Simulated Annealing

Han Gong

April 2021

Abstract

Simulated Annealing (SA) is a probabilistic technique for approximating the global extremum of a given function. This technique is most useful where the function in question has multiple local extrema such as the $f(x) = x^2 - cos(4\pi x)$ and $g(x) = cos(x) + cos(\sqrt{2}x) + cos(\sqrt{3}x)$ as shown in Exercise 10.10 in our text. For $f(x)$, the global minimum is $(0, -1)$. With our simulated annealing algorithm, our calculated minimum is found at $(-3.487 \times 10^{-5}, -0.999)$ in $0.062s$. The percent error in our minimum is $9.721 \times 10^{-6}\%$. For $g(x)$, the global minimum is $(15.959, -2.612)$, the calculated global minimum is found at $(15.959, -2.613)$ in $0.725s$. The percent error in our minimum is $6.684 \times 10^{-5}\%$. We expanded further in a by applying simulated annealing to 2 dimensional function such as the Rastrigin function. We also applied this into some physical problems such as determine the minimum inter-molecular potential between two Argon atoms. We also determined the maximum intensity of Raman spectroscopy of Pyrite crystal (FeS_2) by SA. All the calculation is done by my Windows 10 laptop with I7-4900MQ with 32G of RAM.

Introduction

Global optimization problems are a series of problems that requires to find the global mimima or maxima of a function or a set of functions. These problems have many real world connections, such as minimize the circuit length in a electrical unit, curve fitting in physics, chemistry, biology and so on.

There are many search algorithms that solves the global optimization problems. There is hill-climbing algorithm where it never makes “downhill” moves toward states with lower value (or higher cost) is guaranteed to be incomplete, because it can get stuck on a local maximum. There is also the random walks such as Monte Carlo approaches. In contrast, a purely random walk—that is, moving to

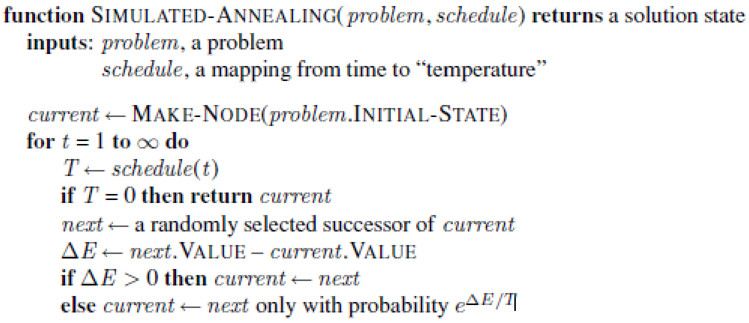
a successor chosen uniformly at random from the set of successors—is complete but extremely inefficient due to its randomness. One is efficient, but does not guarantee the to find the global maximum, therefore its search is not complete. The other is extremely inefficient, but it allows to find the global maximum, therefore the search is complete. Simulated annealing combines the benefits of these two algorithms by combining the hill climbing with the random walks which yields both efficiency and the completeness of the two.
We implemented the simulated annealing algorithm in Python. Solving one dimensional functions that have many local minimums and try to find the global minimum in a given region. We further adapted our algorithm to be able to solve 2 dimensional functions and non-continuous data entries. We used this algorithm to solve for equilibrium distance between two atoms and to investigate the maximum intensity in the Raman Spectrum for a specific compound.

Theory and Model

Simulated Annealing acquired its name from the annealing from metallurgy. Annealing is a process that used to temper or harden the metals by heating them to a temperature and cool down gradually, therefore it allows the material to reach a low energy crystalline state. Simulated Annealing simulates this process. Imagine a ping-pong ball on an uneven, hilly surface. If we just let the ball fall on the surface, it will settle at a local minimum. In order to let the ball to fall on the lowest point on this surface. Like annealing, we keep shaking and gradually lower the intensity of the shaking. That allows the ball to escape the local minimum but not dislodge it from the global minimum. In terms of the code, we the surface is our function in question. The ball is our search progress. The general concept is starting our search in a random point on the function. To simulate the striking the iron in the process of annealing, we implemented a method called $Gaussian\_neighbor(x, T)$. Using Gaussian random number, it will generate a random neighboring points of the current location. The range of the neighboring point will directly proportional to the temperature. We look for a Gaussian neighbor of the current point to escape possible local minimum. We compare the values and look for the downhill steps. As the search goes on, we gradually decrease the range of our Gaussian neighbor. Eventually we will reach the end of the search either the temperature has reached zero or the Gaussian neighbor with given energy cannot find any states that will have a lower energy.

