# Simulated-Annealing

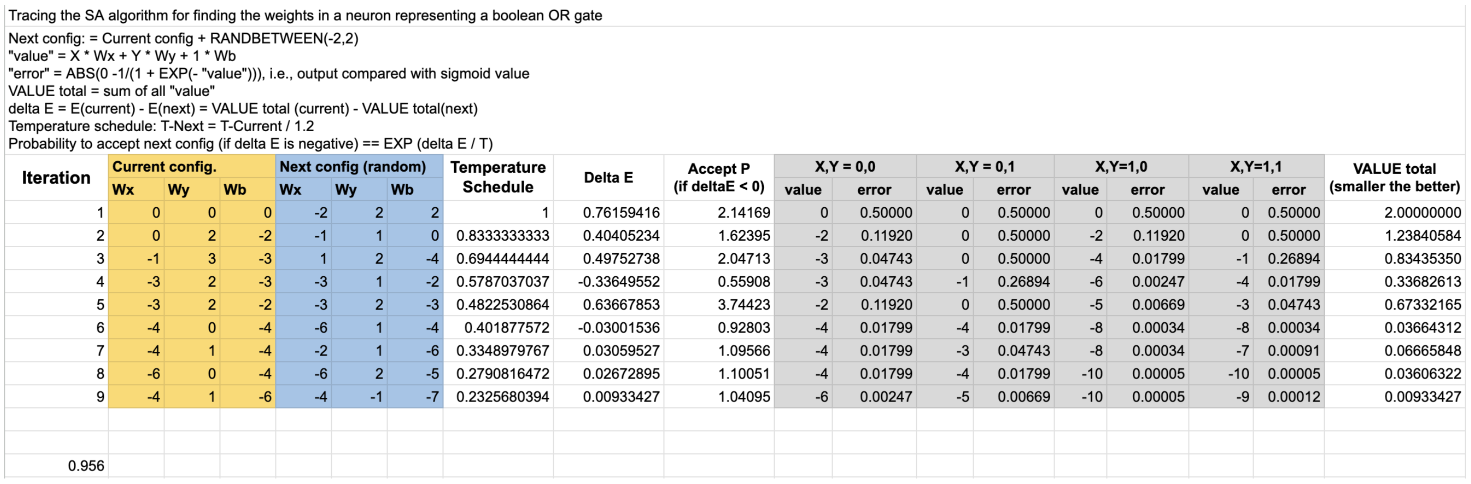
Alexander Garcia

October 2024

## 1 Introduction

This project implements the Simulated-Annealing algorithm for finding weights in a neuron representing a boolean OR gate. My goal is to minimize the objective function by finding the optimal values for the input weights Wx, Wy, and Wb. Temperature is a core part of the algorithm that separates it from the Hill-Climbing algorithm and prevents it from being stuck on shoulders, local minimums, and local maximums. Three different temperature schedules were chosen: 1.2 (same as the example in class), 1.5, and 1.005. Each temperature schedule affected the quality of solutions accepted as well as the maximum iterations before accepting a final solution.

## 2 Google Sheet Tracing SA



Tracing the SA algorithm for finding the weights in a neuron representing a boolean OR gate

Next config: = Current config + RANDBETWEEN(-2,2)
"value" = X * Wx + Y * Wy + 1 * Wb
"error" = ABS(0 -1/(1 + EXP(- "value"))), i.e., output compared with sigmoid value
VALUE total = sum of all "value"
delta E = E(current) - E(next) = VALUE total (current) - VALUE total(next)
Temperature schedule: T-Next = T-Current / 1.2
Probability to accept next config (if delta E is negative) == EXP (delta E / T)

