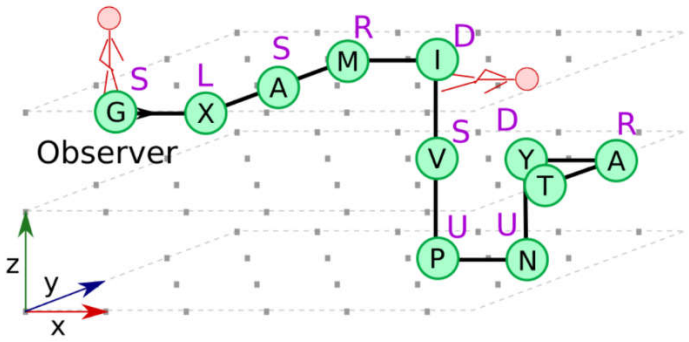
YMIR - Documentation

Overview

YMIR uses a structural representation of amino-acid (AA) sequences to compute an affinity between proteins with physiological properties.

A protein is defined by: 1/ a sequence of AAs, and 2/ its structure. Structures are consecutive positions in a cubic lattice without self-colliding, written as a list of 90 degrees moves in 3D: Up(U), Down(D), Left(L), Right(R), Straight(S).



Example of a protein with AA sequence 'GXAMIVPNYAT', starting position (x=1,y=0,z=2) and structure 'SLSRDSUUDR'. Note: the first move is defined from the axis (S follows x, L follows Y, U follows z), and could also be Backwards (B). The next moves are relative to the previous direction, like a plane would do (see the 'observer'), and can not be backwards.

YMIR computes the **affinity of a receptor** from its sequence (but unknown structure) **towards a predefined epitope** with known sequence and structure. It enumerates all possible structures the receptor could have around the epitope. Then, it uses a matrix of AA-AA interaction strength[1] to calculate a binding energy for each structure. The resulting affinity can be given as the best binding energy or as the statistical binding energy according to a boltzmann distribution. Note, the bolzmann probability of a receptor structure includes its self binding energy by self-contacts.

Inputs:

- **E** Epitope: AA sequence, starting position, structure
- **R** AA Sequence of the receptor
- **n** Minimum number of contacts the receptor should have with the epitope. Discarding receptor structures with too few contacts makes it faster.
- **kT** Temperature (for statistical affinity)
- **L** Optionally, a list of unavailable positions for the receptor (like glycans or unaccessible parts of the epitope)

Outputs:

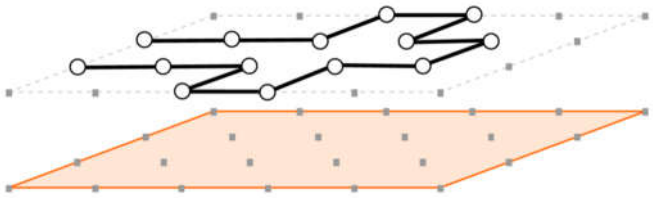
- the best binding affinity of all structures
- the statistical affinity of all structures, with boltzmann distribution of temperature T
- all possible structures of the receptor, with their binding affinity

Outputs that I could implement:

- for each epitope AA, the probability to be bound by a receptor with this sequence
- for each epitope AA, the number/list of receptor of that sequence that bind it

Example of use:

You want the affinity of the receptor sequence 'AGNALIVN' of size 8, to of a stupid and simple epitope, only filled with 'A' (sequence 'AAAAAAAAAAAAA'), and structure 'SSRLLRLLRLLRS', shown below.



Step 1: For the requested receptor sizes, pre-calculate all structures around an epitope (can take a few hours). This is done by creating a new `affinityOneLigand`:

```
affinityOneLigand T = affinityOneLigand(
    "SSRLLRLLRLLRS",                  // E. Structure
    "AAAAAAAAAAAAA",                  // E. AA sequence
    lattice::idFromPosisition(32,32,32),  // E. Starting position*
    8,                                // Receptor size
    4,                                // n. Min number of contacts
    0,                                // (optional) Min of self-contacts of the receptor
    0.4);                             // kT. Temperature
                                      // [*] x,y,z in [0..63]. Use -1 for center of lattice.
```

Step 2: Request the affinity. First element is the best affinity, second is the statistical one.

```
cout << "Best:"        << T.affinity("AGNALIVN").first;
cout <<  "Statistical:" << T.affinity("AGNALIVN").second;
```

Additions:

- ***Every computation of Step 1 is automatically saved*** *into a text file. The name follows the pattern: epitope structure + size receptors + minima nb contacts + 'Compact.txt'. For example, for the present case: SSRLLRLLRLLRS8-4Compact.txt. If the same calculation is performed again, the result is read from the text file instead of recalculated. Further, each call of step 2 is stored in memory, so inside one run of the program, calling again the same sequence becomes faster.*

- *Handling of* ***forbidden points****: By default, a flat square (x,y) of size 10x10 centered from the middle of the lattice are forbidden access to receptors (like in the picture). It is possible to add an argument to step1 at the end, with the list of forbidden positions in the lattice. It can be used to model shielded positions by glycans for instance. To have no forbidden position, give '-1' as forbidden position (use 'vector<int>(1,-1)' as additional argument).*

- *To retrieve the* ***list of structures*** *and their affinity, add 'true' as an option inside step 2: call* 'T.affinity("AGNALIVN", true)'.

- *To* ***visualize the possible structures*** *around an epitope, you need to comment '#define DISABLE_GRAPHICS' inside plot3d.h, and you can put manually your parameters (epitope, size receptors) inside testBigReceptorLigand() located in receptorligand.cpp and call this funtion from the main() function. In that case, the C++ code will need the glut library sudo apt-get install freeglut3-dev in linux.*