PanPA

Release 1.0

Fawaz Dabbaghie

Jul 18, 2023

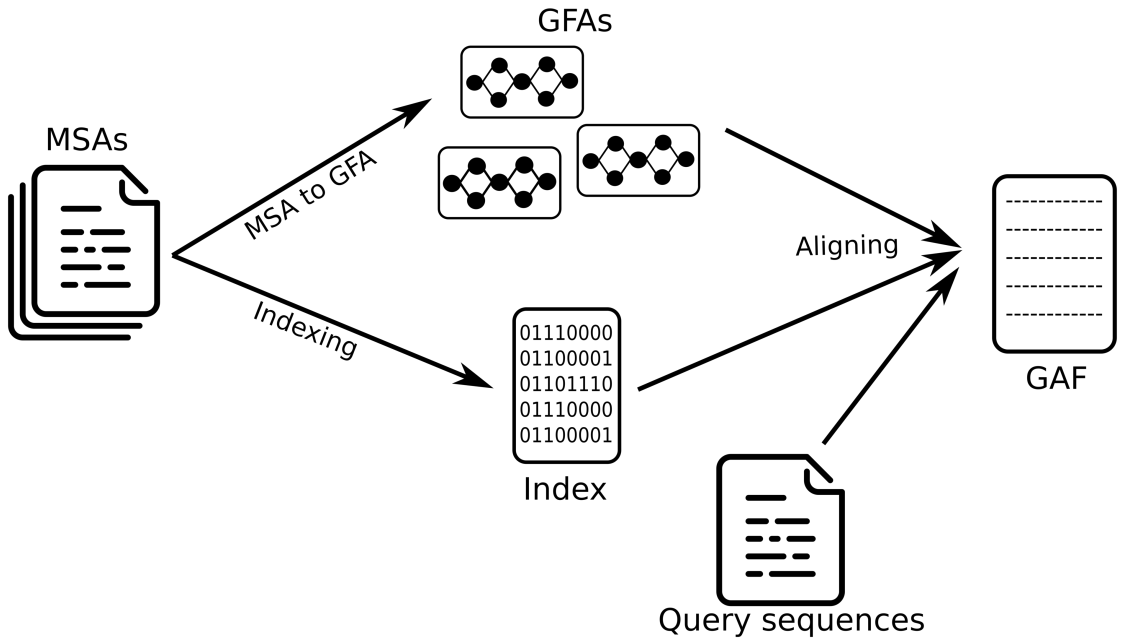
CONTENTS

PanPA is a command line tool written in `Cython` for building and alignments of panproteome graphs. The code base can be found Here.

The idea here is that given a set of MSAs of protein sequences (whether it is the same protein or a protein cluster), each MSA is turned into a Directed Acyclic Graph (DAG) in GFA format, indexes each MSA using *k*-mers or *(w, k)*-minimizers, and align DNA and amino acid sequences back to these graph using the index to find matches to the graph.

The alignment is done using Partial Order Alignment algorithm and the user can choose different substitution matrices and gap penalty score.

More on usage and commands can be found in *Subcommands*



CHAPTER

ONE

INSTALLATION

PanPA is easy to install through the `setup.py` script, the only requirement is `Cython` and `Python >= 3.6`. You can install PanPA locally with `python3 setup.py install --user` if you do not have root access to the operating system you're working on.

You can also use the `environment.yml` file to generate a conda or virtual Python environment and install PanPA there.

CHAPTER

TWO

CONTENTS

2.1 Subcommands

PanPA is separated into 3 subcommands that can be run independently:

1. Building GFA graphs from set of input MSAs with `build_gfa`
2. Indexing the set of MSAs which corresponds to the set of produced GFAs with `build_index`
3. Aligning query DNA or AA sequences back to the graphs produced with `align`

2.1.1 Building GFAs

In this step, PanPA can take as input a directory with MSAs in FASTA format, a text file with a list of input MSAs with each file in one line, or simply input MSAs as command arguments and outputs the corresponding graphs of each MSA in an output directory. The idea here is that there's a 1 to 1 correspondence between the MSAs and the graphs, so later on, the index built from the MSAs can be used to also match against the graphs.

