30
GRID_NODE=0
```

   填入相关信息:

```
PROJECT=ecoli
RAWREADS=/home/chenying/smrt_asm/ecoli/ecoli_filtered.fastq
GENOME_SIZE=4800000
THREADS=4
MIN_READ_LENGTH=2000
CNS_OVLP_OPTIONS=""
CNS_OPTIONS="-r 0.6 -a 1000 -c 4 -l 2000"
TRIM_OVLP_OPTIONS="-B"
ASM_OVLP_OPTIONS="-n 100 -z 10 -b 2000 -e 0.5 -j 1 -u 0 -a 400"
FSA_OL_FILTER_OPTIONS="--max_overhang=-1 --min_identity=-1"
FSA_ASSEMBLE_OPTIONS=""
USE_GRID=false
CLEANUP=0
CNS_OUTPUT_COVERAGE=30
GRID_NODE=0
```

3. 校正reads

   `mecat.pl correct ecoli_config_file.txt`

4. 使用校正后reads组装

   `mecat.pl assemble ecoli_config_file.txt`

5. 查询结果

   - The corrected reads is `${MECAT_PATH}/ecoli/ecoli/1-consensus/cns_reads.fasta`.
   - The extracted longest 30x corrected reads used for trimming is `${MECAT_PATH}/ecoli/ecoli/1-consensus/cns_final.fasta`.
   - The trimmed reads is `${MECAT_PATH}/ecoli/ecoli/2-trim_bases/trimReads.fasta`
   - The assembled contigs is `${MECAT_PATH}/ecoli/ecoli/4-fsa/contigs.fasta`

Input Format

MECAT2可以处理fasta, fastq, H5格式文件. 然而H5格式需先通过`${MECAT_PATH}/DEXTRACT/dextract`转换为fasta格式后采用运行MECAT2

```
$ find pathto/raw_reads -name "*.bax.h5" -exec readlink -f {} \; > reads.fofn
$ while read line; do   dextract -v $line >> reads.fasta ; done <  reads.fofn
```

Program Descriptions

Config File

MECAT2通过配置文件读取所有信息, 包括项目名称, 原始reads, 和多种运行参数, 使用`mecat.pl config cofig_file_name`构建配置文件, 例如以上配置:

```
PROJECT=ecoli
RAWREADS=/home/chenying/smrt_asm/ecoli/ecoli_filtered.fastq
GENOME_SIZE=4800000
THREADS=4
MIN_READ_LENGTH=2000
CNS_OVLP_OPTIONS=""
CNS_OPTIONS="-r 0.6 -a 1000 -c 4 -l 2000"
TRIM_OVLP_OPTIONS="-B"
ASM_OVLP_OPTIONS="-n 100 -z 10 -b 2000 -e 0.5 -j 1 -u 0 -a 400"
FSA_OL_FILTER_OPTIONS="--max_overhang=-1 --min_identity=-1"
FSA_ASSEMBLE_OPTIONS=""
USE_GRID=false
CLEANUP=0
CNS_OUTPUT_COVERAGE=30
GRID_NODE=0
```

- PROJECT=ecoli, 项目名称, 在当前目录会建立ecoli目录用于该项目所有运行
- RAWREADS=, 原始reads文件路径, 需绝对路径
- GENOME_SIZE=, 潜在基因组的大小(bp)
- THREADS=, CPU线程
- MIN_READ_LENGTH=, 校正后和过滤后reads的最短长度
- CNS_OVLP_OPTION="", 在校正阶段用于检测重叠reads的选项(detection overlap candidates) `mecat2pw`
- CNS_OPTIONS="", 校正原始reads的选项 `mecat2cns`
- TRIM_OVLP_OPTIONS="", 在过滤阶段用于检测重叠的选项 `v2asmpm`
- ASM_OVLP_OPTIONS="", 在组装阶段用于检测重叠的选项 `v2asmpm.sh` 默认参数就好
- FSA_OL_FILTER_OPTIONS="", 过滤重叠的选项 `fsa_ol_filter`
- FSA_ASSEMBLE_OPTIONS="", 组装过滤后reads的选项 `fas_assemble`
- USE_GRID=false, 使用多重计算节点(true or false)
- CLEANUP=0, 使用删除中间数据(1表示是/0表示否)
- CNS_OUTPUT_COVERAGE=30, 用于组装的过滤后的, 最长的校正后的reads的覆盖度, 例如30x, 30 x 4800000 = 144MB
- GRID_NODE=0, 计算节点使用数目, 当USE_GRID=1时使用