Program

Here is the list of functions we defined for Simulated Annealing.



```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
      $\Delta E$ ← next.VALUE – current.VALUE
      if $\Delta E > 0$ then current ← next
      else current ← next only with probability $e^{\Delta E/T}$
```

Figure 1: Simulated Annealing Pseudo Code from A Modern Approach to Artificial Intelligence.[1]

```
def f(x)
...
def bounds(x)
...
def monteCarlo_Start():
...
def accept_prob(energy, new_energy, temperature)
...
def energy_function(x):
...
def gaussian_neighbour(x, fraction)
...
def  cooling_schedule()
...
def simulated_Annealing(monteCarlo_Start, energy_function,
    gaussian_neighbour, acceptance, cooling_schedule, maxsteps,
    testing=True)
...
```

$f(x)$ defines the core function we are applying the Simulated Annealing process.
$bounds(x)$ enforces the search bounds of the given intervals in our searching algorithm.
$monteCarlo\_Start()$ returns a random point on the search intervals as the starting point of our search.
$accept\_Prob()$ determines if the new energy generated by the Gaussian neighbor function is acceptable with the current temperature.
$energy\_function()$ is a helper method that will feed the function into the searching algorithm.
$gaussian\_neighbor()$ uses the Gaussian random number to find a random neighbour of the current state in relation of the current temperature.

$cooling\_schedule()$ gives the cooling factor of the temperature.
$simulated\_Annealing(...)$ uses the helper methods above and return the search result and the search progress.

$monteCarlo\_Start(), energy\_function(state), gaussian\_neighbor(state, fraction),$ $simulated\_Annealing(...)$ need to be modified to the two dimensional function for applying the Simulated Annealing in Rastrigin function.

Similarly, $monteCarlo\_Start(), energy\_function(state),$ $gaussian\_neighbor(state, fraction)$ needs to modified to be able to apply Simulated Annealing to the data set in Pyrite's Raman Spectrum.

Results

10.10a

Using Simulated Annealing, find the global minimum of the function with given interval:

$$f(x) = x^2 - cos(4\pi x) \quad (-5 \leq x \leq 5) \tag{1}$$

Where the true global minimum is labeled red in Figure 1 and it is found at:

$$(0, -1)$$

Applying the Simulated Annealing algorithm, in about 0.062s, we found our calculated global minimum at:

$$(-3.486870777251222 \times 10^{-5}, -0.9999999027863399) \approx (-3.48 \times 10^{-5}, -0.999)$$

As we can see in the Figure 1, the calculated minimum and the true minimum is very close. The percent error in the location of the minimum (x value) is 0.0034% and the error of the value of the minimum ($f(x)$ value) is 9.72×10^{-6}%.

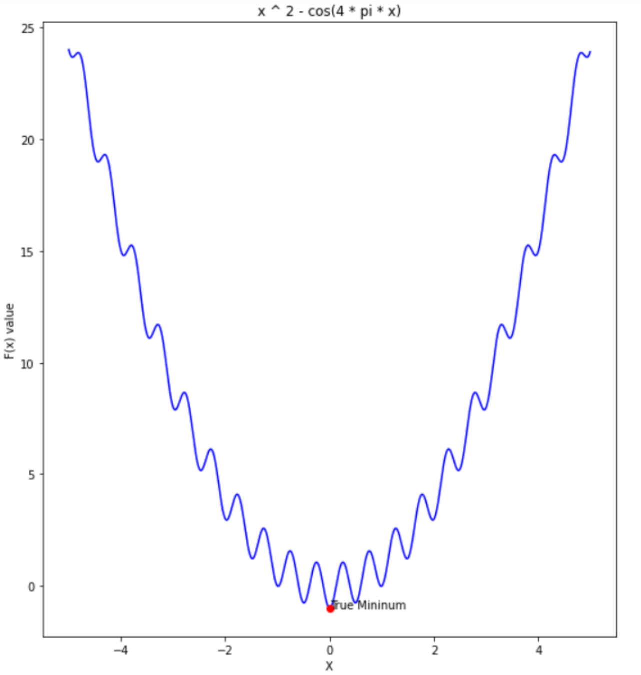


Figure 2: The graph for $f(x) = x^2 - cos(4\pi x)$ from $-5 \leq x \leq 5$.