Listing 1: build_gfa input arguments

```
usage: PanPA build_gfa [-h] [-f IN_FILES [IN_FILES ...]] [-l IN_LIST] [-d IN_DIR] [-c␣
↪CORES] [-o OUT_DIR]

optional arguments:
  -h, --help                    show this help message and exit
  -f IN_FILES [IN_FILES ...], --fasta_files IN_FILES [IN_FILES ...]
                                                Input MSA(s) in fasta format, one or␣
↪more file space-separated
  -l IN_LIST, --fasta_list IN_LIST
                                                a text file with all input MSAs paths␣
↪each on one new line
  -d IN_DIR, --in_dir IN_DIR
                                                Directory path containing one or more␣
↪amino acid MSA in FASTA format (gzipped allowed)
  -c CORES, --cores CORES
                                                Numbers of cores to use for aligning
  -o OUT_DIR, --out_dir OUT_DIR
                                                Output directory where the index files␣
↪and graphs from the MSAs are stored
```

2.1.2 Indexing MSAs

The subommands `build_index` takes the set of MSAs as input, and for each sequence in the MSA, seeds are extracted, the user can specify two types of seeds here, 1) *k*-mers or 2) *(w,k)*-minimizers using the argument `--seeding_alg` which takes either `k_mers` or `wk_min`, then the user can to specify the *k* size with `-k, --kmer_size` and w size with `-w, --window` or keep the default values. The user also needs to give an output file name/location.

The `--seed_limit` argument takes an integer, which specifies a limit to how many MSAs (or graphs) can one seed belong to. E.g. one *k*-mer can be present in all MSAs given, the user can specify a limit on that, and the matches are ordered based on how many times that seed was present in that MSA and the top `n` will be taken. If the user chooses to keep all hits, then `0` is given to this argument and all seed hits will be kept in the index, i.e. with `0`, all matching MSAs/Graphs will be used for alignments

Listing 2: build_index input arguments

```
usage: PanPA build_index [-h] [-f IN_FILES [IN_FILES ...]] [-l IN_LIST] [-d IN_DIR] [-o␣
↪OUT_INDEX]
                                                                    [--seeding_alg SEEDING_
↪ALG] [-k K-MER] [-w WINDOW] [--seed_limit SEED_LIMIT]

optional arguments:
  -h, --help                    show this help message and exit
  -f IN_FILES [IN_FILES ...], --fasta_files IN_FILES [IN_FILES ...]
                                                Input MSA(s) in fasta format, one or␣
↪more file space-separated
  -l IN_LIST, --fasta_list IN_LIST
                                           a text file with all input MSAs paths each on␣
↪one new line
  -d IN_DIR, --in_dir IN_DIR
                                                Directory path containing one or more␣
↪amino acid MSA in FASTA format (gzipped allowed)
  -o OUT_INDEX, --out_index OUT_INDEX
                                                The output index file name
  --seeding_alg SEEDING_ALG
                                                Seeding algorithm. Choices: k_mers, wk_
↪min. Default: k_mers
  -k K-MER, --kmer_size K-MER
                                                K-mer size for indexing the sequencing.␣
↪Default: 5
  -w WINDOW, --window WINDOW
                                                Window size when using w,k-minimizers␣
↪instead of k-mers for indexing. Default:8
  --seed_limit SEED_LIMIT
                                                Indicates how many graphs can a seed␣
↪belong to. Default: 5, give 0 for no limit
```

2.1.3 Align Query Sequences

For aligning query sequences to the graphs, you need to give three main inputs to the subcommand `align`:

1- The index that was built with `build_index` subcommand 2- The input graphs that were built from the same set of MSAs that were used for building the index, the set of grpahs which can be a directory, a text file with list, or given directly in the command. 3- The query sequences in FASTA. If DNA sequences are given, then the user needs to use the flag `--dna` which will then run the frameshift aware alignment algorithm on both the forward and reverse complement of each DNA query sequence.