| Iteration | Current config. | | | Next config (random) | | | Temperature Schedule | Delta E | Accept P (if deltaE < 0) | X,Y = 0,0 | | X,Y = 0,1 | | X,Y=1,0 | | X,Y=1,1 | | VALUE total (smaller the better) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Wx | Wy | Wb | Wx | Wy | Wb | | | | value | error | value | error | value | error | value | error | |
| 1 | 0 | 0 | 0 | -2 | 2 | 2 | 1 | 0.76159416 | 2.14169 | 0 | 0.50000 | 0 | 0.50000 | 0 | 0.50000 | 0 | 0.50000 | 2.00000000 |
| 2 | 0 | 2 | -2 | -1 | 1 | 0 | 0.8333333333 | 0.40405234 | 1.62395 | -2 | 0.11920 | 0 | 0.50000 | -2 | 0.11920 | 0 | 0.50000 | 1.23840584 |
| 3 | -1 | 3 | -3 | 1 | 2 | -4 | 0.6944444444 | 0.49752738 | 2.04713 | -3 | 0.04743 | 0 | 0.50000 | -4 | 0.01799 | -1 | 0.26894 | 0.83435350 |
| 4 | -3 | 2 | -3 | -3 | 1 | -2 | 0.5787037037 | -0.33649552 | 0.55908 | -3 | 0.04743 | -1 | 0.26894 | -6 | 0.00247 | -4 | 0.01799 | 0.33682613 |
| 5 | -3 | 2 | -2 | -3 | 2 | -3 | 0.4822530864 | 0.63667853 | 3.74423 | -2 | 0.11920 | 0 | 0.50000 | -5 | 0.00669 | -3 | 0.04743 | 0.67332165 |
| 6 | -4 | 0 | -4 | -6 | 1 | -4 | 0.401877572 | -0.03001536 | 0.92803 | -4 | 0.01799 | -4 | 0.01799 | -8 | 0.00034 | -8 | 0.00034 | 0.03664312 |
| 7 | -4 | 1 | -4 | -2 | 1 | -6 | 0.3348979767 | 0.03059527 | 1.09566 | -4 | 0.01799 | -3 | 0.04743 | -8 | 0.00034 | -7 | 0.00091 | 0.06665848 |
| 8 | -6 | 0 | -4 | -6 | 2 | -5 | 0.2790816472 | 0.02672895 | 1.10051 | -4 | 0.01799 | -4 | 0.01799 | -10 | 0.00005 | -10 | 0.00005 | 0.03606322 |
| 9 | -4 | 1 | -6 | -4 | -1 | -7 | 0.2325680394 | 0.00933427 | 1.04095 | -6 | 0.00247 | -5 | 0.00669 | -10 | 0.00005 | -9 | 0.00012 | 0.00933427 |
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| 0.956 | | | | | | | | | | | | | | | | | | |

As seen in the figure, twice the algorithm accepted a solution that was considered a bad move. However, this solution was still accepted due to the random number generated in A16 being less than our acceptance probability.

## 3 Temperature schedules

Below are the figures of all temperature schedules used.

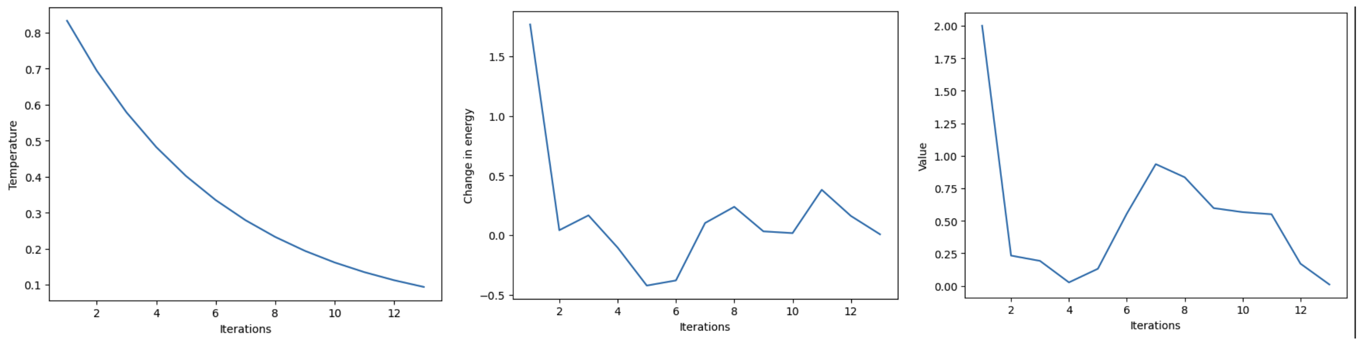


Figure 1: Temperature schedule 1.2

This temperature rate cools rather quickly and results in a lower amount of total iterations as well as less variability in accepted answers

