LatSeF - Manual

Martin Mann - University Freiburg

http://www.bioinf.uni-freiburg.de/

LatPack Tools Package
Version 1.6.4

1 Description

LatSeF implements a chain growth algorithm to simulate Sequential Folding of lattice proteins. It supports:

1. various lattices (see Sec. 3)
2. arbitrary energy functions (see Sec. 4)
3. side chain models
4. PDB (Protein Data Bank) output of the best structure found
5. check for extensibility of last appended monomer
6. symmetry exclusion via move string normalization
7. extension of all structures within a given energy interval above the minimal energy reachable so far

2 Method

LatSeF implements a greedy heuristic of a chain-growth approach. The monomers are placed successively on lattice positions such that the structure forms a selfavoiding walk. For each length all possible structure extensions with one monomer are generated and evaluated. The best structures are considered in the next extension iteration. The process is described in more detail in Sec. 2.1

Due to the lattice restrictions and the selfavoidingness constraint, the procedure may end in non-extensible structures during the iteration and fail. A fast way to overcome this problem is to check the extensibility of the last monomer after its placement. Only extensible structures are considered further and evaluated later. A cheap form of extensibility check is presented in Sec. 2.2.

A description what kind of energy functions are supported is given in Sec. 4. Furthermore, the text file format for the different function types is specified to allow for arbitrary energy functions by the user.

2.1 Chain Growth Algorithm - Backbone Model

Given:

$S = S_1, \ldots, S_n$	:	monomer sequence from alphabet A to fold
$E(S, P)$	:	energy function (see Sec. 4)
N	:	the neighboring vectors of the lattice model to use
ΔE	:	energy interval above the minimal energy for this iteration that are going to be extended in the next

Result:

$L = L_1, \ldots, L_n$	:	3D coordinates of the energetically best sequential placement of S in the lattice

Method:

The approximation follows a greedy structure-elongating chain-growth approach:

1: $B \leftarrow \{L_1 = (0,0,0)\}$ ▷ best structures of last iteration
▷ initialized by placing the first monomer to $(0,0,0)$
2: $C \leftarrow \emptyset$ ▷ structures generated in current iteration
3: **for** $i = 2 \ldots n$ **do**
4: **for all** $L \in B$ **do** ▷ L has length $(i-1)$
5: **for all** $\vec{v} \in N$ **do**
6: **if** $L_{(i-1)} + \vec{v} \notin \{L_1, \ldots, L_{(i-1)}\}$ **then** ▷ selfavoidingness
7: $L' \leftarrow (L_1, \ldots, L_{(i-1)}, L_{(i-1)} + \vec{v})$
8: $C \leftarrow C \cup \{\textsc{Normalize}(L')\}$ ▷ store normalized extension
9: **end if**
10: **end for**
11: **end for**
12: $minE \leftarrow$ minimal energy of all elements in C
13: $B \leftarrow \{c \mid c \in C \text{ and } E(S_{1\ldots i}, c) \leq (minE + \Delta E)\}$ ▷ copy all best
14: $C \leftarrow \emptyset$ ▷ reset structure storage
15: **end for**
16: report best placement $L \in B$ with minimal energy $E(S, L)$

Note: Due to the greedy storing of the energetically best structures only, it may occure that none of the structures in B can be extended in a selfavoiding way (Line 6 gives 'false'). Therefore, C, and in the following B, would get empty and the procedure stops without finding a selfavoiding walk of the whole sequence S. This problem can usually be solved either by increasing ΔE or by adding an extensibility check in line 6. Both results in additional computations and an increased runtime.

2.2 Extensibility Check

A first and cheap variant of extensibility check is to verify that the position of the last appended monomer has at least one free neighbored. This ensures that at least one additional monomer can be added.

But, this is *not* enough to *ensure* extensibility. Still it might be the case that the current placement of the last monomer only allows a limited extension of the chain, not sufficient to place all remaining monomers.

Nevertheless, this check seems to be sufficient in general and the described scenario may only occure for long sequence lengths. The minimal length that makes such a placement possible strongly depends on the lattice model and the size of the corresponding neighborhood, i.e. number of neighboring vectors. In the square lattice, the minimal length is 10. The corresponding structure is given in Fig. 1. The last appended (red) monomer has a free neighbored position (yellow), but this position has none and will not be extensible. This situation cannot be determined using the simple check. Here, a full extensibility check that validates if there is a free path out of the structure has to be done.