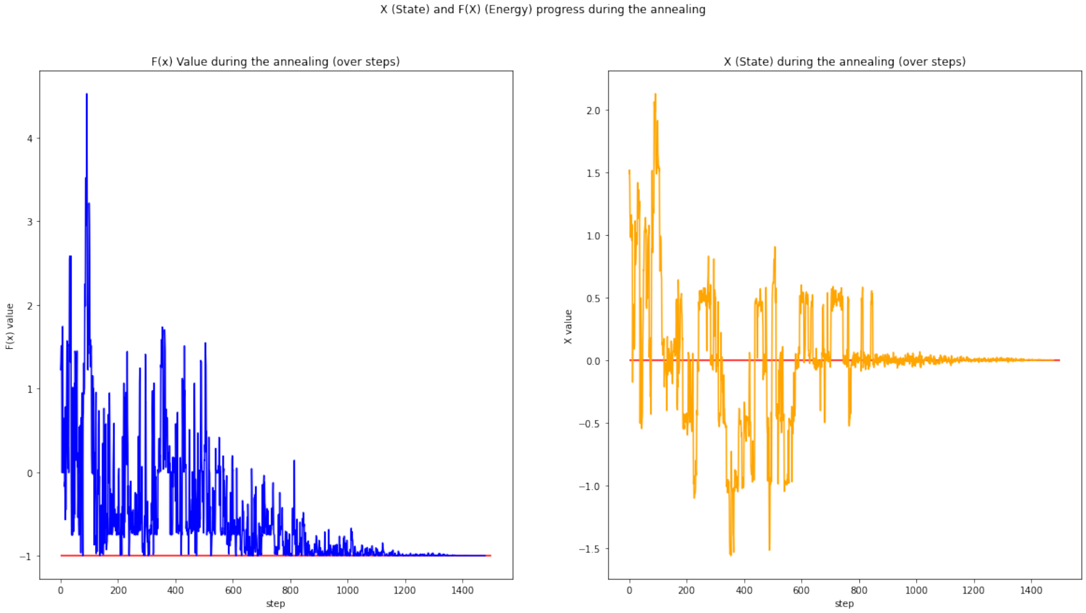


Figure 4: The annealing process of $f(x)$.

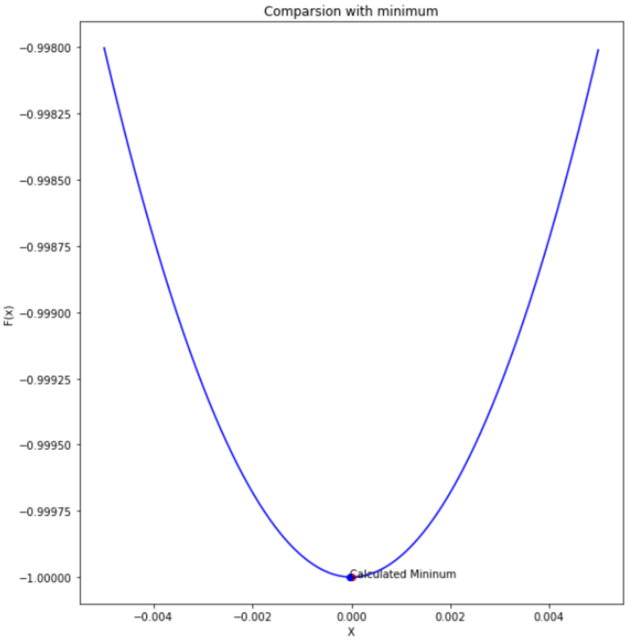


Figure 3: $f(x)$ in interval $(-0.005 \leq x \leq 0.005)$ True minimum and calculated minimum comparison.

As the Figure 3 shows, as the annealing process goes on, the energy $(f(x))$ values decreases gradually. We can see the "shaking" as the numerous peaks in the graph. The red lines in both graphs are the expected values for both $x, f(x)$.

10.10b

Again, using Simulated Annealing to find the minimum of the function $g(x)$ with given interval:

