ANUBI USER GUIDE

This software is designed to optimize protein or peptide candidates for improved target binding. This document provides a detailed explanation of the software installation, usage, and parameter settings to help users better understand how to operate it.

DOWNLOADS

The dependencies are below:

- **GROMACS** (GPU version)
- **gmx_MMPBSA**: gmx_MMPBSA_installation website (if you have MPI error: gmx_mpi and mpirun(MMPBSA) conflict, check your MMPBSA version.)
- **Modeller**: modeller_installation website(**You need licence for that**.)
- **Other dependencies**: Pandas, Numpy, pyyaml, biopython if you don't have them, please use conda to install (or other ways you prefer)

For your reference, the peptide example in our paper was generated using the following software versions: GROMACS 2023.2 (MPI, CUDA), gmx_MMPBSA v1.6.3 based on MMPBSA version 16.0 and AmberTools 20, Modeller 10.6, and Python 3.13.2. Other comparable versions are also acceptable.

For detailed steps:

1. Clone the repository or you can just download the zip through GitHub

   ```
   git clone https://github.com/ZontaLab/ANUBI.git
   ```

2. Find the folder

   ```
   cd ANUBI
   ```

3. Create environment and install all packages

   ```
   conda create -n ANUBI
   ```

   ```
   conda activate ANUBI
   ```

conda config --add channels salilab

conda install modeller

conda install pandas numpy pyyaml biopython

4. Open the *infile.yaml* and edit it, see **PARAMETERS** part to check the details.
5. Run the main script

   python ANUBI_main.py -i infile.yaml

PARAMETERS

For the command "python ANUBI_main.py -i infile.yaml", the only input parameter is the *infile.yaml*, you need to specify the correct path and name.

For the parameters in *infile.yaml*:

Input_files:

structure_file_path:

your **complex** pdb file. It must be **equilibrated complex structure** (protein-protien/protein-peptide), and please use **AMBER style**. Before you put it in the software, please make sure the **receptor firstly and then your protein/peptide (the protein/peptide you want to design)**.

pipeline_mode:

Two modes (**protein mode** and **peptide mode**), for protein mode, we only do **point mutation** in the '***res_pos_list'*** region. For peptide mode, we do point mutation for the entire peptide and after every '***peptide_mode_frequency_control'*** times of point mutation, we randomly choose on from {Add one ALA at the N-terminus; Add one ALA at the C-terminus, remove one at the N-terminus; remove one at the C-terminus}

Basic_settings:

conda_activate_script_path:

the path for command 'conda activate', you can get it by: which activate

GROMACS_executable_path:

The path to run MD simulations in GROMACS (run gmx/gmx_mpi)

conda_gmx_MMPBSA_name:

the environment name for your gmx_MMPBSA

conda_Modeller_name:

the enviroment name for Modeller

gmx_mmpbsa:

receptorFRAG:

How many chains in RECEPTOR (FIRST molecule in the starting pdb file)

ABchains:

How many chains in your deigned PROTEIN/PEPTIDE (SECOND molecule in the starting pdb file)

startingFrameGMXPBSA:

The MD simulation used for the MMPBSA calculation is 5 ns. The value 2000 ps here means that we extract the trajectory starting from 2000 ps, and only use the portion of the trajectory after 2 ns for the MMPBSA calculation. So, **you do not need to change this parameter**.

modeller:

res_pos_list:

Only when you choose **protein mode**, you need to set that. This indicates **the region you intend to design**. For example, '105:B 106:B 107:B 108:B 109:B' means that you are designing residues 105–109 on chain B.

max_mutant:

This specifies the **maximum number of mutants** you want to generate. For example, if you set it to 30, the pipeline will **stop** after finishing the 30th mutant. In practice, we typically use **25–30**, but you can **adjust this based on your needs and your system**.

cycle_num:

We run 10 MD simulations and perform an MMPBSA calculation for each one, which means the **10 MD simulations and MMPBSA calculations are run in parallel**. We then take the **average** of the 10 results as the **final energy value**. In general, it is **not recommended to modify this parameter** unless you have specific requirements.

run:

num_processors:

Number of CPU cores for MMPBSA parallel computation, check your own system.

Metropolis_Temperature:

Metropolis_Temperature = 1/ β. $P = \min\{1, e^{-\beta\Delta\Delta G}\}$, β must be a positive value. When ΔΔG<0, P=1, meaning the candidate is always accepted, and therefore the magnitude of β does not matter in this case. When ΔΔG>0, a larger β makes −βΔΔG more negative, which makes P smaller. We then draw a random number (random.uniform(0, 1)), If randNUM < P, the candidate is accepted; otherwise, it is rejected. Therefore, when ΔΔG>0, increasing β decreases the acceptance probability of unfavourable candidates. **In other words, a higher Metropolis temperature leads to a greater acceptance probability for unfavourable candidates.**

peptide_mode_length_min:

For peptide mode, we have choice to change the length of the peptide. The **minimum length** is determined by your specific system and needs. This setting is only required when you choose the peptide mode.

peptide_mode_length_max:

For peptide mode, the **maximum length** is determined by your specific system and needs. This setting is only required when you choose the peptide mode.

peptide_mode_frequency:

For peptide mode, every N (*peptide_mode_frequency*) steps, we randomly select a strategy that can change the peptide length {Add one ALA at the N-terminus; Add one ALA at the C-terminus, remove one at the N-terminus; remove one at the C-terminus}. This setting is only required when you choose the peptide mode.