Contents

1 Table of Contents

Created by gh-md-toc

2 Assembly Pipeline

Firstly, the relevant modules need to be loaded:

```
$ module load extenv/rdbioseq assemblage
```

After, we concatenate nanopore reads (Minion + Promethion)

```
$ zcat
    /env/cns/proj/projet_CPD/BG/RunsNanopore/191024_MN19040_FAL24795_A/CPD_BG_ONT_1_FAL24795_A
    /env/cns/proj/projet_CPD/BG/RunsNanopore/191216_PCT0037_PAD99440_A/CPD_BG_ONT_1_PAD99440_A
    > nanopore.fastq.gz
$ gunzip -c nanopore.fastq.gz > nanopore.fastq
```

2.1 Remove chimeric reads from fastq file (YACRD)

We run YACRD (version 0.6.0), a chimeras detector tool which performs in two steps:

1. Computation of pile-up coverage for each read:

```
$ nohup benchme minimap2 -t 48 -x ava-ont -g 500 nanopore.fastq nanopore.fastq >
    overlap.paf 2 > minimap.e &
```

2. Then, YACRD takes the resulting PAF (Pairwise Alignement Format) from minimap2 and proceeds to the detection of chimeras. YACRD proposes several post-detection operations among which is "scrubb":

    scrubb: for sequence file all bad region are removed, NotCovered read is removed

    Read scrubbing overlapping recommended parameter: For nanopore data, we recommend using `minimap2` with all-vs-all nanopore preset with a maximal distance between seeds fixe to 500 (option `-g 500`) to generate overlap. We recommend to run YACRD with minimal coverage fixed to 4 (option `-c`) and minimal coverage of read fixed to 0.4 (option `-n`).

```
$ nohup benchme yacrd -i overlap.paf -o report.yacrd -c 4 -n 0.4 scrubb -i
    nanopore.fastq -o nanopore.scrubb.fastq 1> yacrd.o 2> yacrd.e &
```

2.2 Assemble reads with NECAT

NECAT offers an easy-to-use pipeline:

1. Install the tool as follows:

```
$ wget
    https://github.com/xiaochuanle/NECAT/releases/download/v0.0.1_update20200803/necat_20
$ tar xzvf necat_20200803_Linux-amd64.tar.gz
$ cd NECAT/Linux-amd64/bin
$ export PATH=$PATH:$(pwd)
```

2. Create a config file template using the following command:

```
$ necat.pl config config.txt
```

3. Fill and modify the relative information in the template configuration file. In particular, we edit the following variables:

    `PROJECT=necat_all_reads_yacrd`, `ONT_READ_LIST=reads.txt`, `GENOME_SIZE=310000000`, `THREADS=40`, `PREP_OUTPUT_COVERAGE=60`, `CNS_OUTPUT_COVERAGE=40`

the above `reads.txt` file contains the **full** path of the reads file (the fastq file resulting from YACRD -in our case).

```
$ cat reads.txt
/pathTo/nanopore.scrubb.fastq
```

4. We run directly the bridging-step using the following command:

```
$ nohup benchme necat.pl bridge config.txt 1> necat.o 2> necat.e &
```

Note : This command checks and runs the preceding steps first, which consist of:

1. Correcting raw noisy reads

   The pipeline only corrects longest 60X (`PREP_OUTPUT_COVERAGE`) raw reads.
   The corrected reads are in the files `/pathTo/1-consensus/cns_iter${NUM_ITER}/cns.fasta`.
   The longest 40X (`CNS_OUTPUT_COVERAGE`) corrected reads are extracted for assembly, which are in the file `/pathTo/1-consensus/cns_final.fasta`)

2. Assembling the contigs

   The assembled contigs are in the file `/pathTo/4-fsa/contigs.fasta`.

