Documentation

Main files:

models.py	The file contains: • grb_weighted_konig_dual_model • grb_restrained_konig_dual_model • grb_exact_model
solve.py	Solver with graph visualization & illustration. The file contains: • get_data • single_solve • full_solve • hybrid_single_solve • hybrid_full_solve • solMerge • clstsMerge • clsts_distance • distance_matrix • jaccard_merge • cplt_hierarchical_clstering (hamming) • evaluate • get_similar_rows_cols • illustrate • graph_result
mip_solve.py	The streamlined production version of the program with cmdline arguments parsers
parser.py	Snip parser for the ecoli data, nanopore_sorted.bam, only used for solve.py
parser_kmers.py	kmers parser for ecoli data nanopore_sorted.bam, only used for solve.py
parser_kminmers.py	kminmers parser for ecoli data nanopore_sorted.bam, only used for solve.py

Depreciated files:

exact_model.py	File contains the exact model and its manual test.
pulp_model.py	File contains the primal model, the dual model and its solver and different utility functions for the pulp version of the program.
__init__.py	Run the pulp_model.py here.
gen.py	Simple generation of artificial matrices for the pulp_model.py
graph.py	Create bipartite graph from matrix.
gurobi_model.py	File contains the Konig dual model written in gurobipy, its solver and utility functions
utilities.py	Some functions for illustration

Workspace initialization:

Minimal requirements

- `Python 3.9` (install from the [Python website](https://www.python.org/downloads/))

python>=3.9

The package

We will also need the python package manager `pip`. In a terminal :

- to see if it is installed :

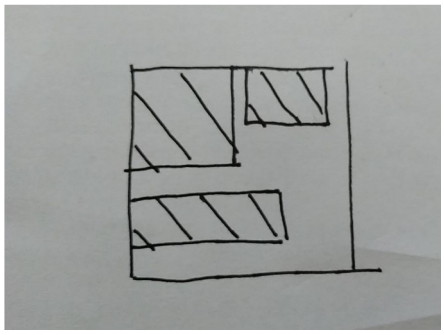
  ```bash which pip  # return the installation path

  pip --version  # if pip is installed, return its version

  ```

- to install it :

  ```bash sudo apt install python3-pip

We also need gurobi solver:

install from the gurobi website: https://www.gurobi.com/downloads/gurobi-optimizer-eula/

Other packages requirements:

You will find in the [requirements](./requirements/) folder the following files:

- [requirements_full.txt](./requirements/requirements_full.txt): the versions of all the necessary packages for all the scripts.

- [requirements-production.txt](./requirements/requirements_production.txt): the versions of the necessary packages for only the mip_solver.py.
  ```

Detailed application pipeline:

1. Read the matrix
2. Hybrid solve
    1. Solve with weighted_konig_model:
        1. Pool Solve:
            - Setting the poolsolve parameter with gurobipy to 2 (fixed in the script) to obtain 10 (by default) best solutions found from the solver.
        2. Solutions pool merging with Jaccard mesure:
            - From the solutions pool, count and sort the frequency of the rows and columns in each of the coordinates of the solutions.
            - Take the most frequent and make it the base solution.
            - Using Jaccard score to compare each of the individual solution in the solution pool and combining them to have 1 single final solution(cluster) if the Jaccard score is better then a specific threshold (crude, fixed at 0.7 in the script)
    2. Improve the cluster found by weighted Konig with the model tolerating error:
        1. If the quality of the cluster found from weighted konig is acceptable (internal pairwise hamming distance of its reads is better than a threshold,fixed 0.8 crude)
        2. Estimate a new matrix with new similar rows and columns on the coordinates of the result cluster using hamming distance
        3. Apply the model tolerating error on the new relaxed matrix
    3. Cut the cluster found from the matrix and create 2 new domains. Solve on the 2 new domains (horizontally for now)
    4. Stop when the matrix reaches a small enough dimension (parameters minRows minCols).
3. Complete hierarchical clustering (complete-linkage clustering):
    1. Cutting horizontally to distinct split the clusters found, (since 1 read can belong to multiple clusters).

    2. Calculating the proximity matrix (pairwise hamming distance on the mode of the rows):
        1. Calculate the mode of the 2 clusters for each column.
        2. Calculate the hamming distance between the mode.
    3. Find the pair of cluster with the smallest hamming distance value, combine them and update the proximity matrix
        - To update the distance between the new combined cluster with the rest of the clusters, we take the max distance of its elements to other clusters.
    4. Stop when reach we only have 1 big cluster or the specified estimation number of clusters.

Parser details:

Snips parser (parser.py):

- Find the informative position in the specified range
- Align all the reads range
- Select the informative position, remove rows and columns that are too ambiguous
- Transform into a matrix of (1,0,-1):
  - 1 is the most frequent nucleotide
  - 0 is the second most frequent nucleotide
  - -1 otherwise

Kmers parser:

- k between 10 and 15
- Classify the presence of kmers in the reads with 1 and 0.
- Classify repeating kmers by its relative position to the reference:
  - the distance between repeating kmers can alter the amount of informative positions for the matrix.
- Look for informative positions and generate a matrix.
  - The threshold could be based on the frequency of kmers (>= number_of_reads/3)

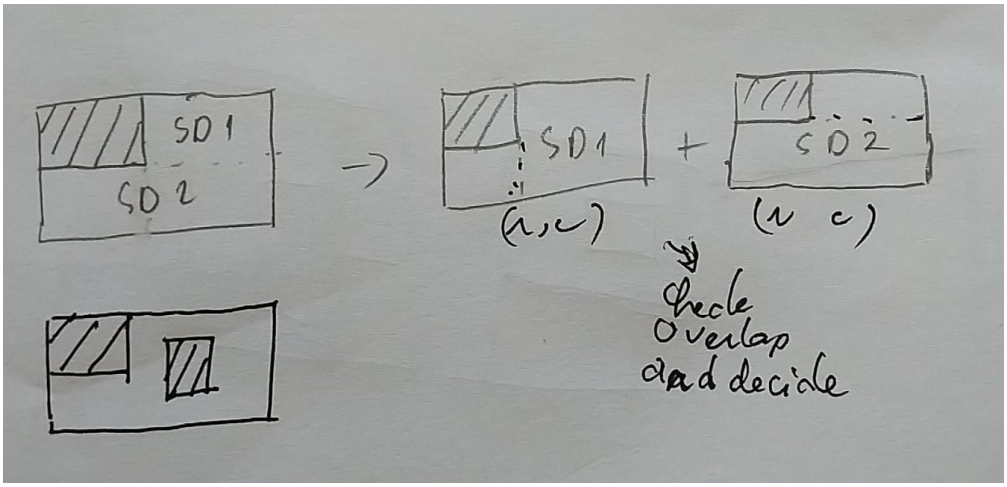
K-min-mers parser:

- Trying to reduce the amount of kmers by compressing them into k-min-mers.
- The reduction rate depends on different parameters.
  - The length of the l-mers (the letters of the new vocabulary set)
  - The hash function (0,N) N
  - The value of k-min
  - The density d (see https://pubmed.ncbi.nlm.nih.gov/34525345/)
- Overall heavily compressed all the reads but after limited testing, the compression present no improvement to the application.

Matrices generated from the snips parser are denser than matrices generated with kmers but with ambiguous positions.

Questions and remarks:

1. Divide and conquer strategy can be modified to be more general i.e. :



2.