The user can also specify the substitution matrix to use for the alignment, or print a list of possible matrices with `--sub_matrix_list`. The user can also specify a certain gap score with `--gap_score`, a cutoff on alignment id with `--min_id_score`, and can set a limit to how many graphs to align to with `--seed_limit`. This step can be made faster by giving more cores. If DNA sequences were aligned to graphs that were build from DNA sequences, then please read about this in the Other Info section.

The output alignment are in GAF format. To learn more about this format please check here, moreover, the Other Info section has some extra information about the output file.

Listing 3: align input arguments

```
usage: PanPA align [-h] [-g IN_FILES [IN_FILES ...]] [-l IN_LIST] [-d GRAPHS] [--index␣
↪INDEX]
                                   [-r SEQS] [--dna] [-c CORES] [--sub_matrix SUB_
↪MATRIX] [--sub_matrix_list]
                                   [-o GAF] [--gap_score GAP_SCORE] [--min_id_score MIN_
↪ID_SCORE]
                                   [--seed_limit SEED_LIMIT]

options:
  -h, --help                     show this help message and exit
  -g IN_FILES [IN_FILES ...], --gfa_files IN_FILES [IN_FILES ...]
                                                    Input GFA graphs, one or more file space-
↪separated
  -l IN_LIST, --gfa_list IN_LIST
                                                    a text file with all input graphs paths␣
↪each on one new line
  -d GRAPHS, --in_dir GRAPHS
                                                    Path to directory with GFA files
  --index INDEX          Path to pickled index file generated in the build step
  -r SEQS, --seqs SEQS  The input sequences to align in fasta format
  --dna                          Give this flag if the query sequences are DNA and not AA
  -c CORES, --cores CORES
                                                    Numbers of cores to use for aligning
  --sub_matrix SUB_MATRIX
                                                    Substitution matrix to use for alignment,
↪ default: blosum62
  --sub_matrix_list      When given, a list of possible substitution matrices will be␣
↪given
  -o GAF, --out_gaf GAF
                                                    Output alignments file path
  --gap_score GAP_SCORE
                                                    The gap score to use for the alignment,␣
↪default: -3
  --min_id_score MIN_ID_SCORE
```

(continues on next page)

(continued from previous page)

```
                                                    minimum alignment identity score for the␣
↪alignment to be outputted, [0,1]
  --seed_limit SEED_LIMIT
                                                    How many graphs can each seed from the␣
↪query sequence have hits to,
                                                    default: 3
```

2.1.4 Align to Single Graph

The user can avoid the extracting seeds step when aligning sequences if the user wants to align to a specific target graph by using the `align_single` subcommand.

In this subcommand, the user needs to provide a target graph in GFA format, the graph has to be a DAG (directed and acyclic), and query sequences in FASTA format.

Listing 4: align to single target

```
usage: PanPA align_single [-h] [-g IN_GRAPH] [-r SEQS] [--dna] [-c CORES] [--sub_matrix␣
↪SUB_MATRIX]
                                                      [--sub_matrix_list] [-o GAF] [--gap_
↪score GAP_SCORE]
                                                      [--min_id_score MIN_ID_SCORE]

options:
  -h, --help                      show this help message and exit
  -g IN_GRAPH, --gfa_files IN_GRAPH
                                                    Input GFA graph to align against
  -r SEQS, --seqs SEQS  The input sequences to align in fasta format
  --dna                           Give this flag if the query sequences are DNA and not AA
  -c CORES, --cores CORES
                                                    Numbers of cores to use for aligning
  --sub_matrix SUB_MATRIX
                                                    Substitution matrix to use for alignment,
↪ default: blosum62
  --sub_matrix_list      When given, a list of possible substitution matrices will be␣
↪given
  -o GAF, --out_gaf GAF
                                                    Output alignments file path
  --gap_score GAP_SCORE
                                                    The gap score to use for the alignment,␣
↪default: -3
  --min_id_score MIN_ID_SCORE
                                                    minimum alignment identity score for the␣
↪alignment to be outputted, [0,1]
```

