**Global Optimization via Basin-Hopping Algorithm**

An explanation of the development, usage, and significance of the algorithm

****

Katie Randolph

Computational Physics, Fall 2017

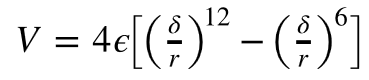
University of Nevada, Las Vegas

**Abstract**

Global optimization of a function with several minima can become a computationally challenging problem when the function gets sufficiently complex. One of several algorithms developed with the aim to increase the efficiency of a global optimization calculation is referred to as Basin-hopping. This is a stochastic method which increases efficiency by modifying the conformational space to remove high-amplitude peaks from the calculation. The current study utilized the basin-hopping algorithm in Python’s Scipy library to explore how the modification of certain parameters affected the outcome of the calculation.

**Background**

*Lennard-Jones Clusters*

Lennard-Jones (LJ) Clusters is a widely used model for interatomic interactions. It refers to the problem of predicting how any N number of atoms in a given system will tend to become geometrically arranged by calculating the amount of energy in all possible configurations. The equation for this calculation is given by:

Where ε is the depth of the potential well, σ is the distance at which the potential crosses zero, and r is the distance between two atoms. For these calculation, it is assumed that ε = σ = 1 for simplicity.

This problem was initially given attention as a method to calculate nucleation rates for noble gases (Wales & Doye, 1997). However, due to nature of the function and the extensive amount of data collected on this problem, it easily lent itself to use in investigating finite size effects on phase transitions and global optimization (Wales & Doye, 1997).

*Global Optimization*

An entire branch of the computational methods field has been dedicated to the detection of the global minimum of a continuous function with several local minimums. This is because global minimization is a problem applicable across several STEM fields, and also computationally expensive to solve. LJ clusters fit in this field because the equation calculates the energy of a molecule as a function of its geometry. This function is referred to as the potential energy surface (PES) of a system (Wales & Doye, 1997). When the PES is calculated at every possible configuration and then the minimum energy of all these values are found, we say the PES has been globally optimized. This makes LJ clusters the perfect model by which to design and test methods of optimization which attempt to increase the calculation efficiency. While some numbers of atoms are quite easy to minimize, others have a much more complicated PES and thus are more taxing to solve. This range of complexity is a desirable trait in a model system because a method can be developed on easy systems and tested on complex ones.

Simulated annealing was one initial method to tackle this problem. This method exploits the physical process of continuously heating up a molecular system and cooling it back down in order to get the system to find a minimum energy state. This method is effective for systems whose PES stays the same from low to high temperatures, but many systems such as LJ38 or LJ75 have free energy states that are different from potential energy states until the system is at temperatures well below melting (Wales & Doye, 1997). This makes measuring an energy minimum a big problem for a method that involves continuous melting and solidifying.

Another method termed Genetic or Evolutionary Algorithm, makes use of Darwinian Evolution to optimize the system. It does this by considering the state of the system on any one iteration as a population. Advancing to the next iteration is the same as moving forward a generation. As generations progress, the fitness of the population is defined to increase as the energetic state decreases. Between generations, there is some chance of the population mutating or swapping characteristics, which randomly changes future generations. Random changes which result in lower energy will be preferred as more fit and thus more likely to pass on that set of traits to the next generation as higher numbers of these characters are able to accumulate. After several generations, the result is a minimized function. It was once the case that this method was quite efficient for systems with small and symmetric ground states, but often failed for a system with a more complex PES (Schönborn et al., 2008). However, researchers continued to improve the efficiency of this algorithm to the point that it has become one of the most effiecient option for complex problems. This was achieved by optimizing a number of things including the improvement of the initialization condition to choose a point based on predicted physical parameters as opposed to random generation. In addition, researchers realized it was possible to weight not only fit individuals, but also identify and nurture fragments of those structures that made the individual structure more fit. A very smart idea came from recognizing the nature of genetic drift that can occur in any population growing from a small number of individuals. Genetic drift occurs when a certain characteristic lended to fitness in the early stages of the developing population and as such became ubiquitous throughout the population several generations later. In this computational context, this corresponds to the problem of getting trapped in local minima which keeps the algorithm from finding the global minimum. This problem was addressed by the idea of “antiseeding” which means that the algorithm stores the function’s x and y value on each trial in order to avoid revisiting that point several times. This astute modification encourages the algorithm to more thourougly sample the entire landscape, thus improving the probability of finding the global minimum (Lyakhov et al., 2013).

A third clever approach to this problem would be to realize that the high energy transition states included on the PES severely slow down the computation by any method because of the time it takes to overcome those high energetic barriers. A class of methods, called hyperspace deformation, mathematically transform the PES to make it smoother and easier to minimize. The problem with some of these methods is that it is unclear if the mathematically transformed space maps reliably back onto the conformational space (Wales & Doye, 1997).