The bridged contigs are in the file `/pathTo/6-bridge_contigs/bridged_contigs.fasta`.

```
$ mv /pathTo/6-bridge_contigs/bridged_contigs.fasta NECAT_bridged_contigs.fasta
```

Note: If `POLISH_CONTIGS` is set, the pipeline uses the corrected reads to polish the bridged contigs. Although, we set this parameter to `true` we skip the polished assembly `/pathTo/6-bridge_contigs/polished_contigs.fasta` and we, instead, polish the `/pathTo/6-bridge_contigs/bridged_contigs.fasta` file with 3 rounds of racon followed by 2 rounds of Hapo-G.

2.3 Polish NECAT contigs (3 x racon + 2 x Hapo-G)

Polishing can be done with `nanopore_assembly_pipeline`:

```
$
  /env/ig/soft/rdbioseq/assemblage-snapshot//linux-noarch/bin/nanopore_assembly_pipeline
  --step polishing --assembly NECAT/6-bridge_contigs/NECAT_bridged_contigs.fasta
  --assembly_dir NECAT/6-bridge_contigs --pe1
  Reads_Illumina/BG_4_191217_R1.fastq.gz --pe2
  Reads_Illumina/BG_4_191217_R2.fastq.gz --pe1
  Reads_Illumina/BG_6_191121_R1.fastq.gz --pe2
  Reads_Illumina/BG_6_191121_R2.fastq.gz --qos h72
```

2.4 Separate haploid sub-assemblies (HaploMerger2)

To obtain the reference (haploid) assembly of T.molitor we use HaploMerger2. Soft-masking the genome needs to be done before.

Soft-mask NECAT contigs:

```
$ jobify -b -p normal --qos=h72 -e meg.e -o meg.o 'meg -BMCMD msub -f
  Polished_NECAT_contigs.fasta -dir masking -species insecta -rps
  -nbr_cores_repeatMasker 10'
$ jobify -b -q prod -c 10 'maskFastaFromBed -fi Polished_NECAT_contigs.fasta -bed
  masking/collapse/pos2msk -fo masking/soft.msk.genome.bedtools.fa -soft'
```

Download HaploMerger2:

```
$ wget
  https://github.com/mapleforest/HaploMerger2/releases/download/HaploMerger2_20161205/HaploM
$ tar -zxf HaploMerger2_20180603.tar.gz
```

Clean up fasta sequences from “illegal” characters:

```
$ cat soft.msk.genome.bedtools.fa | HaploMerger2_20180603/bin/faDnaPolishing.pl --legalizing --maskShortPortion=1 --noLeadingN --removeShortSeq=1 > genome_cleaned.fa
```

Add to PATH `chainNet`, `lastz`, `SSPACE` and `GapCloser` and set “openable” file handle limit:

```
$
   PATH=/pathTo/HaploMerger2_20180603/Molitor/chainNet_jksrc20100603_centOS6/:/pathTo/HaploMe
$
   PATH=/pathTo/HaploMerger2_20180603/Molitor/chainNet_jksrc20100603_centOS6/chainNet:/pathTo
$ ulimit -n 800000
```

Create a file corresponding to 5-10% of the genome for the matrix:

```
$ getseq -len -f genome_cleaned.fa | awk 'BEGIN {p=1; } { sum+=$2; if(sum>30000000) { p=0; } if(p) { print $1; }}' > list_5-10_percent
$ getseq -list list_5-10_percent -f genome_cleaned.fa > genome_5-10_percent.fasta
$ getseq -len -f genome_cleaned.fa | sort -k2,2nr | awk 'BEGIN {p=0; } { sum+=$2; if(sum>30000000) { p=1; } if(p) { print $1; }}' > list_other
$ getseq -list list_other -f genome_cleaned.fa > genome_other.fasta
```

Create the score matrix with wrapper `lastz_D_Wrapper.pl`:

```
$ jobify -b -p normal -c 24 '/pathTo/HaploMerger2_20180603/bin/lastz_D_Wrapper.pl --target=genome_5-10_percent.fasta --query=genome_other.fasta --identity=95'
```

Rename the file, according to the message in `lastz_D_Wrapper.pl-job-eelefthe-etna.11911401.out`:

