GArNet Manual (ver. 250210)

Purpose and Overview

This Python script can optimize mutation combination by adopting the following calculation steps: running short generation of virtual evolution by GAOptimizer and selection of mutations by summation of the results based on network theory. The mutation candidates were selected from the libraries containing homologs of sequence of target protein. Currently, three selection pressure (HiSol, REU and both of the scores) can be used for the optimization (20250210). Key steps include:

1. **Short runs of GAOptimizer to generate mutation networks**:
   - Generate multiple sequence alignments (MSAs) from a template protein (STP) and sequence library (>1).
   - Introduce mutations (random or recombination-based) to create child generations.
   - Score the variants using Rosetta Energy Units (REU) or a hydrophobicity-based HiSol score.
   - Total *n* generation of virtual evolution was performed by GAOptimizer; the n value can be defined by users. After the evolution, mutation candidates were selected by comparing sequences between STP and the generated elite structure. The selected mutations and co-occurrence of the mutations were represented as a network.
   - The above steps were repeatedly performed (total *m*; the value can be defined by users).
2. **Selection of mutation candidates based on network theory**:
   - Construct a mutation network by summation of the networks generated.
   - Selection of the mutations by analyzing the mutation network based on centrality.
   - Generate a list of mutation candidates and mutated structure.

Required Python Libraries and Dependencies

- Python 3.11.7
- **Pyrosetta4**: Rosetta's Python interface (requires a Rosetta license and installation).
- **BioPython (version 1.83)** (Bio): For parsing sequence alignments.
- **Numpy (version 1.26.4)**: For numerical computations.

- **Networkx (version 3.1)**: For network analysis.
- **Matplotlib (version 3.8)**: For network visualization.
- **MAFFT (version 7.525)** (external tool, must be installed on the system for sequence alignment).
- **Scipy (version 1.11.4)**, if not included with numpy/matplotlib distributions, may be required for polyfit (linear regression in network analysis steps).
- **CPU:** Intel® CoreTM i5-12400 (Max. 4.40 GHz)
- **OS:** Rocky Linux 9.3

Input Data

1. **Template Protein (STP)**:
   A PDB file containing at least one chain that will serve as the template.
2. **Sequence Library**:
   A directory containing FASTA files of homologous sequences. The script will randomly select files in the library to generate MSAs and calculate amino acid frequency distributions.
3. **Command-line Arguments**:
   - -PDB <pdb_file>: Path to the template protein's PDB file.
   - -DIR <library_directory>: Path to the directory with sequence library files.
   - -CHAIN <chain_name>: Chain identifier in the PDB to focus on.
   - -GENNUM <integer>: Number of generations by GAOptimizer. This corresponds with the *n* value.
   - -RPTNUM <integer>: Number of times the entire optimization cycle is repeated. This would be a basement of the *m* value.
   - -REU and/or -HISOL: Fitness functions. At least one required.
     - -REU: Use Rosetta Energy Units for scoring.
     - -HISOL: Use the hydrophobicity-based HiSol score for scoring.
   - -MUTNUM <value>: Number of mutations (absolute number or a percentage, e.g., 10, 30per). This corresponds with the "*k*" value.
   - Optional flags:
     - -OUTPUT <output_file>: The path of output log filename (default is output.log).
     - -UNCHANGECUT: Remove mutations that do not change the original residue in network analysis.

        - -SKIPGAOptimizer: Skip the GA part if already done.
        - -SKIPNET: Skip the process to generate the mutation network if already done.
    - Not recommended options (legacy):
        - -RANK or -SELECT with parameters like power degree|count|centrality, etc.
          The recommended usage is to rely on centrality-based (-RANK centrality between and -SELECT power centrality between below) methods.

Running the Script: Example Command

python GArNet.py -PDB my_template.pdb -DIR my_library -CHAIN A -GENNUM 50 -RPTNUM 2 -REU -MUTNUM 10

This command:

- Uses my_template.pdb as the STP and selects chain A.
- Uses the my_library directory for sequences.
- Runs for 50 generations (GAOptimizer step) repeated twice (RPTNUM).
- Uses REU as the fitness score.
- Introduces about 10 mutations.
- Outputs logs to output.log by default.

Output Data

1. **Intermediate Files and Directories**:
    - tmp_mafft.inp, tmp_mafft.out: Input and output for MAFFT alignments.
    - pdb_score_<tag>.fasta: This file includes the scores of the generated mutant.
    - tmp_intmsa.out: The INTMSA formatted amino acid frequency matrix.
    - Multiple directories are created:
        - temporary_pyrosetta, temporary_elite, temporary_selected or storing intermediate PDBs and data from each generation.

        - temporary_sample_mutations, temporary_mutataions, temporary_previous_mutation: mutations adopted by the program were saved in these directory.

2. **Log Files**:
    - output.log: Main log file. Records generation-wise scores, average score for the 30 parent structures, and scores for the elite structure.
    - program_time.fasta: Records execution time for various steps and total runtime.
3. **Mutation and Network Files**:
    - mutations_order.fasta: Records the order and frequency of each adopted mutation.
    - tmp_mutations_list.fasta: Temporary listing of mutation sets.
    - edges_all_count.fasta: Edge adoption frequencies for the mutation network.
    - sub_mutations_list.fasta: Intermediate file tracking additional/reduced mutations in final selection steps.
4. **Final Selection Files**:
    - select_mutations_<tag>.fasta: Selected mutations after network-based optimization.
    - best_select_mutations_<tag>.fasta: The best final chosen mutation set.
    - GAN-<tag>_ORIGINAL.pdb and GAN-<tag>_*_BEST.pdb: Generated PDB files of the final selected mutants. The _BEST.pdb file denotes the top-performing variant. These pdb data will be output in the directory named “temporary_final_mutate_<tag>”.
5. **Graph and Network Analysis Outputs**:
    - network_output.py: Automatically generated Python script to draw and analyze the mutation network.
    - Running network_output.py creates a plotted network (display can be uncommented for visualization).
    - extract_mutations_<tag>.py: This script performs graph drawing and selection of mutations.

Recommended Usage

- Use either -REU or -HISOL (or both) as your fitness functions.
- Avoid legacy options (marked as "Not recommended"). Stick to centrality-based selection methods.
- Ensure MAFFT is installed and accessible.
- If you rerun the script with different parameters, consider cleaning up old temporary directories.

Troubleshooting

- If input parameters are missing, the script will print an error and exit.
- If no fitness function is provided, the script will fail.
- Check dependencies and ensure pyrosetta is correctly installed and licensed.