Global Optimization of Lennard-Jones Clusters using Basin-Hopping Technique

Dylan Durkee

PHYS 300 – Computational Physics

Dr. Qiang Zhu

December 2019

1. Introduction

Optimization techniques have been of significant interest in various fields such as economics and finance, electrical and civil engineering, materials science, and more. Economists use optimization techniques in methods such as control theory to model labor-market behavior, for example, whereas electrical engineers apply optimization to the routing of circuitry. There are many techniques available for finding local minima and maxima of objective functions of one or more variables. In principle, local optimization is relatively straightforward, but not suitable for finding global minimums in systems with many local minima.

Global optimization techniques have been developed to tackle this issue: researchers have implemented techniques such as simulated annealing, Monte Carlo, as well as ones which take advantage of the principles of quantum tunneling. Here we are interested in the **basin hopping** technique, first described by David Wales and Jonathan Doye. In this paper, I introduce the fundamental principles behind the basin hopping algorithm in the scipy module and optimize three Leonard-Jones configurations of N=13, 15, and 17 using basin hopping. In doing so, I show the results of adjusting several important parameters in the function and provide an analysis of the parameter values that generated the global minima for the three systems.

2. Theoretical Background

In their work, Wales and Doye tested their basin hopping global optimization method to minimize Leonard-Jones (LJ) potential clusters with many atoms. At the time, techniques such as simulated annealing and those based on quantum tunneling could not sufficiently describe both large (say, N>50) and small (say, N<10) clusters.[1] This motivated Wales and Doye in part, to develop an optimization method that could apply to clusters of N atoms, and whose algorithm could extend to other problems involving rugged potential energy surfaces (PES) with many local minima such as in the modelling of ground state structures for biomolecular systems .[2] A simple Leonard Jones potential is shown in figure 1 followed by the equation which describes the total potential energy of the configuration.

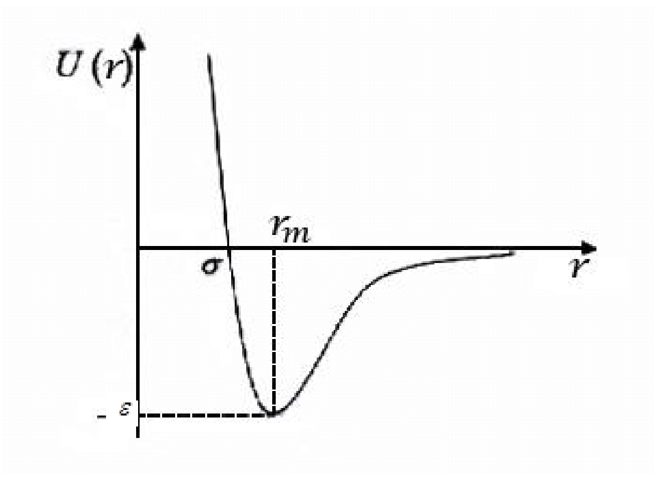


Figure 1: Leonard Jones potential for a two-atom system showing potential energy as a function of separation distance.
Source: Davoud Raoufi et al. "Study of Carbon Atoms Deposited on Graphene Layer Using Molecular Dynamics Simulation." AIP Conference Readings, 2019.

The LJ potential is described by the following equation:

$$V(\boldsymbol{r}) = r\varepsilon\left[\left(\frac{\delta}{r}\right)^{12} - \left(\frac{\delta}{r}\right)^{6}\right]$$

where r is the separation distance between each pair of atoms, ε is the depth of the potential well, and δ represents the distance at which the inter-particle distance is equal to zero. In this paper, both values are taken to be equal to 1. As the number of atoms in the system increase, the number of local minima in the PES drastically increases and therefore selecting appropriate parameter values to pass into the basin hopping optimizer becomes increasingly important.