The solution to this was thought of by Li & Scheraga in 1987 and refined by Wales & Doye as published in their 1997 paper where they termed this new method Basin Hopping. Basin Hopping is a Monte Carlo minimization method which only samples discrete sets of energy minimums (basins) and leaves out the higher energy transitional states (Fig. 1).

This method consists of an algorithm which iterates over these three steps:

1. Random displacement of atom coordinates
2. Local minimization
3. Decide to accept or reject the new coordinates based on the following function being larger or smaller than a random number:

As such, this process is stochastic on two levels: both at the random displacement of the coordinates and when deciding to accept or reject new coordinate values.

Upon analyzing these three steps, it can be seen that there are three parameters which considerably impact the outcome of the computation. The number of iterations clearly has an impact, but more interesting to consider is the effect of step size of the random displacement (from 1) and the effect of the temperature factor in the accept or reject criteria (from 3), where the temperature factor is equal to kT in the equation listed.

The step size of coordinate displacement (S) should be chosen to be close to the typical separation between local minima of the function being optimized (separation of basins in the x direction). The temperature condition (T) should be comprable to the separation of height of function values between minima (separation of basins in the y direction). A higher temperature means that larger jumps in the function value will be accepted.

The current study seeks to analyze the effects of modifying these parameters across different N number of atoms in LJ clusters.

**Figure 1:**

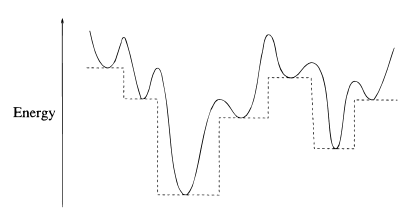


Fig. 1: The solid curve represents the PES for a given system and the dotted line shows the transformed energy space used in the basin-hopping procedure.

**Current Study: Methods and Results**

The computations in this study were run in Python and made use of the basin hopping algorithm stored in the Scipy library (see appendix for code). Calling this algorithm requires specification of the function to be optimized, the initial position, the number of iterations, the temperature condition (T), and the step size of coordinate displacement (S). I ran this alorithm for 12 atoms, 13 atoms, and 19 atoms (N=12, N=13, & N=19, respectively).

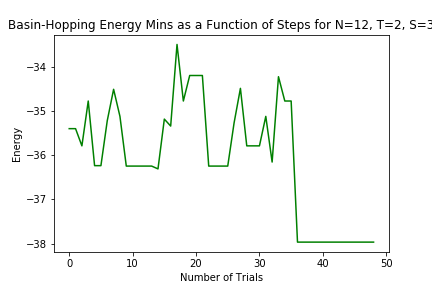
All models were evaluated for correctness based on the values documented by the Wales Group at Cambridge, which gives values for the aforementioned number of atoms of:

|  |  |
| --- | --- |
| Number of Atoms | Global Minimum |
| N=12 | -37.967600 |
| N=13 | -44.326801 |
| N=19 | -72.659782 |

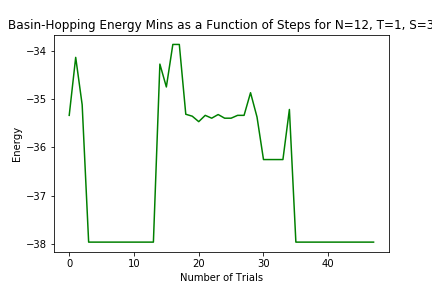
This and all other numbers of atoms from 1-150 can be found at the following website: http://doye.chem.ox.ac.uk/jon/structures/LJ/tables.150.html

*N=12*

The PES for twelve atoms is relatively simple to solve. This is reflected by the wide range of effective parameters. The global minimum was found under 40 iterations for a temperature equal to 2.0 (T=2.0) and step size equal to 3.0 (S=3.0).

****

When the step size was held constant but the temperature was dropped a full integer to 1.0, something interesting happened:



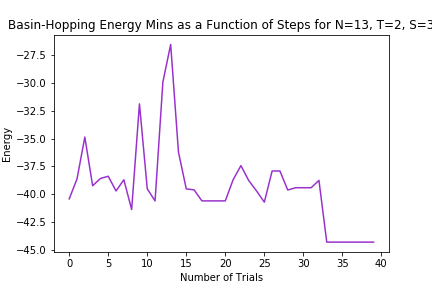
This is interesting because the global minimum was found in 3 steps, but there was enough energy for the algorithm to accept values that jumped back out of that basin, and it took many more trial to find the global minimum again. Notably, each time the global minimum was found, the algorithm remained in that basin for several iterations.

