REMC for Protein Folding

Paul Etheimer

September 15, 2022

Outline

Table of Contents

Monte Carlo Algorithms

Definition (MC algorithm)

A Monte Carlo algorithm is a problem solving procedures that uses randomness to approach a solution too complex to solve deterministically.

The HP protein folding model

Definition (HP model)

The Hydrophobic-Polar protein folding model is a highly simplified model of protein folding in space, that relies on the dominance of the *hydrophobic effect* on soluble protein folding. It sorts amino acid in two categories, hydrophobic or polar.

The Replica Exchange Monte Carlo

Definition (REMC)

The Replica Exchange Monte Carlo (or parallel tempering), is a type of MC algorithm. It runs parallelly multiple MC models, at different temperatures, and exchanges them based on their energy.

The VSHD neighbourhood

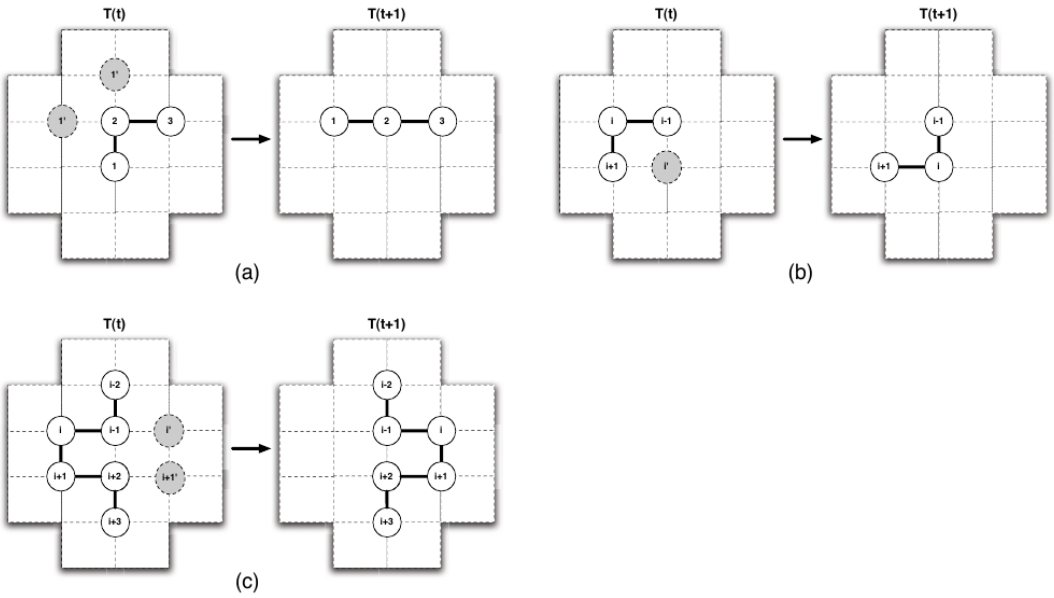

Figure: The 3 moves implemented : (a) end move (b) corner move (c) crankshaft move (figure from the article)

Definition (VSHD)

A *neighbourhood* of exploration, consisting of three moves a conformation can perform:

- an *end move* (only for end residues)

The VSHD neighbourhood

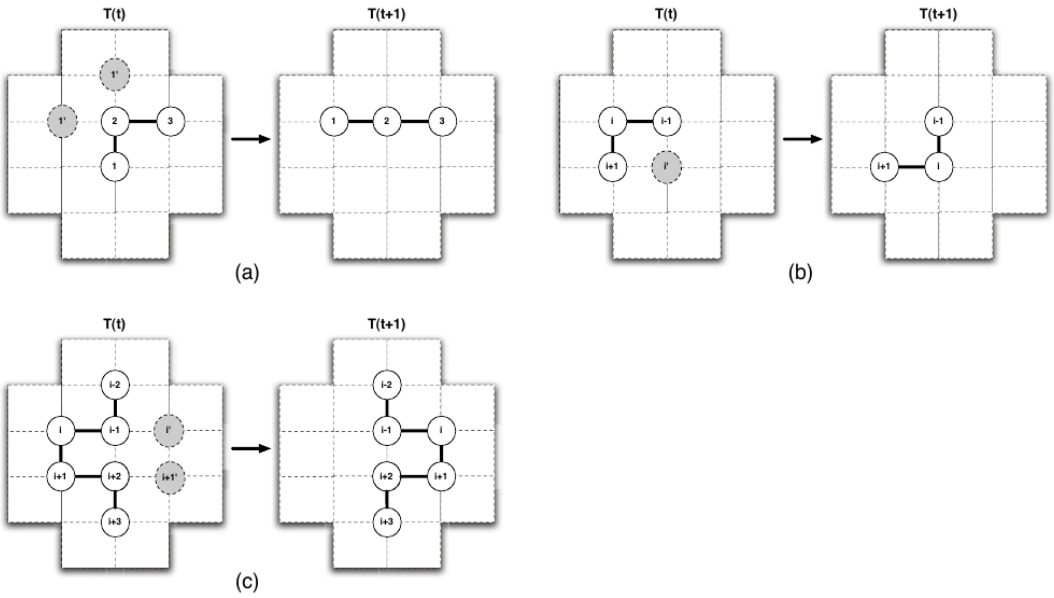

Figure: The 3 moves implemented : (a) end move (b) corner move (c) crankshaft move (figure from the article)