$$g(x) = cos(x) + cos(\sqrt{2}x) + cos(\sqrt{3}x) \tag{2}$$

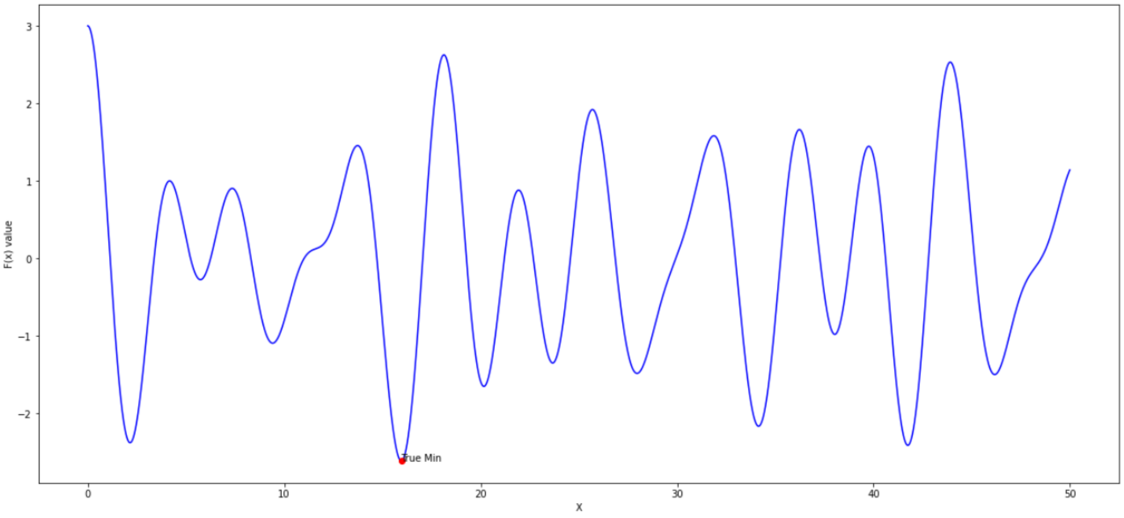


Figure 5: $g(x)$ $(0 \leq x \leq 50)$ with global minimum.

Where the global minimum with in the given interval is labeled red in Figure 4 and it is found at:

$$(15.\overline{95}, -2.61259)$$

Applying the Simulated Annealing algorithm, in about 0.725s, we found our calculated global minimum at:

$$(15.959734248803832, -2.6125882538306424) \approx (15.95, -2.61)$$

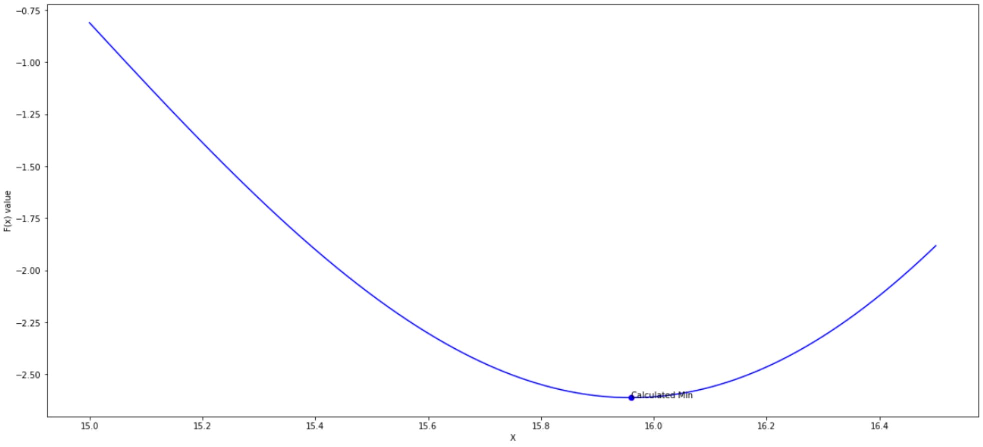


Figure 6: $g(x)$ $(15 \leq x \leq 16.4$ Global minimum and calculated global minimum comparison.

As we can see from Figure 5, our calculated minimum is very close to the true global minimum. The percent error in the location of the minimum (x value) is 0.000866% and the error of the value of the minimum ($g(x)$ value) is 6.68×10^{-5}%.

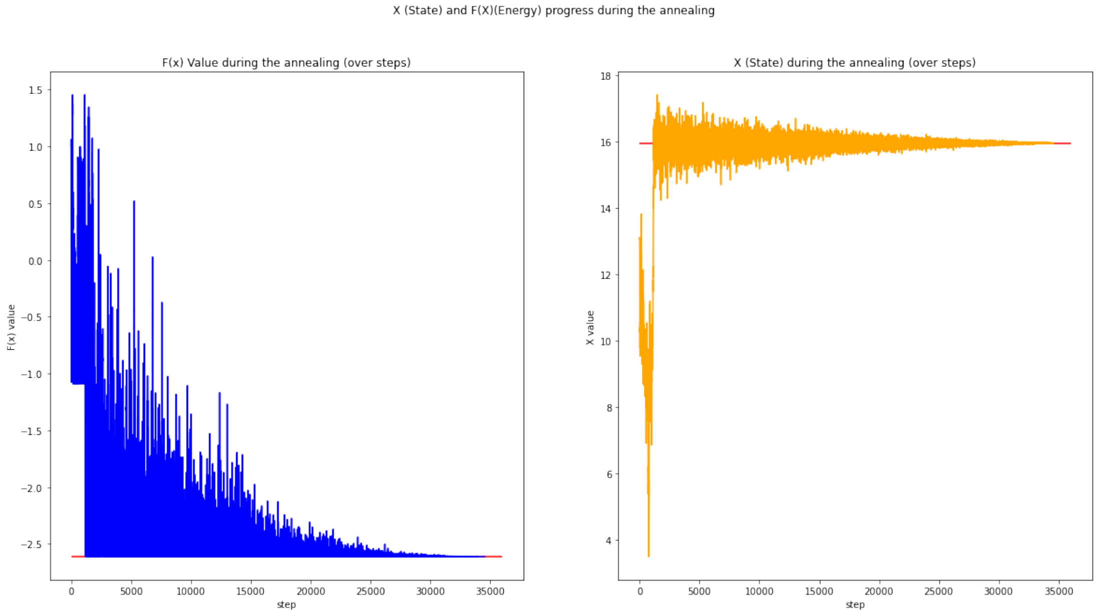


Figure 7: The annealing process of $g(x)$.