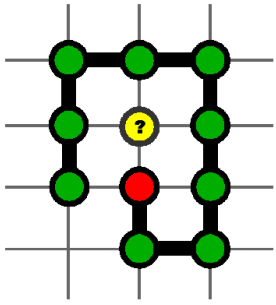

Figure 1: Non-extensible structure in the 2D-square lattice that is not rejected by the simple extension check. Here the last position (red) can be extended (yellow position). But afterwards no extension will be possible any more.

Given:

$L = L_1, \ldots, L_n$	:	3D coordinates of a lattice protein structure
N	:	the neighboring vectors of the lattice model to use

Result:

TRUE/FALSE	:	whether or not position L_n has a free neighbored position

Method:

The approach performs a full check of all neighbored positions if necessary :

1: **for all** $\vec{v} \in N$ **do**

```
    i ← 1
    isFree ← TRUE
    while i < n and isFree do
        isFree ← L_i ≠ (L_n + v⃗)                ▷ check if occupied
        i ← i + 1
    end while
    if isFree = TRUE then
        return TRUE                              ▷ this neighbored position is free
    end if
end for
return FALSE                                     ▷ no free neighbored position found
```

2.3 Symmetry exclusion via normalization

The normalization is done indirectly via a conversion of the 3D coordinates into absolute move string representation. Here, a string is produced that represents the neighboring vectors between successive monomer coordinates. The resulting move string is normalized based on the neighboring and automorphism information of the given lattice. This information is used to create move exchange tables for each automorphism which is utilized to determine the lexicographically smallest symmetrical move string. The reconversion of this normalized move string into 3D coordinates results in the unique symmetrical (normalized) structure.

3 Available Lattices

Several lattice models can be used to fold a structure.

The currently supported lattice models and the corresponding neighboring vectors are:

ID	Name	Neighborhood vectors	#
SQR	Square	$\{\pm(1,0,0), \pm(0,1,0)\}$	4
CUB	Cubic	$\{\pm(1,0,0), \pm(0,1,0), \pm(0,0,1)\}$	6
FCC	Face Centered Cubic	$\left\{ \begin{smallmatrix} \pm(1,1,0), \pm(1,0,1), \pm(0,1,1), \\ \pm(1,-1,0), \pm(1,0,-1), \pm(0,1,-1) \end{smallmatrix} \right\}$	12

4 Energy Functions

LatSeF supports arbitrary energy functions that are based either on contacts or on distance intervals. The specification of an energy function has to be given in text format and defines the allowed sequence alphabet as well.

In general, the energy of a sequence S of length n with structure coordinates P is determined by

$$E(S,P) = \sum_{1 \le i+1 < j \le n} e(S_i, S_j, P_i, P_j). \tag{1}$$

Here, $e(S_i, S_j, P_i, P_j)$ is a placeholder for the specific evaluation function that is given for the different types in the following.

4.1 Contact Based Energy Function

A contact based energy function for an alphabet A is defined by an energy table $E^c : |A| \times |A| \to \mathcal{R}$ such that

$$e_c(S_i, S_j, P_i, P_j) = \begin{cases} E^c[S_i, S_j] & \text{if } P_i \text{ and } P_j \text{ are neighbored} \\ 0 & \text{else} \end{cases} \tag{2}$$

For example, a function like this was used by Lau and Dill to define the widely used HP-model [1].

Text File Encoding

The LatSeF text file enconding of a contact based energy function consists of two parts: the alphabet elements and the energy table. A consecutive string of the alphabet elements in the first line determines the allowed protein sequence characters (the alphabet) and the dimensions of the energy table that is read from the remaining file.