Definition (VSHD)

A *neighbourhood* of exploration, consisting of three moves a conformation can perform:

- an *end move* (only for end residues)
- *corner move*

The VSHD neighbourhood

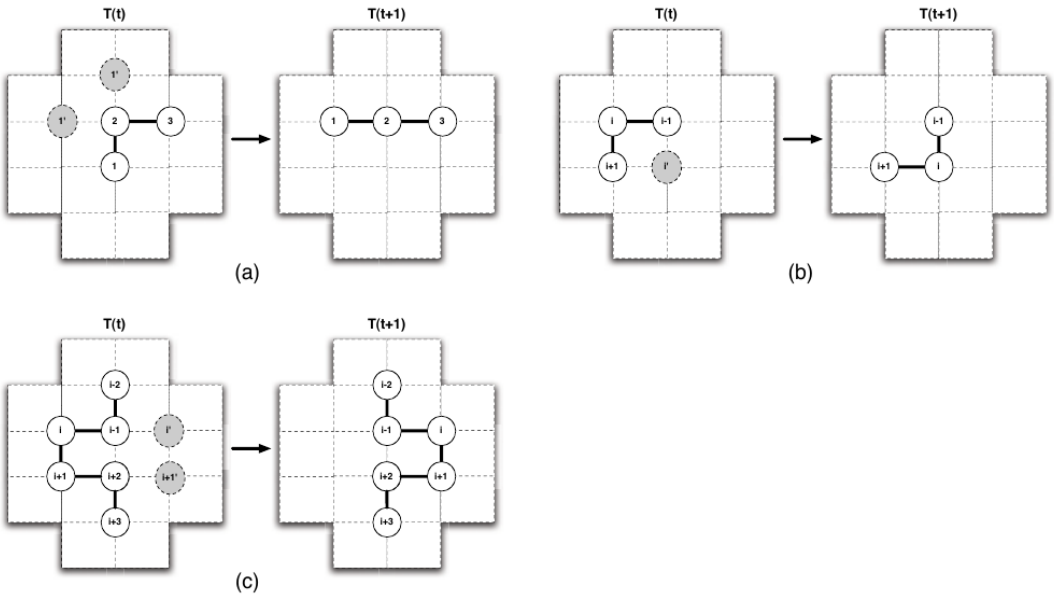

Figure: The 3 moves implemented : (a) end move (b) corner move (c) crankshaft move (figure from the article)

Definition (VSHD)

A *neighbourhood* of exploration, consisting of three moves a conformation can perform:

- an *end move* (only for end residues)
- *corner move*
- *crankshaft move*

Table of Contents

Object-oriented programming

- Python 3.10 with numpy 1.22

Object-oriented programming

- Python 3.10 with numpy 1.22
- 3 classes:
  - AminoAcid - A basic amino acid, attributes: `position`, `hp_type`, `index`
  - Conformation - The main class, attributes: `lattice`, `amino_list`, `sequence`, `energy`, `line`
  - Move - a class for movement, attributes: `move_type`, `conf`, `number`, `new_position`, `old_position`