In comparison to $f(x)$, we see from Figure 6 that it takes much longer to find the global minimum for $g(x)$ than the previous function. We see from the graph on the right that our $g(x)$ value hesitated around -1 in the beginning but it quickly lowered to -2.6. The reason is that as we see the graph of $g(x)$, we see that there are multiple local minimums around -1, which takes time for the algorithm to process and identify them as local minimums.

Two Dimensional Annealing: Rastrigin Function

Rastrigin function[2], invented by the L.A. Rastrigin, is a function that is one of the benchmarks for optimization functions due to it's complex shape and numerous local minima. The function in question is the following:

$$R = x^2 + y^2 - cos(18x) - cos(18y) + 2 \tag{3}$$

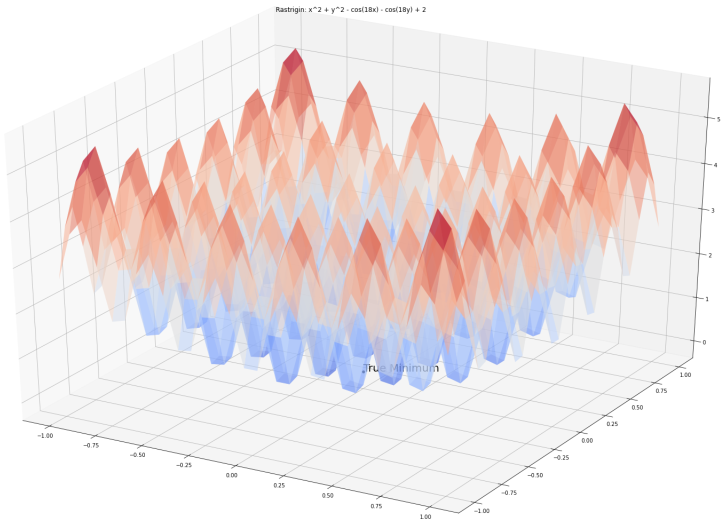


Figure 8: The graph of $R(x)$ in interval $(-1 < x < 1)$, $(-1 < y < 1)$.

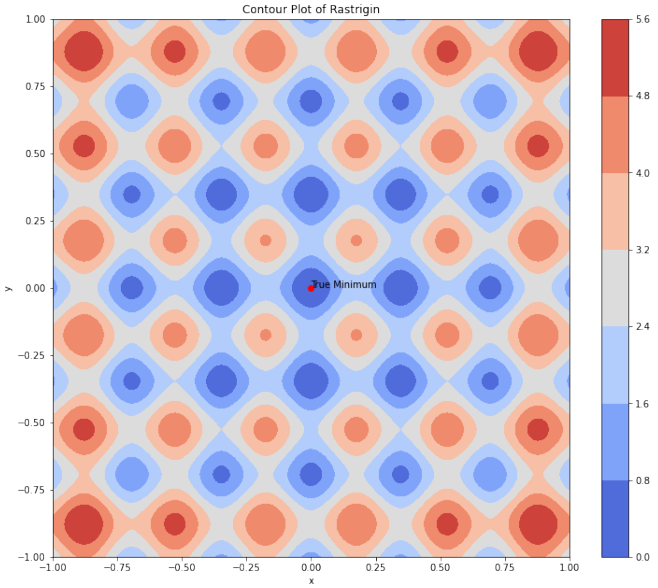


Figure 9: The contour plot of $R(x)$ in interval $(-1 < x < 1)$, $(-1 < y < 1)$.

