Genetic Algorithm

Molecular Statistic

2013

1 Introduction

The genetic algorithm is a general minimization algorithm, which in this project, is to be used to minimize the energy of alkane molecules, in the flavor of the general Monte-Carlo approach. The energy is calculated using the force field MMFF94 as implemented in the software package OpenBabel.

1.1 Force field and dihedral angle

We will be using a classic force field, and not quantum mechanics to simulate the molecule. The energy for the molecule in a certain configuration is calculated as

$$E_{\text{configuration}} = \sum_{i}^{\text{bonds}} k_i(r_i - r_{i,e})^2 + \sum_{i}^{\text{angles}} k_i(\sigma_i - \sigma_{i,e})^2 + \sum_{i}^{\text{dihedrals}} V_i[1 \pm \cos(n_i\omega_i)] + \sum_{i>j}^{\text{atompairs}} \left(-\frac{A_i A_j}{r_{ij}^6} + \frac{B_i B_j}{r^1 2_{ij}} + \frac{q_i q_j}{r_{ij}} \right) \tag{1.1}$$

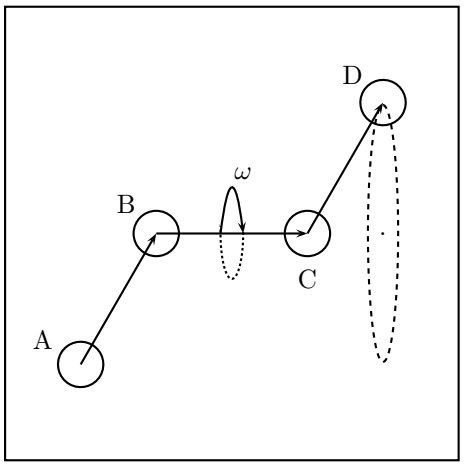

Figure 1.1: Illustrating the dihedral angle ω between 3 bonds and 4 carbon centers, A, B, C and D.

To simplify things, we will only be focusing on the dihedral part of the energy expression. In geometry, a dihedral (or torsion) angle is the angle between two planes, and in our alkane case it is the angle between two CH_2 groups, as illustrated in Figure 1.1. We want to change the dihedral angles in such a way that the lowest configuration energy is found, based on the dihedral configuration Ω. Where the configuration Ω is defined as a set of the systems dihedral angles ω_i between each of the successive CH_2 groups, $\Omega_\beta = \{\omega_1, \omega_2, ..., \omega_N\}$.

In the beginning each ω_i is assigned an initial random value, taken from a uniform distribution, between 0 and 360 degrees. The task is then to use the genetic algorithm and modify these ω_i angles so the total molecular energy will be minimized in respect to the geometrical configuration.

1.2 Genetic Algorithm

The genetic algorithm is an optimization algorithm that is based on genetic inheritance. Hence the name. The genetic algorithm is an efficient algorithm to search through conformational space and locate minima. Here we employ the genetic algorithm to try and locate the lowest energy conformation for an alkane.

As described before we define the geometry of an alkane chain through its dihedral angles ω_i between each of the successive CH_2 groups, and so by modifying these angles we change the structure and thereby the energy.

We start out by defining a state vector Ω_β as $\Omega_\beta = \{\omega_1, \omega_2, ..., \omega_N\}$ where we have N dihedral genetic angels. Each ω_i is assigned an initial random value taken from a uniform distribution from 0 to 360 degrees.
We then create a set of K state vectors. For each of these state vectors, we define a corresponding energy of the molecule $E_\beta(\Omega_\beta)$. The energy is obtained via the force field. From now on, these K state vectors are referred to as *parents*.

The parents are then changed as a result from the following algorithm steps;

1. **Mating.**
   Two parents Ω_β and Ω_α of length N can combined to give birth to two children with the "genome" of the parents represented by the its state vector. The state vector Ω_β is then being split from the first M elements into the first child, and $(M - N)$ into the next child. Same procedure for parent Ω_α. The cut index M is to be determined randomly.

2. **Mutation.**
   After the children has been born, sometimes, a random mutation is inserted in their 'genome'. This means randomly select one of the N angles ω_i and assign it a new random value, to a probability of a mutation rate, `mutatation_rate`.

