TSP Simulated Annealing - Guilherme de Abreu Lima Buitrago Miranda - 2018054788

August 1, 2021

1 Introdução à Física Estatística Computacional

1.1 O Problema do Caixeiro Viajante: Solução por Simulated Annealing

Aluno: Guilherme de Abreu Lima Buitrago Miranda

Matrícula: 2018054788

1.1.1 Imports

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import random

plt.style.use('seaborn-colorblind')
plt.ion()
```

1.1.2 Funções

As funções abaixo foram extraídas do enunciado do exercício ou criadas por mim.

[2]:
```
def calc_dist(x, y, N):
    dist = np.eye(N)
    for i in range(N):
        for j in range(N):
            dist[i][j] += np.sqrt(((x[i] - x[j]) ** 2) + ((y[i] - y[j]) ** 2))

    return dist
```

[3]:
```
def calc_ener(cam, dist, N):
    ener = 0
    for i in range(N-1):
        ener += dist[cam[i], cam[i+1]]

    ener += dist[cam[0], cam[N-1]]
    return ener
```

[4]:
```python
def escolhe_caminho(cam, N):
    ncam = np.zeros(N,dtype=np.int16)
    i=np.random.randint(N)
    j=i
    while j==i: # escolhe j de forma que
        j=np.random.randint(N)
    if i>j:
        ini = j
        fim = i
    else:
        ini = i
        fim = j
    for k in range(N):
        if k >= ini and k <= fim:
            ncam[k] = cam[fim-k+ini]
        else:
            ncam[k] = cam[k]

    return ncam, ini, fim
```

[5]:
```python
def calc_dist_cam(cam, ncam, ini, fim, N, dist):
    esq = ini-1
    if esq<0:
        esq = N-1
    _dir = fim+1
    if _dir>N-1:
        _dir = 0
    de = -dist[cam[esq], cam[ini]] - dist[cam[_dir], cam[fim]] +␣
↪dist[ncam[esq],ncam[ini]] + dist[ncam[_dir],ncam[fim]]
    return de
```

[6]:
```python
def mcstep(cam, N, dist, ener, T):

    ncam, ini, fim = escolhe_caminho(cam, N)
    de = calc_dist_cam(cam, ncam, ini, fim, N, dist)

    if de < 0 or random.random() < np.exp(-de/T):
        ener = calc_ener(ncam, dist, N)
        cam = ncam

    return ener, cam
```

[7]:
```python
def execute_all(n):
    x = np.random.rand(n)
    y = np.random.rand(n)

    dist = calc_dist(x, y, n)
```

```
    ener_return = []
    Ts_return = []
    cams_return = []
    xys = []

    for t in range(len(Ts)):

        ener_ = []
        Ts_ = []
        cams = []

        T = Ts[t]

        cam = np.arange(n, dtype=np.int16)
        ener = calc_ener(cam, dist, n)

        while T > Tf:
            eners = []
            for i in range(Niter):
                ener, cam = mcstep(cam, n, dist, ener, T)
                eners.append(ener)

            ener_.append(np.mean(eners))
            cams.append(cam)
            Ts_.append(T)
            T *= dt

        ener_return.append(ener_)
        Ts_return.append(Ts_)
        cams_return.append(cams[-1])
        xys.append([x, y])

    return xys, Ts_return, ener_return, cams_return
```

[8]:
```
def plot(Ns, pairs, Ts, Ts_, eners, last_cams):
    for i in range(len(Ns)):
        for j in range(len(Ts)):
            x, y = pairs[i][j]
            T = Ts_[i][j]
            ener =  eners[i][j]
            cam = last_cams[i][j]