> genome_5-10_percent.genome_other.raw.1584372088.xx_xx.q is needed to be modified and has its name changed to genome_5-10_percent.genome_other.q in order to feed to axtChain!

```
$ mv genome_5-10_percent.genome_other.1584372088.25_75.q genome_5-10_percent.genome_other.q
```

Copy the following matrix to the `scoreMatrix.q` file, according to `genome_5-10_percent.genome_other.q` file:

A	C	G	T
70	-93	-50	-92
-93	100	-87	-50
-50	-87	100	-93
-92	-50	93	70

Rename genome:

```
$ mv genome_cleaned.fa molitor_cleaned.fa
$ gzip molitor_cleaned.fa
```

2.4.1 Haplomerger steps

2.4.1.1 A (round 1)

Prepare (and enter) the folder:

```
$ mkdir A_round1
$ cd A_round1
```

Copy all scripts with prefix `hm.batchA*` to the new working directory `A_round1`.

Edit the script suffixed with extension `initiation_and_all_lastz` and set the respective variables in order to use all available cores (eg: for 36 cores and an assembly size of 440MB, we set targetSize to 13MB). For instance, while editing the script `hm.batchA1.initiation_and_all_lastz`:

```
name=$1
threads=36
identity=80
targetSize=13000000
querySize=1600000000
```

The `*.initiation_and_all_lastz` script is subsequently launched as a job:

```
$ jobify -b -o out.hm.batchA1.initiation_and_all_lastz -e
   err.hm.batchA1.initiation_and_all_lastz -n 36 -q normal --qos=h72
   ./hm.batchA1.initiation_and_all_lastz molitor_cleaned
```

Edit the script suffixed with extension `chainNet_and_netToMaf` and set the number of threads again (perhaps less are needed this time). For example while editing `hm.batchA2.chainNet_and_netToMaf`:

```
name=$1
threads=24
```

The `*.chainNet_and_netToMaf` script is subsequently launched as a job:

```
$ jobify -b -o out.hm.batchA2.chainNet_and_netToMaf -e
   err.hm.batchA2.chainNet_and_netToMaf -n 24 -q normal
   ./hm.batchA2.chainNet_and_netToMaf molitor_cleaned
```

Then, submit the `*.misjoin_processing` script:

```
$ jobify -b -o out.hm.batchA3.misjoin_processing -e
   err.hm.batchA3.misjoin_processing -n 1 -q normal
   ./hm.batchA3.misjoin_processing molitor_cleaned
$ cd ../
```

2.4.1.2 A (round 2)

Prepare (and enter) the folder:

```
$ mkdir A_round2
$ cd A_round2
```

Copy all scripts with prefix `hm.batchA*` to the new working directory `A_round2`. The fasta file generated during step `A_round1` will now be used for step `A_round2`. So,

```
$ ln -s A_round1/molitor_cleaned_A.fa.gz A_round2/molitor_cleaned.fa.gz
```

Edit the script suffixed with extension `initiation_and_all_lastz` and set the respective variables.

The `*.initiation_and_all_lastz` script is subsequently launched as a job:

```
$ jobify -b -o out.hm.batchA1.initiation_and_all_lastz -e
    err.hm.batchA1.initiation_and_all_lastz -n 36 -q normal --qos=h72
    ./hm.batchA1.initiation_and_all_lastz molitor_cleaned
```

Subsequently, set the variables of `*.chainNet_and_netToMaf` script and submit it as a job:

```
$ jobify -b -o out.hm.batchA2.chainNet_and_netToMaf -e
    err.hm.batchA2.chainNet_and_netToMaf -n 24 -q normal
    ./hm.batchA2.chainNet_and_netToMaf molitor_cleaned
```

Then, submit the `*.misjoin_processing` script:

```
$ jobify -b -o out.hm.batchA3.misjoin_processing -e
    err.hm.batchA3.misjoin_processing -n 1 -q normal
    ./hm.batchA3.misjoin_processing molitor_cleaned
$ cd ../
```

2.4.1.3 B

Prepare (and enter) the folder:

```
$ mkdir B_steps
$ cd B_steps
```

Copy all scripts with prefix `hm.batchB*` to the new working directory `B_steps`. The fasta file generated during step `A_round2` will now be used for `B_steps`. So,

```
$ ln -s A_round2/molitor_cleaned_A.fa.gz B_steps/molitor_cleaned.fa.gz
```

Edit the script suffixed with extension `initiation_and_all_lastz` and set the respective variables.