Notably, I also ran this computation with parameters T=2.0, S=2.0 ; T=1.0, S=2.0 ; T=2.0, S=2.5 ; and T=3.0, S=2.5. All of these runs successfully found the minimum, supporting the claim that a wide range of parameters can be used to find the minimum of this simple PES.

*N=13*

This is the first of what are called, “magic numbers” in the LJ cluster set. It is referred to as magic because it is the first number of atoms of which it is possible to build a complete icosahedron (Fig 2a); a highly stable geometric structure. It has a single funnel to the global minimum from all local minimazations allowing efficient relaxation from high energy state.

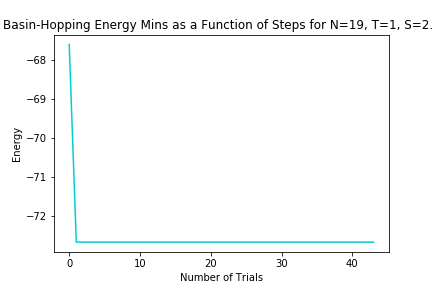
When temperature was set to 2.0 and stepsize to 3.0, the global minimum of this structure was found under 35 steps.



This number of atoms required a much tighter range for effective parameters. In addition to the above, I also tried T=1.0, S=2.0, as well as T=2.5, S=3.5 but these parameters failed to find the correct global minimum. In 50 iterations it only got down to -32.7660 and -41.472 respectively. The one other that succesfully found the minimum was T=2.0, S=2.7, which is very close to the values used to create the above graph.

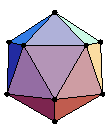
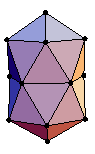
*N=19*

This is another special LJ cluster number of atoms because its global minimum is a double icosahedron **(**Fig. 2b).This number has a shallower gradient towards the global minimum and of the three reported numbers, the required the tightest effective parameters. I ran T=2.0, S=3.5 ; T=1.0, S=3.0 ;T=2.5, S=2.7, but these all failed to find the appropriate minimum. However, when I set temperature equal to 1.0 and step size equal to 2.7, the algorithm found the global minimum on the second trial and stayed there for tens of steps:

****

This suggests that though the correct parameters are difficult to find, once they are found, the algorithm drops to the global minimum quickly and remains their stably.

**Figure 2:**

****

b.

a.

Fig. 2a. shows the icosahedron made by N=13 global minimum.

Fig. 2b. shows the double icosahedron made by N=19 atoms.

*N=38*

This number of atoms was mentioned in the introduction as one of the test numbers for the efficiency of any algorithm. This number has a global minimum that is much harder to reach compared to several of its local minima due to the layout of its PES. Below is an image of its disconnectivity graph, which shows all the minima in respect to their distance from each other.

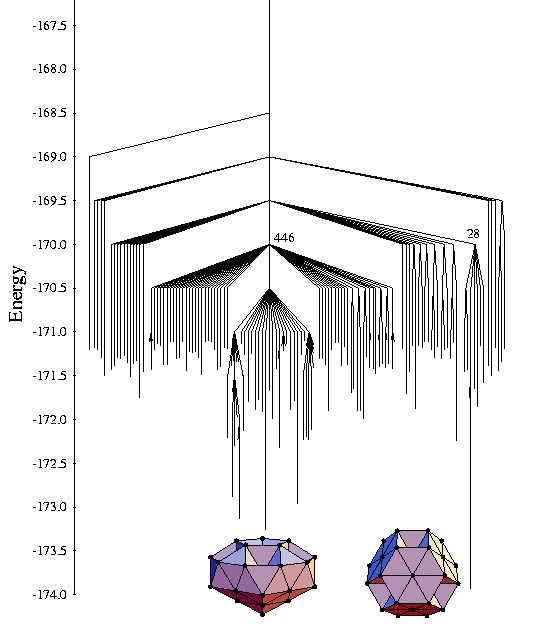


Image Credit: http://doye.chem.ox.ac.uk/research/forest/LJ.html

As can be seen from the graph, the global minimum circled on the far right has a large energy barrier between it and other local minima which can be more readily funneled to. In addition to this, it has a large chuck of local minima very close to it, such that if by random chance the algorithm does explore that branch, it may get stuck in a number of other minima before bouncing out.

For landscape such as this one, the evolutionary algorithm’s ability to antiseed would be of particular use. However in this paper, I did not have that tool at my disposal. I tried 8 different parameter sets where I iterated across 150 trials, but did not succeed in finding the global minimum. Perhaps if there were another type of parameter similar to the idea of antiseeding in the genetic algorithm but for basin hopping, the calculation for this particular PES could vastly improve.