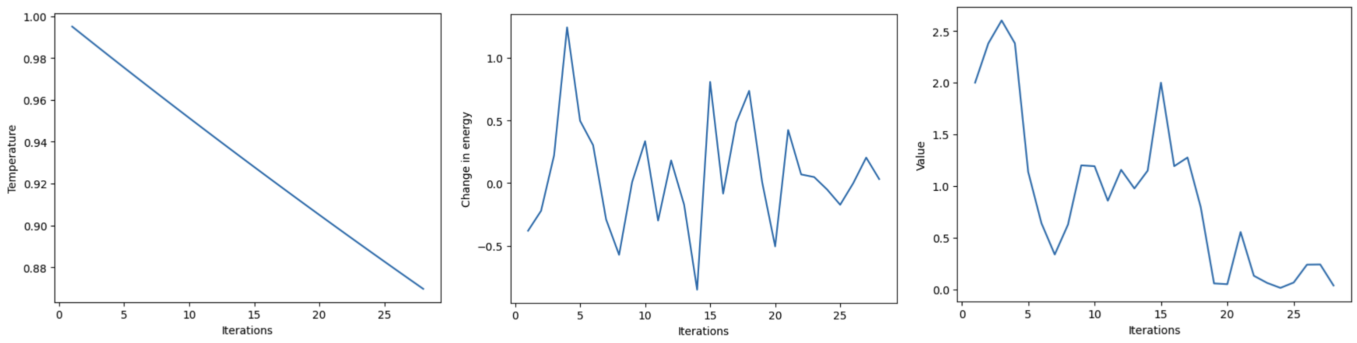


Figure 2: Temperature schedule 1.005

Here I notice an increase in iterations as the temperature decreases at a slower rate. The range of accepted solutions also increases as the algorithm is less strict on accepted answers.

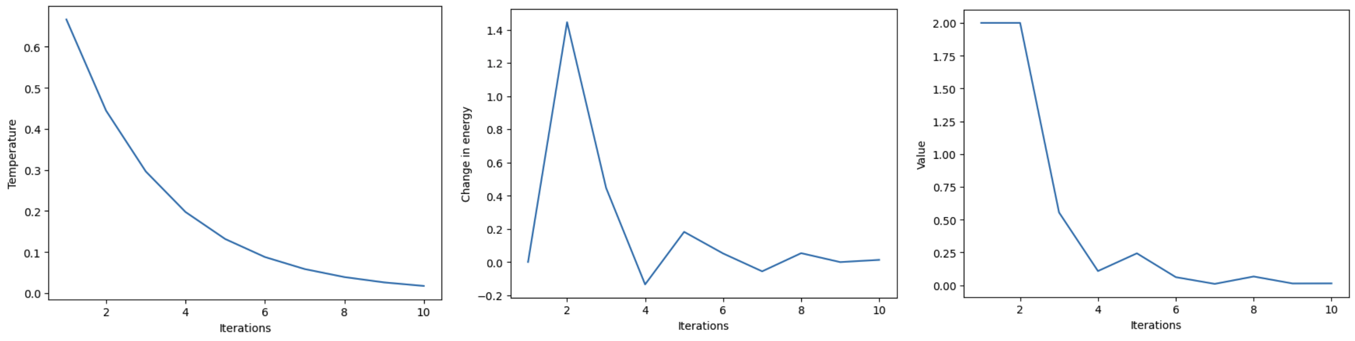


Figure 3: Temperature schedule 1.5

This temperature schedule is similar to 1.2 as it has far less variety in solutions. The total iterations also decreased relative to a temperature schedule of 1.005 as the rapid temperature decrease causes the algorithm to be strict earlier.

## 4 Closing remarks

Overall this was a fun challenging project. No other resources were used apart from lectures on simulated annealing and the pseudo code from the book. The most difficult part was watching lecture videos of the algorithm and trying to figure out how to implement that in code. Especially when discarding the next config until the random number generated was less than the acceptance probability as the excel sheet nor the pseudo code give an example of how to implement in code.