2.2 Full Experiment

Here we describe a full working example of how to use **PanPA** to generate a panproteome of some assemblies and perform alignments with query sequences against this panproteome. We will collect annotations from NCBI of 10 E. coli assemblies. The general steps are:

1. Downloading annotations of the example assemblies
2. Separating the downloaded into two groups, one for building the Panproteome and one for alignment (leave one out)
3. Generating protein clusters
4. Generating MSAs from the protein clusters
5. Generating graphs in GFA format from the MSAs
6. Indexing the set of MSAs/GFAs
7. Aligning the left-out samples back to the generated panproteome

2.2.1 Requirements

For this example, you need to install `PanPA`, `mmseqs` for clustering, and some MSA software like `clustalo` for example. You can get `mmseqs` using conda, brew, docker, or simply downloading the precompiled version with `wget https://mmseqs.com/latest/mmseqs-linux-avx2.tar.gz; tar xvfz mmseqs-linux-avx2.tar.gz`

2.2.2 Data

In this example, we will be using 10 E. coli assemblies/annotations randomly selected from RefSeq. The list of ftp links are listed in `ftp_links.txt` Here

Accession	FTP Link
GCF_000002515.2	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/002/515/GCF_000002515.2_ASM251v1
GCF_000002725.2	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/002/725/GCF_000002725.2_ASM272v2
GCF_000002985.6	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/002/985/GCF_000002985.6_WBcel235
GCF_000005825.2	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/825/GCF_000005825.2_ASM582v2
GCF_000005845.2	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2
GCF_000006605.1	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/006/605/GCF_000006605.1_ASM660v1
GCF_000006625.1	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/006/625/GCF_000006625.1_ASM662v1
GCF_000006645.1	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/006/645/GCF_000006645.1_ASM664v1
GCF_000006725.1	https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/006/725/GCF_000006725.1_ASM672v1

2.2.3 Step 1: Download annotations

To download the annotations using the FTP links from RefSeq with the download script

```
$ bash download_proteins.sh ftp_links.txt
```

This will download 10 the proteins FASTA file for each assembly.

2.2.4 Step 2: Separating into groups

We can use 9 of these assemblies to generate the protein clusters, hence graphs and use the last 1 to align back to the grpahs generated. Therefore, we can mix all the proteins form 9 of these assemblies to generate the clusters and leave one out for the alignment.

- Let's keep one of these FASTA files for the alignments later, this one was chosen randomly

```
$ gzip -cd GCF_000006625.1_ASM662v1_protein.faa.gz > GCF_000006625.1_ASM662v1_protein.
↪fasta && rm GCF_000006625.1_ASM662v1_protein.faa.gz
```

- We can now merge all sequences from the other 9 into one FASTA file

```
$ for f in *faa.gz;do gzip -cd $f >> all_proteins.fasta && rm $f;done
```

- You can use `fasta_fastq_statistics.sh` to calculate simple statistics on any FASTA or FASTQ file. However, it only accepts files where each sequence is contained in one line. Therefore, we can use this one-liner to remove the new lines in the sequence

```
$ awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);}  END {printf("\n");}' < all_
↪proteins.fasta | sed '/^$/d' > tmp && mv tmp all_proteins.fasta

$ bash scripts/fasta_fastq_statistics.sh all_proteins.fasta
61979 reads
27500039 total length
443.699 average read length
```

2.2.5 Step 3: Generating clusters with mmseqs

Now that we have all proteins from the 9 assemblies, we can cluster them using `mmseqs`. The parameters chosen here are just an example, but this of course can be changed.

```
$ ./mmseqs easy-linclust all_proteins.fasta all_proteins_cluster tmp --min-seq-id 0.4
$ rm -r tmp/
```

After running `mmseqs`, we get several outputs, a table with cluster names and sequences in the cluster `all_proteins_cluster_cluster.tsv`, a FASTA file with the representetive sequences `all_proteins_cluster_rep_seq.fasta`, and a FASTA file with all sequences `all_proteins_cluster_all_seqs.fasta`.