In this project, I use scipy's basin hopping module to find the global minima of LJ clusters. This function takes in several parameters, one of them being the function to minimize, and the initial starting positions of (in the case of a LJ potential) the atoms. Below is scipy's basin hopping function along with its possible parameters one can pass in.

```
def basinhopping(func, x0, niter=100, T=1.0, stepsize=0.5,

            minimizer_kwargs=None, take_step=None, accept_test=None,

            callback=None, interval=50, disp=False, niter_success=None,

            seed=None):
```

Most of the parameters are optional; I use the default minimizer implemented in the source code (conjugate gradient), as well as the default Metropolis acceptance test criterion. In my experiments, I set the niter_success parameter equal to 10 for all configurations so that the program terminates if a new minimum function value is not found after ten iterations. Finally, I adjust the parameters for temperature **T**, and the **stepsize**. For the algorithm to function optimally, choosing appropriate values for these two parameters are crucial.

2.1 The Basin Hopping Algorithm

The basin hopping algorithm is an iterative stochastic algorithm that has three primary steps:

1) Random perturbation of the current coordinates

We take a random step with an optimal step size applied to each Cartesian direction. The random step in either direction is a value chosen between $x_o - stepsize$ and $x_o + stepsize$.

2) Local minimization

Next, local minimization is done on the current position along the PES using the module's default conjugate gradient minimizer.

3) Accept or reject the new function value.

$$e^{-(func(new)-func(old))/T} \qquad [1]$$

The probability of accepting or rejecting the new function value is determined by equation 1, where *T* is the value of the *temperature* parameter. Thus, the acceptance criterion is largely dependent on the temperature value chosen for the LJ configuration of interest. The value for *T* should be comparable to the difference in *function* value between local minima as opposed to the height of the walls between two local minima. This acceptance criterion uses Monte Carlo algorithms and is based on the Metropolis criterion. In determining whether to accept or reject coordinates based on the new function value, first an upper bound is created via the following variable:

$$w = math.\exp\left(\min\left(0, -float(energy_{new} - energy_{old}\right) * \frac{1}{T}\right) \qquad [2]$$

Next, a random number is generated between 0 and 1, using a random number generator.

$$u \in [0,1] \qquad [3]$$

If the random number *u* is less than or equal to *w* in equation 2, then the new coordinates and function value are accepted – otherwise, they are rejected.

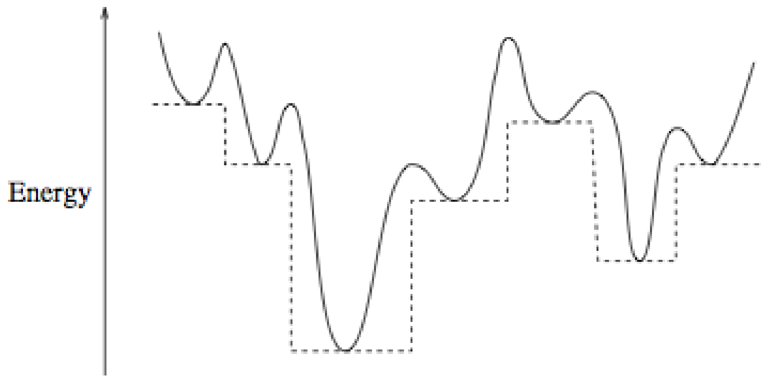


Figure 2: 2D visualization of the basin-hopping global optimization technique. Staggered "staircases" around each local minimum demonstrate possible locations for "hopping," where each minimum is optimized and a subsequent step taken until the global minimum is found.
Source: https://esa.github.io/pagmo2/docs/cpp/algorithms/mbh.html

Figure 2 visualizes how the basin-hopping technique works. In this figure, the solid line represents a possible PES with many local minima. The basin hopping function transforms the geometry of the PES into a series of staggered basins, then iterates through the algorithm outlined above until the specified number of iterations are complete. Note that the function does not know when it has found the global minimum; therefore, it is up to the user to convince himself that it has.

