```
import machine
import math
import numpy as np
import sys

# the distance matrix to be used in the
# traveling salesman problem
distances = np.matrix([
    [0,10,20,5,18],
    [0,0,15,32,10],
    [0,0,0,25,16],
    [0,0,0,0,35],
    [0,0,0,0,0]
])
# making it symmetric for ease of use
distances = distances + distances.T

if __name__ == '__main__':

    if len(sys.argv) != 3:
        T = 5000
        h_charge = 0.5
        b_charge = -0.2
    else:
        T = int(sys.argv[1])
        h_charge = float(sys.argv[2])
        b_charge = float(sys.argv[3])

    # create a boltzmann machine with the hamiltonian_error_charge as
the weight
    # assigned to any edge that will violate the hamiltonian, and a
bias_charge
    # as the weight assigned to the loop edges (points the node to
itself) --
    # this effectively acts as the bias term in the consensus function
    b = machine.boltzmann(hamiltonian_error_charge=h_charge,
bias_charge=b_charge)

    # create the network using the distances as the foundation for the
weight matrices
    b.create_network(distances)

    # anneal the network starting at temperature T and a schedule
function that
    # decrements T after every cycle
    machine.anneal(b,T=T,schedule=lambda T: math.log10(T) if T > 100
else 0.1)
```