3. **Evolution.**
   Survival of the fittest. The energy of the formed child is evaluated like the parents, minimize the geometry of its state vector. If the energy if the child is lower than one of the parents, then the child is kept and the parent with the larger energy is removed, where the child takes its parents place. If the energy of the child is larger than a parent, there is still a probability P that it survives the evolution step. This is where the Monte Carlo enters the algorithm The probability P is defined as proportional to the Boltzmann factor, i.e.

   $$P = e^{-\Delta E/(k_B T)} \tag{1.2}$$

   where ΔE is the energy difference between the child and the parent, and T is the temperature of the system. In the following we set $k_B = 1$, so all temperatures are in units of k_B.

   When *one* child has replaced a parent, we consider this to be the end of the generation.[1] This means that no more mating of parents should be carried out and we start over.

[1] A big hint here would to look up the `break` command for loops

2 Assignment

The main task of this assignment is to implement the genetic algorithm, and minimize the energy of three alcane chains, $C_{10}H_{22}$, $C_{20}H_{42}$ and $C_{40}H_{82}$.

2.1 Setup

To successfully do this assignment you need to install some software dependencies on your computer. This is done as easy as writing the follow command in the terminal.

```
sudo apt-get install openbabel python-openbabel
```

2.2 Examples

You will be using the module `molecule.py` which in turn uses the openbabel package. The module is imported the usual way and is found on the course website.

```
import molecule as mol
```

Please also download the `butane_example.py` from the course website and look at the source code.

There are three functions you will be using from the module.

```
mol.generate_alkane(N)    # Generates an alkane chain of N length
mol.set_dihedral(angles) # Sets the dihedral angles of the molecule. Requires
    list as parameter
mol.get_energy()          # Calculates the current energy of the molecule.
    Returns energy.
```

The molecule and the coordinates itself is defined and stored in the `molecule` module. To save the structure you can use the following function

```
mol.save_molecule('molecule_name.xyz')
```

which will save the structure in XYZ format. You can use the program Avogadro or JMol to open and see the structure.

2.3 Simulation Setup

Before you finish the actual simulation you need to create the functions that simulates the system. To make it easy to start we have split the setup into small manageable steps.

1. Use the code from `butane_example.py` and familiarize yourself with it.
2. Create a list of angles from 0.0 to 180.0 degrees and calculate the energy of Butane for each angle. Plot the result.
3. Implement the following optimization algorithm, called **Greedy optimization**:
   (a) Create a random list of dihedral angles
   (b) Create a integer `no_generations` which represents how long the algorithm will go on.
   (c) Create a for-loop and loop over the no of generations. For each loop generate a random dihedral state and calculate the energy.
   (d) If the energy is lower than the previous, save the new energy and the new state, otherwise discard new state the away and continue the search.
4. Create a function that takes `m` and `dihedral_list` as parameters and returns the energy of the configuration.
5. Create a function that takes the parameters `parent_alpha`, `parent_beta` and `mutation_rate`. Have this function mate the two parents and create two children based on the genetic algorithm step 1 and 2. Use numpy's `ranint` generate a random cut index M

```
m = np.random.ranint(0, N)
```

6. Create two lists, one for the containing the state vectors and one containing the energy related to the state vector. Fill them up with `no_parents` of random dihedral states, and calculate the energy for each.
7. Finish the algorithm by creating a for-loop and loop over number of generations defined, mating each parent pair, selecting what parent and what child survives based on the Genetic algorithm.

2.4 Simulations

Now that you have a working simulation, it is time to do the actual simulations:

- **Simulation 1**
  Minimize $C_{10}H_{22}$, $C_{20}H_{42}$ and $C_{40}H_{82}$ and plot the mean energy for each generation.
- **Simulation 2**
  Check for correlation between temperature T and the energy found after G generation for each molecule. Check for temperatures between 0.5 and 10.0.
- **Simulation 3**
  Check for correlation between mutation rate in the interval [0:0.5] and the energy found after G generation for each molecule.
- **Simulation 4**
  Compare the Greedy and Genetic algorithm by plotting the energy vs generation.
- **Simulation 5**
  Check the population size K of parents. Plot the number of generations it took to converge as a function of the number of K parents.
- **Simulation 6**
  Make the Genetic Algorithm better. You have to come up with one changes to the mating and mutation routine. Change the algorithm to make your own personal minimization algorithm. Document the results.