3. Methods

I write two functions to pass into scipy's basin hopping function: a position initializer for N-atoms, as well as the total energy function which we wish to minimize. The source code for these functions are shown below.

```python
@jit
def init_pos(N, L=5):
    """
    Intialize starting positions in x,y,z coordinates for N number of atoms.

    Return: Array with random Cartesian positions of N atoms.
    """
    return L*np.random.random_sample((N*3,))


@jit
def total_energy(positions):
    """
    Calculate the total energy

    input:
    positions: 1*N array which represents the atomic positions in Cartesian coordinates.

    output
    E: the total energy
    """
    E = 0

    N_atom = int(len(positions)/3)

    for i in range(N_atom-1):
        for j in range(i+1, N_atom):
            pos1 = positions[i*3:(i+1)*3]
            pos2 = positions[j*3:(j+1)*3]
            dist = np.linalg.norm(pos1-pos2)
            r6 = dist**6
            r12 = r6*r6
            E += 4*(1/r12 - 1/r6)
    return E
```

The total_energy function is the function we wish to minimize using scipy's basin hopping algorithm. Below, I show the source code for a typical run, using the N=13 atom configuration as an example.

```
true_13 = -44.326801
N_13 = 13
pos_13 = init_pos(N_13)
start = time.time()
res_13_1 = basinhopping(func=total_energy, x0=pos_13, niter=20, T=0.5, stepsize=2.0, niter_success=10, disp=Tr
ue)
print("Total time to convergence: ", time.time()-start, 's')
print("Global minimum value: ", res_13_1.fun)
print("True minimum value: ", true_13)
```

In this project, I looked at three LJ configurations: N=13, 15, and 17. First, I tested extreme values for the parameters – a large temperature value with small stepsize, and a large stepsize with small temperature value.

Then, I adjusted the temperature and stepsize parameters until a suitable range was found that successfully found the minimum energy values for each configuration. I kept the number of iterations below 40 for all three configurations. Furthermore, to keep the searches unbiased, I did not input a seed into the basin hopping function for any configuration. Finally, using the atomic positions returned by the basin hopping function, I visualize each LJ structure with 3D plots.

4. Results & Analysis

Firstly, choosing too large and too small of temperature and stepsize values were insufficient in finding global minima. In either case, the basin hopping function either got "stuck" in a local minimum from which it could not hop out or did not have suitable step size to hop between local minima to the global minimum basin. The results show that there is not one set of temperatures and stepsizes per LJ configuration that lead to global optimization; rather, there is a small range of values that *more often* have success in finding the global minimum. This can be seen in **Table 1**, which shows parameter values that lead to success for each configuration.

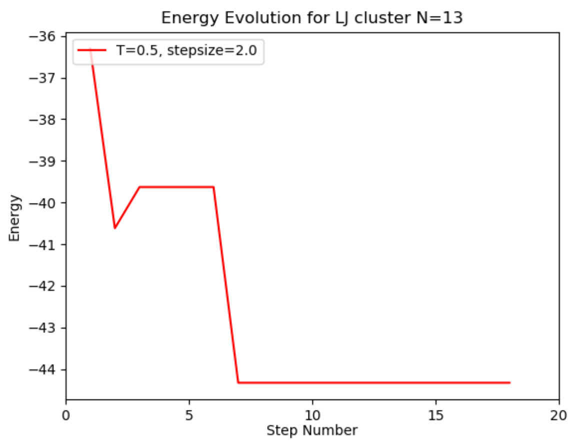
Number of Atoms	Temperature (*T*) Values	Stepsize Values	Global Minimum(s)	True Ground State Energies
13	0.3, 0.5	2.0, 3.0	-44.32680141953	-44.326801
15	3.2, 3.4	4.2, 4.4	-52.32262726182	-52.322627
17	3.7, 3.8	4.9, 5.5	-61.30714608155, -61.31799466008	-61.317995

Table 1: Range of values for parameters *temperature* and *stepsize* which successfully found the global minimum for the three configurations (N=13, 15, and 17). Also shown are the global minima found by the basin hopping searches, as well as the true ground state energies as reported by Cambridge.