**Discussion**

The basin-hopping algorithm stored within Python’s Scipy Library is simple and efficient for predicting the lowest energy state for a LJ system, and more generally, finding the global minimum of any system. By adjusting a few simple parameters, these tools enable any researcher to quickly and easily find the global minimum of a system. It is important to note, however, that said researcher should have an understanding of the physical principles governing the target system to be optimized. Understanding the physics help the researcher to pick the best method of optimization as well as reasonable parameters from the start, making calculations much more efficient.

Analyzing the most efficient computational method find the global minimum is a very important topic in physics, chemistry, material sciences, and biology. This is because finding the global minimum of a molecular system gives reliable predictions about how that system will be structured. Any field concerned with molecular behavior need first be concerned with that system’s structure, as structure can predict function. For example, Li & Scheraga developed the basin hopping method as a way to predict how proteins would fold, based off their primary amino acid chain structure. This is a computationally expensive task due to the number of amino acids and complex molecular interactions that occur between them before, during and after the folding process. This idea of predicting structure is not limited to biological molecules, but is also widely used to predict crystalline structures of both known and novel materials (Lyakhov et al., 2013).

By developing optimally efficient methods using LJ clusters as model systems, the road has been paved for researchers to explore and predict a wide range of problems more intricate problems, from how polymers self-assemble, to understanding properties of network forming fluids, and several topics in between.

**References**

Doye, J. P., Miller, M. A., & Wales, D. J. (n.d.). Lennard-Jones Clusters. Retrieved November 27, 2017, from http://doye.chem.ox.ac.uk/research/forest/LJ.html

Li, Z., & Scheraga, H. A. (1987 ). Monte Carlo-minimization approach to the multiple-minima problem in protein folding. *Proc National Academy Science U S A.* *84*(19), 6611-6615. doi:10.1007/springerreference\_72507

Lyakhov, A. O., Oganov, A. R., Stokes, H. T., & Zhu, Q. (2013). New developments in evolutionary structure prediction algorithm USPEX. Computer Physics Communications, 184(4), 1172-1182. doi:10.1016/j.cpc.2012.12.009

Miller, M., PhD. (n.d.). Mark Miller's Research Page. Retrieved November 10, 2017, from https://www.dur.ac.uk/research/directory/staff/?id=11538

Schönborn, S. E., Goedecker, S., Roy, S., & Oganov, A. R. (2009). The performance of minima hopping and evolutionary algorithms for cluster structure prediction. *The Journal of Chemical Physics,* *130*(14), 144108. doi:10.1063/1.3097197

Scipy.optimize.basinhopping. (n.d.). Retrieved November 10, 2017, from https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.basinhopping.html

Wales, D. J., & Doye, J. P. (1997). Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *The Journal of Physical Chemistry A,* *101*(28), 5111-5116. doi:10.1021/jp970984n

Wales, D. J., Munro, L. J., & Doye, J. P. (1996). What can calculations employing empirical potentials teach us about bare transition-metal clusters? *Journal of the Chemical Society, Dalton Transactions,* (5), 611. doi:10.1039/dt9960000611

Wales Group. (n.d.). Table of Lennard-Jones Clusters Minimum Values. Retrieved November 27, 2017, from http://doye.chem.ox.ac.uk/jon/structures/LJ/tables.150.html

**Appendix**

*Define the function, energy calculations, and initial position*

import numpy as np

def LJ(r):

r6 = r\*\*6

r12 = r6\*r6

return 4\*(1/r12 - 1/r6)

def total\_energy(positions):

E = 0

N\_atom = int(len(positions)/3)

#positions = [x0, y0, z0, x1, y1, z1, ..... , xn, yn, zn]

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]

#print('pos1: ', pos1)

#print('pos2: ', pos2)

dist = np.linalg.norm(pos1-pos2)

#print(i,j, dist)

E += LJ(dist)

return E

def init\_pos(N, L=5):

return L\*np.random.random\_sample((N\*3,))

*Call basin hopping algorithm and set parameters*

from scipy.optimize import basinhopping

N\_atom = 19

pos = init\_pos(N\_atom)

res = basinhopping(total\_energy, pos, niter=50, T=1.0, stepsize=2.7, disp=True)

*For Graphs*

import pandas as pd

from pandas import ExcelWriter

from pandas import ExcelFile

import matplotlib.pyplot as plt

df = pd.read\_excel('C:\\Users\\Katie\\Documents\\School\\Physics\\N19\_T1\_S2.7.xlsx')

#print (df.columns)

MinEnergy = df['energy']

plt.plot(MinEnergy, color="darkturquoise")

plt.xlabel('Number of Trials')

plt.ylabel('Energy')

plt.title('Basin-Hopping Energy Mins as a Function of Steps for N=19, T=1, S=2.7')

plt.savefig('C:\\Users\\Katie\\Documents\\School\\Physics\\N19\_T1\_S2.7.png')

plt.show()