```
V2asmpm -Pworkpath -Tt -Sx -Ey -B

-Pworkpath 工作目录是workpath
-Tt 用t个CPU线程
-Sx -Ey 这两个参数一起表示只计算第x到第y卷(包括第y卷), 通常工作目录下会有000001.fasta,
-B 如果有这个选项, 就输出二进制格式的比对结果; 如果没有这个选项, 就输出文本格式的比对结果
```

The MECAT2 Workflow

为了方便, 将所有过程整合到了一个perl脚本文件`metcat.pl`, 按照一下步骤运行:

`mecat.pl config`, 构建配置文件. `mecat.pl correct`, 校正原始reads, 包含一些两个步骤:

使用`mecat2pw`检出重叠的候选reads(detection overlap candidates); 使用`mecat2cns`根据检出重叠的候选reads校正原始reads

`mecat.pl assemble`, 使用一下三个步骤组装校正后reads:

使用`v2trim.sh`通过二步骤提取出30x 最长的校正后reads, 过滤掉低质量次级序列:

使用`v2lcr`和`v2sr`根据重叠过滤掉低质量序列; 使用`v2asmpm.sh`检出提取reads中的重叠

通过三步组装过滤后的reads:

使用`v2asmpm.sh`检出过滤后reads的重叠; 使用`fsa_ol_filter`过滤掉低质量重叠; 基于高质量重叠, `fsa_assemble`组装校正后reads为contigs

Pairwise Mapping Tool mecat2pw

```
mecat2pw -j [task] -d [fasta/fastq] -w [working folder] -t [# of threads] -o
[output] -n [# of candidates] -a [overlap size] -k [# of kmers] -g [0/1]
```

- -j [task] 任务名称, 0用于仅检测重叠候选[detect overlapping candidates only], 1用于以M4格式输出重叠. 若意欲修正噪音reads, 输出重叠的候选就足够(if we are to correct noisy reads, outputing overlapping candidates is enough)
- -d [fasta/fastq] reads名称
- -w [working folder] 用于存储临时结果的路径
- -t [# of threads] CPU线程, 默认1
- -o [output] 输出文件名称
- -n [# of candidates] 考虑到gapped extension的候选数目(number of candidates considered for gapped extension), 默认为100. 由于each chunk大约2GB, 因此number of candidates(NC)应根据基因组大小设定: GS < 20 M, NC设为200; GS > 20 M 和 GS < 200M, NC设为100; GS>200M, NC设为50
- -a [overlap size] 仅输出大于该值的重叠. 默认2000
- -k [# of kmers] 在两个reads间的两个blocks具有大于等于k kmer匹配将会考虑为一个匹配的block对. 默认为: 4
- -g [0/1] 输出gapped 延伸的起点(1)或(0), 默认为0

[输出][https://github.com/xiaochuanle/MECAT2]

略

Reference Mapping Tool mecat2ref

`mecat2ref`用于向参考基因组比对SMRTreads:

```
mecat2ref -d [reads] -r [reference] -w [folder] -t [# of threads] -o [output] -b [# of results] -m [output format]
```

- -d [reads] fasta/fastq格式reads文件
- -r [reference] fasta格式参考基因组文件
- -w [folder] 存储临时文件的路径
- -t [# of threads] CPU线程数目
- -o [output] 输出文件名称
- -b [# of result] 输出最佳 b 比对
- -m [output format] 输出格式: 0=ref, 1=M4, 2=SAM, 默认为0

[输出][https://github.com/xiaochuanle/MECAT2]

略

Correction Tool mecat2cns

`mecat2cns`是一个用于高噪音单分子测序reads的实用性错误修正reads. 准确度等同于`pbdagcon`, 速度比拟`FalconSense`. 输入文件可以是`can`或`M4`格式:

```
mecat2cns [options] overlaps-file reads output
```

- -i [input type] 输入格式, 0位`can`, 1为`M4`
- -t [# of threads] CPU线程
- -p [batch size] reads被分割的批次大小

- -r [ratio] 最小的比对率
- -a [overlap size] 使用大于该值的重叠
- -c [coverage] 最小覆盖度, 默认为6
- -l [length] 校正后序列的最小长度

其他选项的默认值为:

```
-i 1 -t 1 -p 100000 -r 0.6 -a 1000 -c 4 -l 2000
```

假如输入为`M4`格式, 那么[overlap-file]中重叠结果必须包含gapped extension起点, 这意味着`mecat2pw`中的-g选项设置为1, 否则`mecat2cns`将会运行失败.

输出: 校正后序列为给定的fasta格式, 校正后序列头信息为:

```
>A_B_C_D
```

where

- `A` is the original read id
- `B` is the left-most effective position
- `C` is the right-most effective position
- `D` is the length of the corrected sequence

mecat2elr

为`mecat.pl`用于从校正后数据中提取30X最长序列的程序:

```
mecat2elr [the input fasta file from mecat2cns] [genome size]  [coverage] [the output filename]
```

Overlap Filter fsa_ol_filter

用于过滤掉低质量重叠:

```
fsa_ol_filter [optioins] overlaps filtered_overlaps
```

- --min_length=INT 最短reads, 默认2500
- --max_length=INT, 最长reads, INT_MAX
- --min_identity=DOUBLE, 最小一致性重叠, 默认90
- --min_aligned_length=INT, 重叠的最短比对长度, 默认2500
- --max_overhang=INT, 最大的重叠overhang, 默认为10
- --min_coverage=INT, 最小的碱基覆盖度, 默认为-1
- --max_coverage=INT, 最大碱基覆盖度, 默认为-1
- --max_diff_coverage=INT, 最大的碱基覆盖度差异, 默认为-1
- --coverage_discard=DOUBLE, 舍弃碱基覆盖度比率, 默认为0.01. 如果`--max_coverage`或`--max_diff_coverage`为负值, 该值将会重设为百分数(100 - coverage_dicard)
- --overlap_file_type="|m4|paf|ovl", 重叠文件格式(默认为:""). ""=为文件名延伸,'m4'为M4格式,

'paf'=为PAF格式, minimap2生成, 'ovl'=为OVL格式, falcon生成
- --bestn=INT, 针对每个read输出5‘或3'最佳的n个重叠(默认为10)
- --genome_size=INT, 基因组大小, 将和--coverage一起决定reads的最大长度
- --coverage=INT, 覆盖度, 将和--genome_size一起决定reads的最大长度
- --thread_size=INT, 线程数目(默认4)

Assemble Tool fsa_assemble

用于从过滤后的重叠和校正后reads构建contigs. 算法类似FALCON:

```
fsa_assemble [optioins] filtered_overlaps
```

- --min_length=INT, 最小reads长度(默认0)
- --min_identity=INT, 最小重叠一致性(默认0)
- --min_aligned_length=INT, 最小的重叠比对长度(默认0)
- --min_contig_length=INT, 最小的contigs长度(默认500)
- --read_file=STRING, fasta或fastq格式reads文件
- --overlap_file_type='|m4|paf|vol', 重叠文件格式(默认“”). ""=文件名延伸, 'm4'=M4格式, 'paf'=PAF格式, minimap2, 'ovl'=OVL, FALCON生成的OVL格式
- --output_director=STRING, 输出文件路径(默认为当前:.)
- --select_branch='no|best', 当在graph中遇到branches时所采用的方法 , 'no'=不选择任何branch, 'best'=选择最大可能的brance
- --thread_size=INT, 线程数目(默认4)