As the Figure 7 shows, $R(x)$ and the other Rastrigin have the surface like a egg carton. We can see from the Figure 8, our function $R(x)$ has the global minimum at (0, 0, 0).
Applying the Simulated Annealing algorithm, in about 0.146s, we found our calculated global minimum at:

$$(-3.6229689922447766\times10^{-5}, 0.0014510406202214374, 0.00034339413950412734) \approx (-3.62\times10^{-5}, 0.00145, 0.00$$

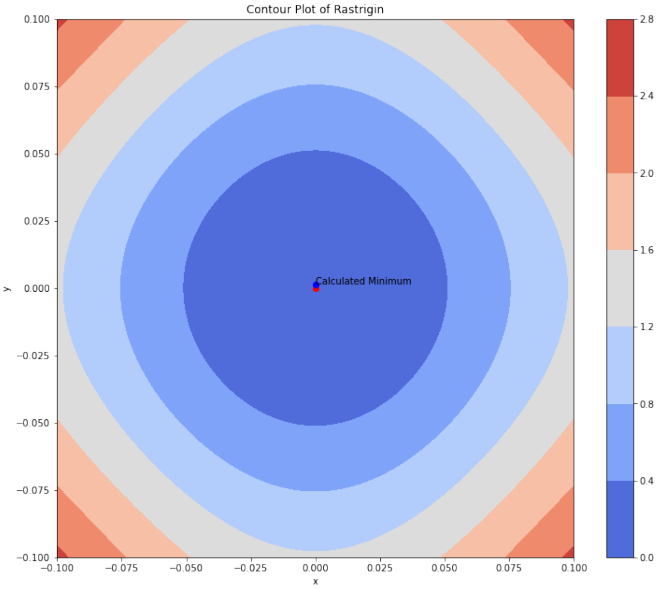


Figure 10: The contour plot of $R(x)$ with global minimum and calculated global minimum comparison.

The percent error of our calculate global minimum is 0.0343%

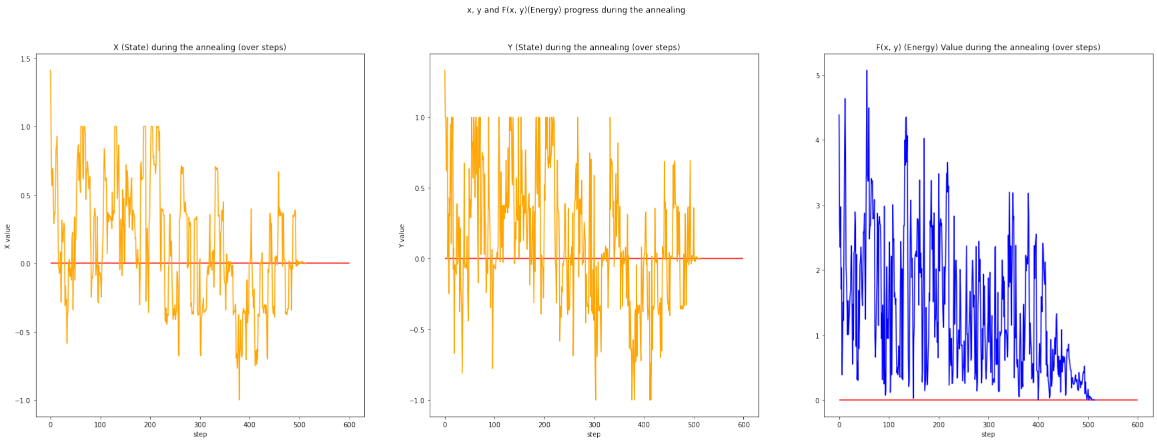


Figure 11: The annealing process of $R(x)$.

Like previous examples, one the right most graph, we see an overall trend where the energy is converging to the expected value (red line). However, we see even more fluctuations than previous examples. If we examine the contour plot of $R(x)$ in Figure 8, we see that there are 25 valleys with local minima. Therefore we see the algorithm is trying to decide which one has a lower value (energy).

Lennard-Jones Potential

Lennard-Jones Potential[3] describes the potential energy of interaction between two non-bonding atoms or molecules based on their distance of separation.

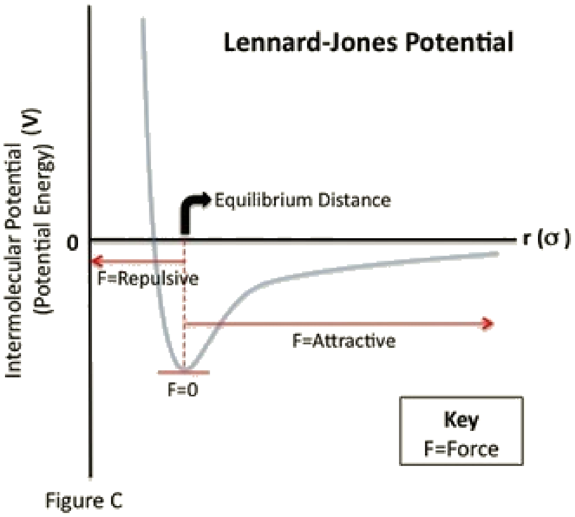


Figure 12: Lennard-Jones Potential[3].