The `*.initiation_and_all_lastz` script is subsequently launched as a job:

```
$ jobify -b -o out.hm.batchB1.initiation_and_all_lastz -e
    err.hm.batchB1.initiation_and_all_lastz -n 36 -q normal --qos=h72
    ./hm.batchB1.initiation_and_all_lastz molitor_cleaned
```

Subsequently, set the variables of `*.chainNet_and_netToMaf` script and submit it as a job:

```
$ jobify -b -o out.hm.batchB2.chainNet_and_netToMaf -e
    err.hm.batchB2.chainNet_and_netToMaf -n 24 -q normal
    ./hm.batchB2.chainNet_and_netToMaf molitor_cleaned
```

Launch, sequentially, three more scripts, waiting for the first to finish before the next one starts.

Haplomerger:

```
$ jobify --chrono -b -o out.hm.batchB3.haplomerger -e err.hm.batchB3.haplomerger
    -n 1 -q prod ./hm.batchB3.haplomerger molitor_cleaned
```

Refine unpaired sequences:

```
$ jobify --chrono -b -o out.hm.batchB4.refine_unpaired_sequences -e
    err.hm.batchB4.refine_unpaired_sequences -n 16 -q prod
    ./hm.batchB4.refine_unpaired_sequences molitor_cleaned
```

Lastly, merge paired and unpaired sequences:

```
$ jobify --chrono -b -o out.hm.batchB5.merge_paired_and_unpaired_sequences -e
    err.hm.batchB5.merge_paired_and_unpaired_sequences -n 1 -q prod
    ./hm.batchB5.merge_paired_and_unpaired_sequences molitor_cleaned
$ cd ..
```

2.4.1.4 C (Not Applicable)

N/A

2.4.1.5 D (round 1)

Prepare (and enter) the folder:

```
$ mkdir D_round1
$ cd D_round1
```

Copy all scripts with prefix `hm.batchD*` to the new working directory `D_round1`. Two fasta files result from step B. Normally, only the reference haploid assembly needs to be processed (it usually has the biggest size). This assembly will be used for step `D_round1`.So,

```
$ ln -s B_steps/molitor_cleaned_ref.fa.gz D_round1/molitor_cleaned.fa.gz
```

Edit the script suffixed with extension `initiation_and_all_lastz` and set the respective variables.

The `*.initiation_and_all_lastz` script is subsequently launched as a job:

```
$ jobify -b -o out.hm.batchD1.initiation_and_all_lastz -e
    err.hm.batchD1.initiation_and_all_lastz -n 36 -q normal --qos=h72
    ./hm.batchD1.initiation_and_all_lastz molitor_cleaned
```

Subsequently, set the variables of `*.chainNet_and_netToMaf` script and submit it as a job:

```
$ jobify -b -o out.hm.batchD2.chainNet_and_netToMaf -e
    err.hm.batchD2.chainNet_and_netToMaf -n 24 -q normal
    ./hm.batchD2.chainNet_and_netToMaf molitor_cleaned
```

Remove tandem assemblies with:

```
$ jobify -b -o out.hm.batchD3.remove_tandem_assemblies -e
    err.hm.batchD3.remove_tandem_assemblies -n 1 -q normal
    ./hm.batchD3.remove_tandem_assemblies molitor_cleaned
$ cd ..
```

Repeat D steps one more time.

2.4.1.6 D (round 2)

Prepare (and enter) the folder:

```
$ mkdir D_round2
$ cd D_round2
```

Copy all scripts with prefix `hm.batchD*` to the new working directory `D_round2`. The fasta file generated during step `D_round1` will now be used for `D_round2`. So,

```
$ ln -s D_round1/molitor_cleaned_D.fa.gz D_round2/molitor_cleaned.fa.gz
```

Edit the script suffixed with extension `initiation_and_all_lastz` and set the respective variables.