An example energy file for the HPNX-model is:

```
HPNX
-4.0  0.0  0.0  0.0
 0.0 +1.0 -1.0  0.0
 0.0 -1.0 +1.0  0.0
 0.0  0.0  0.0  0.0
```

4.2 Distance Interval Based Energy Function

A distance insterval based energy function for an alphabet A is defined by a consecutive set of k distance intervals with the upper bounds $d^{up}_{1\ldots k}$ and an energy table $E^i_{1\ldots k} : |A| \times |A| \to \mathcal{R}$ for each of them. Given the distance to interval index function idx we define the evaluation function

$$e_i(S_i, S_j, P_i, P_j) = E^i_{idx(P_i,P_j)}[S_i, S_j] \tag{3}$$

$$idx(P_i, P_j) = \arg\min_k \left(|P_i - P_j| \leq d^{up}_k\right) \tag{4}$$

Text File Encoding

The LatSeF text file enconding of a distance interval based energy function consists of three parts: the alphabet elements, the upper bounds of the intervals and the energy tables for the interval. A consecutive string of the alphabet elements in the first line determines the allowed protein sequence characters (the

alphabet) and the dimensions of the energy tables. The second line contains a whitespace separated list of the upper interval bounds. Their number sets the number of energy tables read from the remaining file. The interval bounds are expected to be given in Ångstroems. For a correct scaling of the bounds it is necessary to give the average distance of two consecutive C_α-atoms in the underlying model to LatSeF (see input parameters in Sec. 5).

An example energy file that encodes the HPNX-model using a distance interval based energy function is:

```
HPNX
3.7 3.9 999999

 0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0

-4.0  0.0  0.0  0.0
 0.0 +1.0 -1.0  0.0
 0.0 -1.0 +1.0  0.0
 0.0  0.0  0.0  0.0

 0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0
```

The number 999999 is used as a placeholder for $+\infty$, i.e. the upper bound of the last distance interval. It is important to know that the average C_α-distance in the underlying model was 3.8 Å. Therefore, the resulting interval energy function corresponds to the contact based energy function of the previous section; only distances close to the average C_α-distance are taken into account.

5 Program Parameters

Input

-seq The sequence to fold co-translationally. It has to be conform to the alphabet given by the energy file (see **-energyFile**).

-energyFile A file that encodes the used alphabet and the specific energy function (see Sec. 4 for format details).

-energyForDist If present, the input of **-energyFile** will be interpreted as distance interval energy function. Otherwise a contact based energy functino is expected.

-energyCalphaDist Specifies the average distance of two consecutive C_α-atoms in the underlying model. This value is needed to scale the intervals

of a distance interval based energy function onto the C_α-distances of the lattice model in use.

Lattice Settings

-lat The lattice model to use for the sequential folding. The available list of lattice identifiers is given in Sec. 3.

Side Chain Settings

-sideChain If present, a representation of two monomers per amino acid is done. One for the backbone atoms and one representing the side chain.

Chain Growth Parameters (see Method Sec. 2)

-deltaE Energy interval above the minimal energy found. Only structures with an energy within this interval are extended in the next iteration.

-noExtCheck If present, no check for extensibility of the last monomer is done. Otherwise, the last appended backbone monomer is checked to have at least one free neighbored position.

-extMax The maximal number of structures that are going to be extended in the next iteration. This parameter only influences the memory consumption of LatSeF by restricting the size of the candidate set C of the algorithm (see Sec. 2.1). If C exeeds the given size, the chain growth is stopped. Note, both sets, B and C, can maximally contain the given number of candidate structures.

Output

-print The number of final structures to print maximally in the end. The structures are given in absolute moves to reduce space consumption. The move encoding is:

move	neighbor vector
L	$(+1, 0, 0)$
R	$(-1, 0, 0)$
F	$(0, +1, 0)$
B	$(0, -1, 0)$
U	$(0, 0, +1)$
D	$(0, 0, -1)$

-printBest If present, only the structures with minimal energy are printed.

-best2PDB If present, it defines the file to that the best structure is written in PDB-format. Here, all monomers are considered to be a Histidine amino acid.

Miscellaneous

-v Give verbose output during computation.

-s Give only minimal output during computation.

-help Prints the available program parameters.

6 Contact

Martin Mann
Bioinformatics Group
University Freiburg, Germany

http://www.bioinf.uni-freiburg.de/

References

[1] Kit Fun Lau and Ken A. Dill: **A Lattice Statistical Mechanics Model of the Conformational and Sequence Spaces of Proteins**, *Macromolecules* 1989, **22**(10):3986–3997