The formula descirbes the Lennard-Jones Potential is given by:

$$V(r) = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}] \tag{4}$$

V : The intermolecular potential between the two atoms or molecules.
ϵ : The well depth and a measure of how strongly the two particles attract each other.
σ : The distance at which the intermolecular potential between the two particles is zero.
r : The distance of separation between both particles.

Given that the $\sigma = 3.40$ Angstroms and $\epsilon = 0.977kJ/mol$ for two Argon molecules. We have the expression of the Lennard-Jones potential V_{Ar}:

$$V_{Ar} = 4(0.977)[(\frac{3.40}{r})^{12} - (\frac{3.40}{r})^{6}] \tag{5}$$

The minimum of this function is where the equilibrium state between two particles is found.

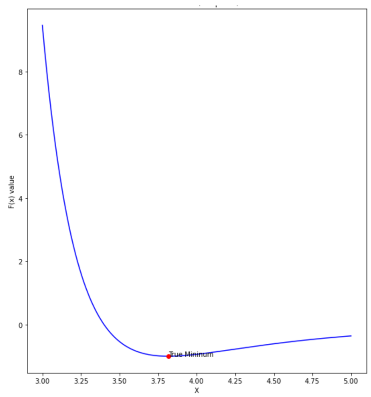


Figure 13: The graph of V_{Ar} in interval $3 < r < 5$.

The Figure 11 shows the graph of the Lennard-Jones Potential for 2 Argon with the global minimum labeled at:

$$(3.816, -0.997)$$

Applying the Simulated Annealing algorithm to find the distance between two Argon particles when they are in the equilibrium state. In about 0.081s, we found our calculated global minimum at:

$$(3.8182540260815117, -0.9969912918417237) \approx (3.81, -0.996)$$

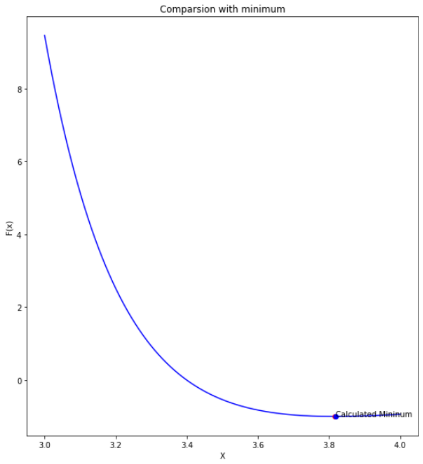


Figure 14: The graph of V_{Ar} with global minimum and calculated global minimum comparison.

The percent error of our calculate global minimum is 0.000873%

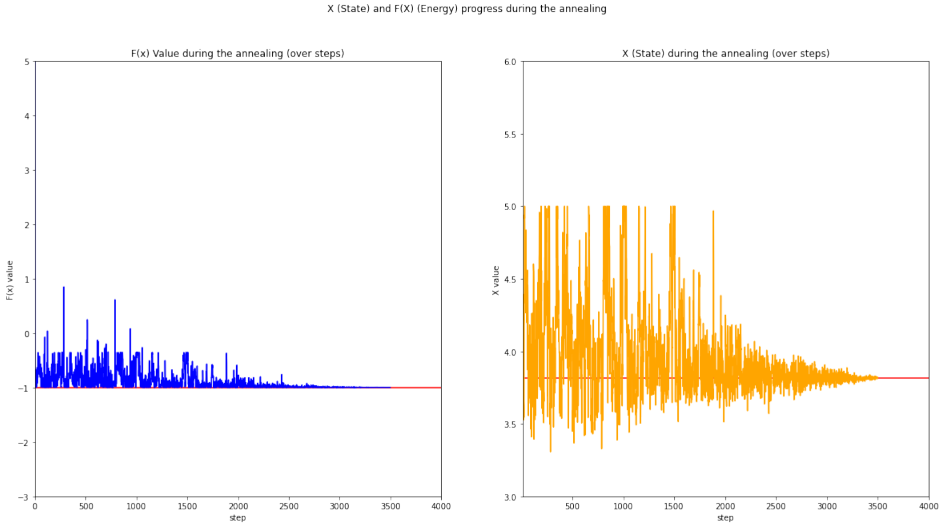


Figure 15: The annealing process of V_{Ar}.

Raman Spectroscopy of Pyrite Crystal

Raman Spectroscopy is a chemical analysis technique which provides insights about chemical structure, crystallinity and more without destroying the chemical itself. It is based upon the interaction of light with the chemical bonds within a material. A Raman spectrum features a number of peaks, showing the intensity and wavelength position of the Raman scattered light. Each peak corresponds to a specific molecular bond vibration.