The class relations

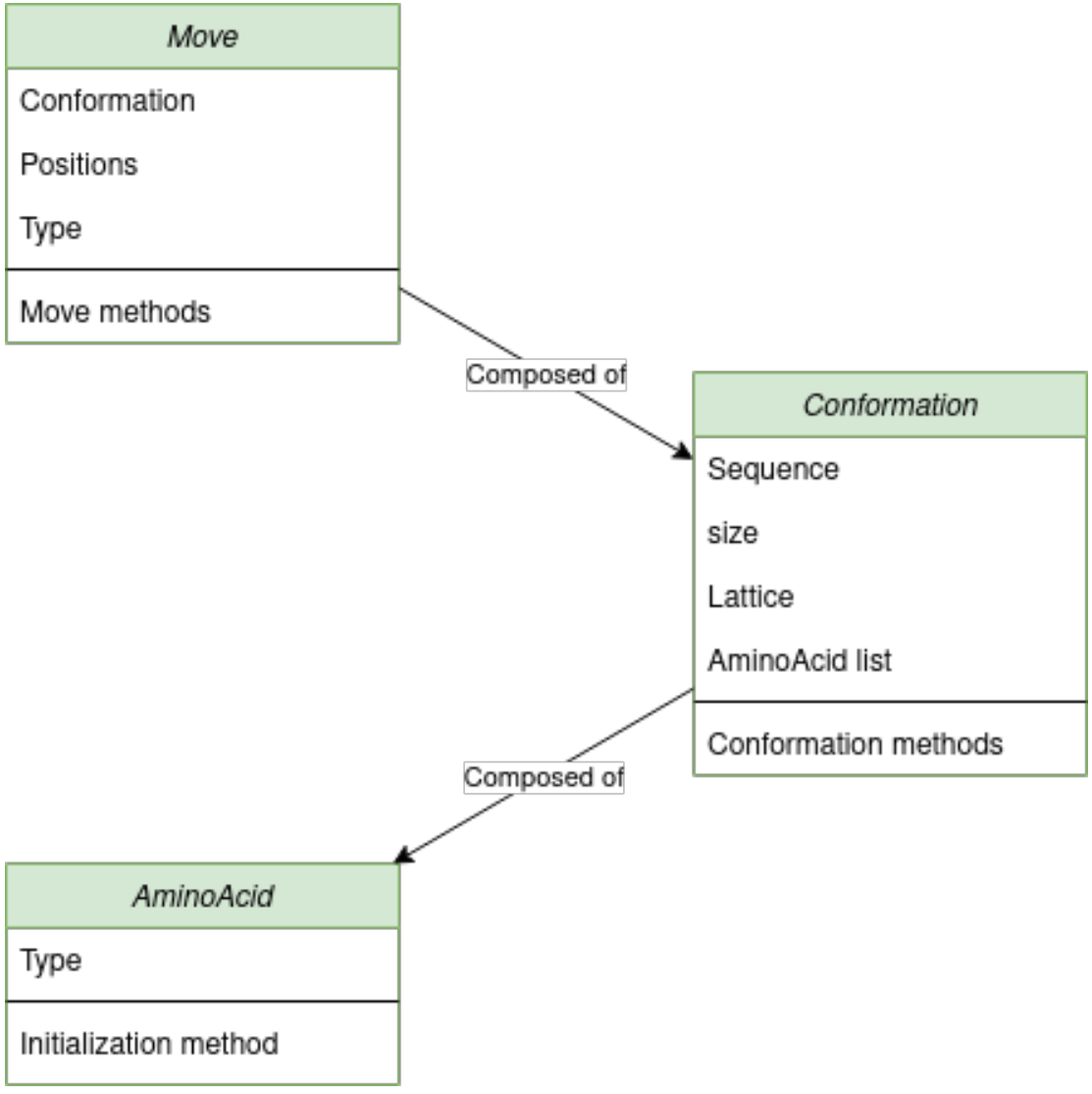

Figure: The 3 classes implemented and their relationships

Two functions and a main

- `mc_search`: Takes a conformation, drives its evolution step by step (internal loop)

Two functions and a main

- `mc_search`: Takes a conformation, drives its evolution step by step (internal loop)
- `remc`: Orchestrates all the replicas, stopping either at a target energy or at a specified maximum number of steps

Two functions and a main

- `mc_search`: Takes a conformation, drives its evolution step by step (internal loop)
- `remc`: Orchestrates all the replicas, stopping either at a target energy or at a specified maximum number of steps
- `main.py`: Parses the arguments with `argparse`, creates the conformation and runs the optimization

Around the program

- Used podman - but limitation to files entry in a CLI
- Used conda for file handling

Table of Contents

Middling at best

- Slow algorithm: two minutes for 110 residues with 5000 total steps (with random walk initialization)

Middling at best

- Slow algorithm: two minutes for 110 residues with 5000 total steps (with random walk initialization)
- Buggy crankshaft moves

Middling at best

- Slow algorithm: two minutes for 110 residues with 5000 total steps (with random walk initialization)
- Buggy crankshaft moves
- Not much movement without *pull moves*, especially with a line start position

Table of Contents

Bad decisions

- The exaggerated OOP (`Move`) and deepcopy usage

Bad decisions

- The exaggerated OOP (`Move`) and deepcopy usage
- No reproducible examples

Bad decisions

- The exaggerated OOP (`Move`) and deepcopy usage
- No reproducible examples
- No non-random test environment

Knowledge limits

- Lack of knowledge about debugging in Python

Knowledge limits

- Lack of knowledge about debugging in Python
- Lack of knowledge about performance profiling in Python

Tool limits

- Manipulating heavy lattices is unwieldly in Python

Tool limits

- Manipulating heavy lattices is unwieldly in Python
- Python is a slow language, especially with low-level programs such as this

Paper limits

- At least a mistake : a wrong comparison sign cost a lot of time
- The omission of the mention of the usage of the Boltzmann constant is also surprising

Procedure `MCsearch`$(\phi,\ c,\ \nu)$

Input: ϕ – the number of search steps to perform, c – the current conformation, and ν the search neigbourhood
Output: c' – the modified conformation
for $i \leftarrow 1 \dots \phi$ **do**
  $c' \leftarrow c$;
  $k \leftarrow \widehat{\mathcal{U}}(1, n)$;
  $c' \leftarrow \mathcal{M}(c', k, \nu)$;
  $\Delta E \leftarrow E(c') - E(c)$;
  if $\Delta E \leq 0$ **then**
    $c \leftarrow c'$;
  else
    $q \leftarrow \mathcal{U}(0, 1)$;
    if $q > e^{\frac{-\Delta E}{T}}$ **then**
      $c \leftarrow c'$;
    endif
  endif
endfor

Figure: The problematic $>$ sign

Thank you

Questions?