For the N=13 and N=15 LJ clusters, two combinations of temperature and stepsize values successfully found the global minimum, which matches the true ground state energy reported on the Cambridge database.[3] One combination of temperature and stepsize values successfully found the global minimum for the N=17 cluster; another combination came fairly close to the true energy, but did not succeed in finding the global minimum value. In all cases, global minima were found in under 40 iterations; the number of iterations required for convergence was generally greater for the larger N=15 and N=17 configurations. As more atoms were introduced into the system, higher values were required for success for the two parameters. A general trend for the evolution of these parameter values with respect to the number of atoms is difficult to deduce, namely since there is a much larger difference in parameter values between the N=15 and N=13 clusters than between the N=15 and N=17 clusters. Surprisingly, following a linear trend in parameter values did not lead to much success for the N=17 configuration.

Interestingly, the parameter values in table 1 worked only for a *greater number* of runs than other values, but not for all runs – this was true for all three configurations. As the algorithm is stochastic, convergence to the global minimum is highly dependent on the initial starting positions of the atoms, which determines the landscape of the PES. Therefore, a set of temperature and stepsize values may not lead to success for all starting positions. For example, if the initial randomized structure generates an initial local minimum much greater in energy than the desired global minimum, the values in table 1 will not lead to success. This is due to the parameters placing constraints on the random perturbations of the atom positions and the acceptance criteria for new function values. In principle, one can pass in a seed in order to constrain the randomized starting positions; however, I chose not to do this to keep the basin hopping search unbiased.

In Figures 3 and 4, I plot the energy evolution as a function of steps for each LJ configuration: here, I show the plots whose parameters had the best success.



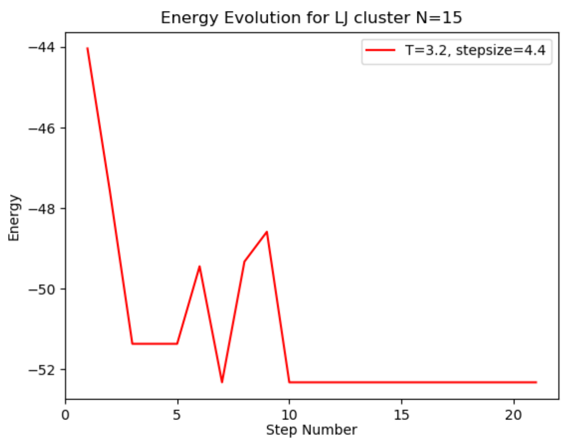


Figure 3: Energy Evolution as a function of iterations (steps) for **Left:** N=13 atoms, and **Right:** N=15 atoms. Chosen parameter values show a smooth transition to the global minimum. **Bottom:** N=17 atoms.

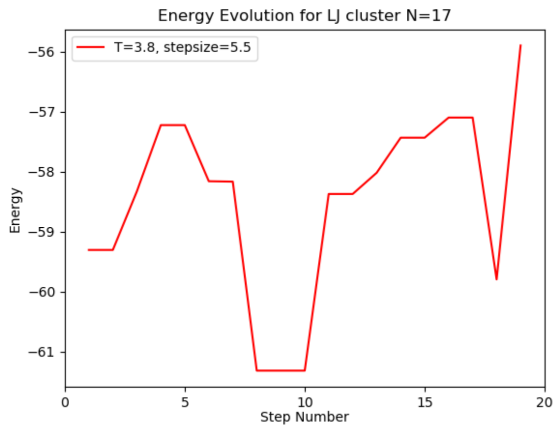


Figure 4: Energy evolution as a function of iterations (steps) for the N=17 configuration.