The crystal we are studying is Pyrite. Pyrite is a brass-yellow mineral with a bright metallic luster. It has a chemical composition of iron sulfide FeS_2 and is the most common sulfide mineral. It forms at high and low temperatures and occurs, usually in small quantities, in igneous, metamorphic, and sedimentary rocks worldwide. It has a striking similarity with Gold therefore it is nicknamed ”Fool’s Gold”[4].

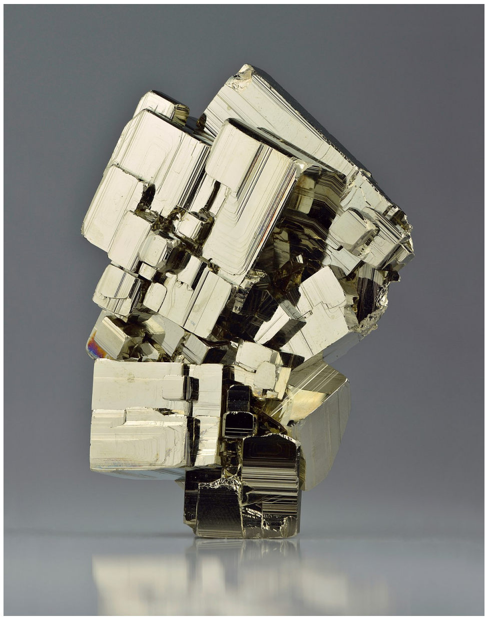


Figure 16: Pyrite cubic crystals.[5]

We gathered Raman Spectrm for Pyrite in RRUFF[6] website. The plot of our data is shown below:

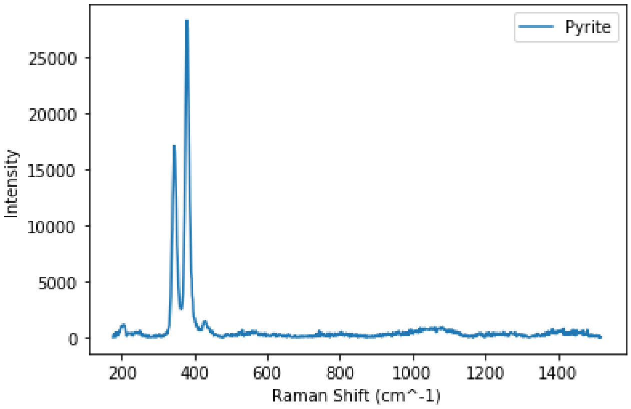


Figure 17: Raman Spectrm for Pyrite.

We did have to modify our algorithm to accept a series of data points instead of a continuous function. We are able to find the maximum intensity at (379, 28249.875) in 0.169s. The true maximum of the data set is at (379, 28249.875). Since we are looking for a data point in a data set, there is no error in this case.

Conclusion

The simulated annealing algorithm we implemented is able to find the global minimums for the two given function in Exercise 10.10. For 10.10a, the calculated global minimum is found at $(-3.487 \times 10^{-5}, -0.999)$ and the percent error in the calculated minimum is 9.72×10^{-6}. For 10.10b, the calculated global minimum is found at $(15.959, -2.613)$ and the percent error in the calculated minimum is 6.684×10^{-5}. Both of the result calculated within one second. The expansion, we modified our algorithm to fit two dimensional function such as the Rastrigin function. Within $0.2s$, we find the global minimum and the error is 0.034%. One of our physical application is to find the equilibrium distance between two Argon atom using the Lennard-Jones potential. In $0.081s$, we find the equilibrium distance is 3.818 Angstroms and the percent error in the equilibrium is 0.0008%. For the Raman spectroscopy of Pyrite, we achieve our result $(379, 28249.875)$ in 0.169s. One possible improvement of this program is to implement a loop to run this algorithm multiple times to achieve a better average result. Due to the randomness of the Monte Carlo starting point, we can have some outlying results. The will be able to greatly reduce the possibility of the outlying results.

References

1. Russell. S, Norvig. P. Artificial Intelligence, A Modern Approach. ISBN: 978-0-13-604259-4

2. Rastrigin function https://en.wikipedia.org/wiki/Rastrigin_function

3. Lennard-Jones Potential https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Physical_Properties_of_Matter/Atomic_and_Molecular_Properties/Intermolecular_Forces/Specific_Interactions/Lennard-Jones_Potential

4. Pyrite https://geology.com/minerals/pyrite.shtml#:~:text=Pyrite%20is%20a%20brass%2Dyellow,metamorphic%2C%20and%20sedimentary%20rocks%20worldwide

5. Pyrite https://en.wikipedia.org/wiki/Pyrite

6. RRUFF Pyrite https://rruff.info/Pyrite/R050190