We need each cluster to be in a separate FASTA file, you can then use `scripts/extract_clusters.py` which a simple Python script that takes a simple txt file with sequences names and the FASTA file with all sequences and an output directory, and it outputs the sequences of each cluster in a separate FASTA file:

```
$ cut -f1 all_proteins_cluster_cluster.tsv | uniq > cluster_names.txt
$ mkdir clusters
$ python3 scripts/extract_clusters.py cluster_names.txt all_proteins_cluster_all_seqs.
↪fasta clusters/
```

2.2.6 Step 4: Generating MSAs from clusters

This, of course, can be done using many different MSA tools, for this tutorial we used `clustalo`, where we first move all clusters that contain one sequence because there's nothing to do, then we run `clustalo` on each clusters to generate an MSA.

```
$ python3 /scripts/alignment_validation/move_1seq_file_to_msa.py clusters msas
$ for f ``ls -1 clusters/``;do ./clustalo --in clusters/$f > msas/$f;done
```

This will take some time to run as there are many clusters.

2.2.7 Step 5: MSA to GFA

Now that we have many MSAs, we can use `PanPA` to generate a graph for each MSA.

```
$ mkdir graphs
$ PanPA build_gfa -d msas/ -c 4 -o graphs
```

The `build_index` subcommand can take several cores and run in parallel, here we gave it 4 cores, and finished converting all clusters to graphs in about 2 minutes on a standard laptop.

2.2.8 Step 6: Indexing

We need to also index the MSAs where we use the index to guide the alignment to which graphs to align to as we have a 1 to 1 equivalency between an MSA and a GFA, if a seed points to e.g. MSA1 then we align to GFA1. The user can choose several parameters for indexing and can increase or decrease the seed size depending on the data used.

```
$ PanPA build_index -d msas/ --seeding_alg wk_min -k 5 -w 3 --seed_limit 0 -o index_k5_
↪w3_no_limit.index
```

This step takes a bit more tan 1 minute

2.2.9 Step 7: Aligning

Finally, we have generated graphs and an index, we can give both of these to the `align` subcommand in `PanPA` and some query sequences to do the alignments.

```
$ PanPA align -d graphs/ --index index_k5_w3_no_limit.index -r GCF_000006625.1_ASM662v1_
↪protein.fasta --min_id_score 0.5 --seed_limit 30 -c 4 -o GCF_000006625.1_aligned.gaf
```

This subcommand can also take several cores which makes alignment faster. For these parameters the aligment was done in about a minute.

2.3 Other Info

Some extra information about using **PanPA** and the output GAF format.

2.3.1 GAF Format

The GAF (Graph Alignment Format) was described here here. For **PanPA**, a couple more tags were added to keep all important information related to the alignments.

When aligning amino acid sequences against amino acid graphs, the tag `gid` gives the target graph where the query sequenced aligned.

when aligning DNA sequences using the argument `--dna`, where the frameshift aware alignment algorithm is used, another tag is added `DNA` which has two values, froward and reverse, indicating if the DNA query sequence was aligned in the forward or reverse complement direction.

2.3.2 DNA to DNA

PanPA was mainly designed to align amino acid and DNA sequences against amino acid graphs. It is possible though to also generate the index and graphs from MSA sequences in DNA alphabet. However, when aligning DNA query sequences back to these graphs, you do not need to use the `--dna` argument, as this argument tries to align DNA against amino acids and uses the frameshift aware alignment algorithm. However, you just need to omit the `--dna` argument, and use `--sub_matrix dna` which is basically edit distance (1 for a match, 0 for mismatch). Therefore, you might want to also change the gap penalty with `--gap_score` argument. In this mode however, **PanPA** will not try to also align the reverse complement, so the user might want to provide this in the query sequence FASTA file.

For example `PanPA align -d graphs_from_dna_seqs/ -r query_dna_sequences.fasta -o query_dna_sequences.gaf --sub_matrix dna --gap_score -1 -c 10`