The `*.initiation_and_all_lastz` script is subsequently launched as a job:

```
$ jobify -b -o out.hm.batchD1.initiation_and_all_lastz -e
   err.hm.batchD1.initiation_and_all_lastz -n 36 -q normal --qos=h72
   ./hm.batchD1.initiation_and_all_lastz molitor_cleaned
```

Subsequently, set the variables of `*.chainNet_and_netToMaf` script and submit it as a job:

```
$ jobify -b -o out.hm.batchD2.chainNet_and_netToMaf -e
   err.hm.batchD2.chainNet_and_netToMaf -n 24 -q normal
   ./hm.batchD2.chainNet_and_netToMaf molitor_cleaned
```

Remove tandem assemblies with:

```
$ jobify -b -o out.hm.batchD3.remove_tandem_assemblies -e
   err.hm.batchD3.remove_tandem_assemblies -n 1 -q normal
   ./hm.batchD3.remove_tandem_assemblies molitor_cleaned
$ cd ..
```

2.5 Polish haploid genome (2 x Hapo-G)

Submit the 4 following scripts sequentially, waiting for the first job to finish before the next one starts. These scripts are components of the Hapo-G polisher pipeline.

```
$ ccc_msub /pathTo/scripts_for_Polishing_with_Hapo-G/bwa_pass_1.sh
$ ccc_msub /pathTo/scripts_for_Polishing_with_Hapo-G/Hapo-G_pass_1.sh
$ ccc_msub /pathTo/scripts_for_Polishing_with_Hapo-G/bwa_pass_2.sh
$ ccc_msub /pathTo/scripts_for_Polishing_with_Hapo-G/Hapo-G_pass_2.sh
$ cp /pathTo/Polishing/Hapo-G/Hapo-G_round_2/Hapo-G.fasta Haploid_Polished.fa
```

2.6 Scaffolding by using HiC data (SALSA)

Firstly, the relevant modules need to be loaded:

```
module load gcc/4.9.0
module load python/2.7
module load picard-tools/2.6.0
module load bwa
module load samtools
module load bedtools
```

Index fasta files with samtools and bwa:

```
$ jobify -p small -b samtools faidx Haploid_Polished.fa
$ jobify -p small -b bwa index Haploid_Polished.fa
```

Reads alignment and filtering can be done with Arima pipeline:

```
$ jobify -b -o out_parallel.arima -e err_parallel.arima -q xlarge -c 36
   ./pipeline_mapping_arima_Molitor_parallel.sh
```

Get metrics of scaffolds with the following commands:

```
$ module load exonerate
$ fastalength Salsa2_Scaffolds.fasta > Salsa2_Scaffolds.lengths
$ cut -f1 Salsa2_Scaffolds.lengths | sort -r  -n >
   Salsa2_Scaffolds.descending.lengths
$ head -15 Salsa2_Scaffolds.descending.lengths | awk '{split($0,a,","); sum +=
   a[1]} END {print sum}'
```

2.7 Rename and Filter Scaffolds

```
$ module load exonerate
```

Filter scaffolds by setting minimum scaffold size to the size of the shortest nanopore read used for the NECAT assembly.

minSize= 34991, so we set this value to 35kb

```
$ formatAssembly -c map_oldToNew_names_NEW_names.txt -gap 1 -agp -f
   Salsa2_Scaffolds.fasta -o Tenebrio_molitor_v1.fasta -len_limit 35000 -pfx
   Tenebrio_molitor_scaffold_ -v
$ assemblyMetrics Tenebrio_molitor_v1.fasta > Tenebrio_molitor_v1.stats
$ scaf2contigs -i Tenebrio_molitor_v1.fasta -o Tenebrio_molitor_v1_contigs.fasta
$ assemblyMetrics  Tenebrio_molitor_v1_contigs.fasta >
   Tenebrio_molitor_v1_contigs.stats
$ mv scaffolds.agp Tenebrio_molitor_v1_scaffolds.agp
```