For the smaller configurations of N=13 and N=15, the basin hopping function had a much easier time finding the global minimum. Furthermore, once the global minimum value was found, each run remained in this basin as can be seen in the plots in figure 3 which shows relatively smooth transitions into the global minima. On the other hand, the energy evolution for the N=17 configuration was much more jagged, and the function did not remain in the global energy basin once found. Thus, as more atoms are introduced into the system, the energy evolution curves become less smooth. This can be attributed to the exponential increase in the number of local minima across the PES as the system becomes larger.

Finally, one useful feature of scipy's basin hopping module is that it returns the final coordinates of the atoms that produce the global minimum. With these atomic positions, I use matplotlib.pyplot's 3D plotting functions in order to visualize the ground state structures for each configuration. Figure 5 shows that the most symmetric configuration is for the N=13 icosahedral structure.

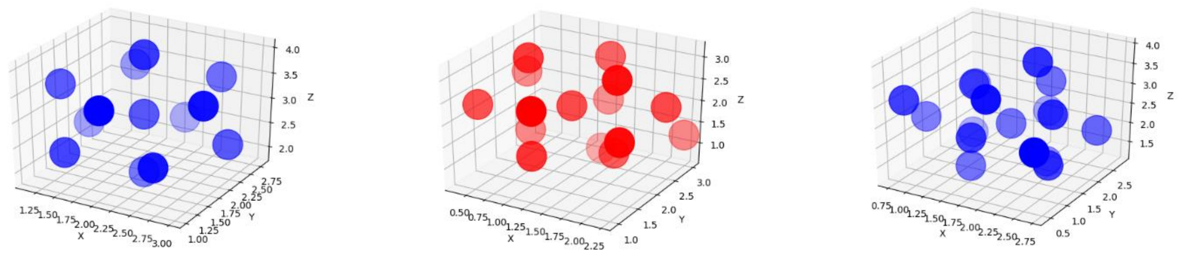


Figure 5: Final ground state structures visualized by matplotlib.pyplot's Axes3D package. From left to right: N=13, N=15, and N=17 configurations.

5. Conclusion

Basin hopping is a useful global optimization technique whose adjustable parameters allow for successfully finding global minima of complex PES. Here, I looked at three LJ configurations which were relatively small: N=13, N=15, and N=17 atoms. Several combinations of temperature and stepsize parameter values successfully found the global minimum for each configuration in under 40 iterations. Unexpectedly, the difference in parameter values in successively large configurations did not seem to follow a linear trend. For smaller configurations N=13 and N=15, the basin hopping function found the global minimum much smoother, and the energy evolution as a function of iterations showed a smoother transition to global minimum than for the N=17 configuration. As basin hopping is a stochastic algorithm, it is important to note that although certain parameter values find the global minimum value more successfully than others, success is not guaranteed. One must consider that the initial PES generated by the randomized initial starting positions constrains one to a certain region along the PES. Thus, in applying the basin hopping algorithm to other systems, it is crucial to have some knowledge of the nature of the PES, namely how many local minima there are as well the distance in function value between them. Having this background will allow one to choose suitable parameter values as well as putting boundaries on the initial starting positions.

[1] J. Wales, David and P. K. Doye, Jonathan. “Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters up to 110 Atoms,” *Journal of Physical Chemistry Letters, A.* 101 5111-5116 **1997**.
[2] Roeder, Konstantin and J. Wales, David. “Mutation Basin-Hopping: Combined Structure and Sequence Optimization for Biomolecules,” *Journal of Physical Chemistry Letters*, **2018.**
[3] The Cambridge Cluster Database, D. J. Wales, J. P. K. Doye, A. Dullweber, M. P. Hodges, F. Y. Naumkin F. Calvo, J. Hernández-Rojas and T. F. Middleton, URL http://www-wales.ch.cam.ac.uk/CCD.html.