            plt.plot(T, ener)
            plt.title("Média de Energia por Temperatura. N = " + str(Ns[i]) + "␣
↪T = " + str(Ts[j]))
            plt.show()
```

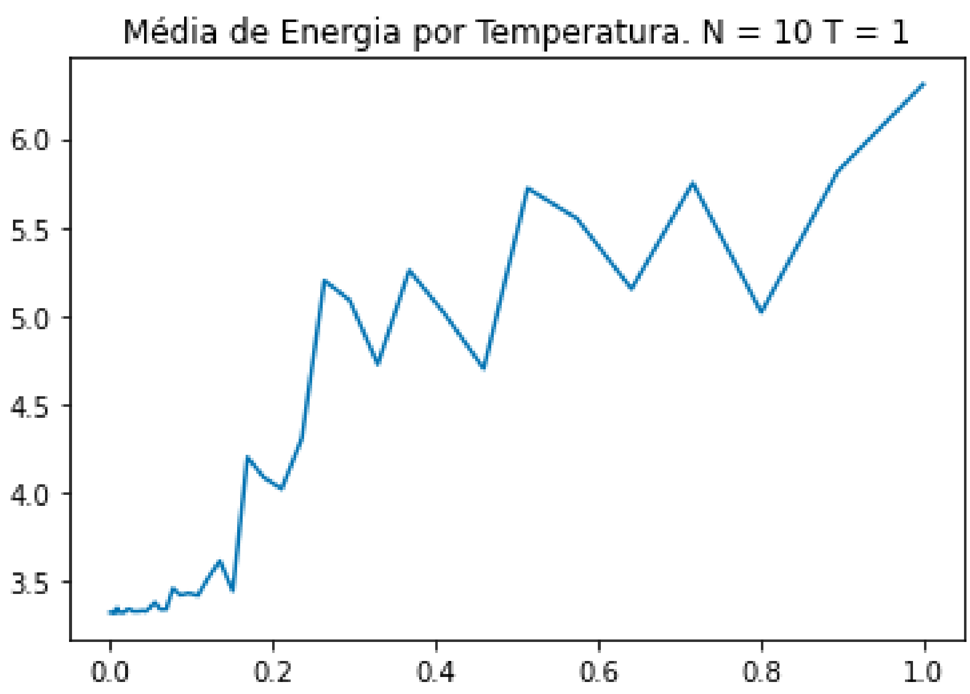
```python
            plt.scatter(x, y)
            plt.plot(x[cam], y[cam])
            plt.title("Caminho final. N = " + str(Ns[i]) + " T = " + str(Ts[j]))
            plt.show()
```

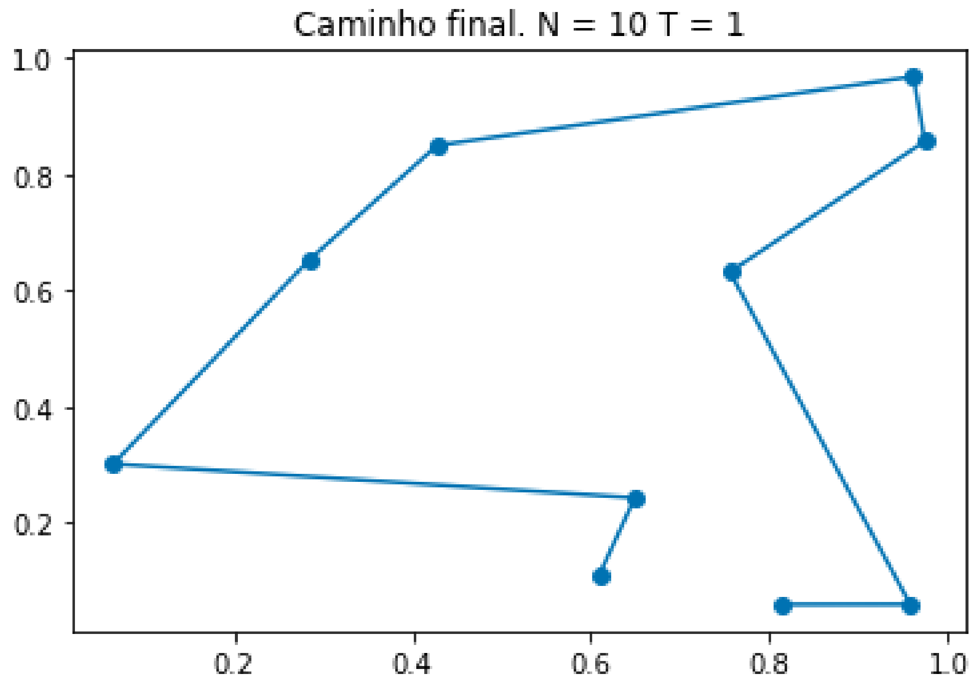
[9]:

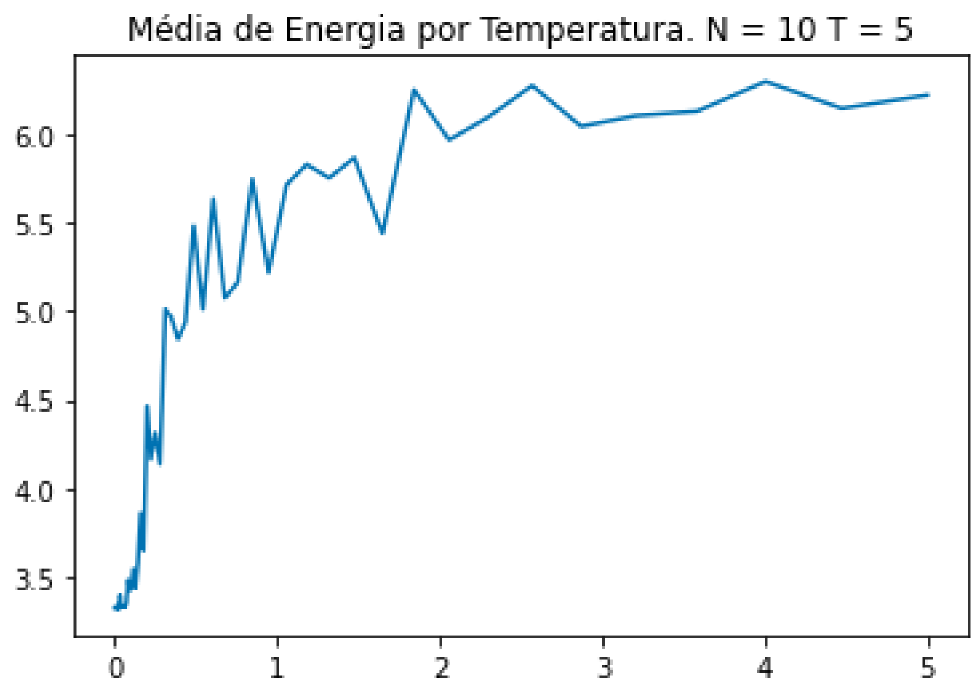
```python
Ns = [10, 25, 40, 60]
Ts = [1, 5, 10]

Niter = 100
dt = 0.895
#Tf = 0.00745
Tf= 0.001

pairs = []
Ts_ = []
eners = []
last_cams = []

for n in Ns:
    pair, t, ener, last_cam = execute_all(n)
    pairs.append(pair)
    Ts_.append(t)
    eners.append(ener)
    last_cams.append(last_cam)
```

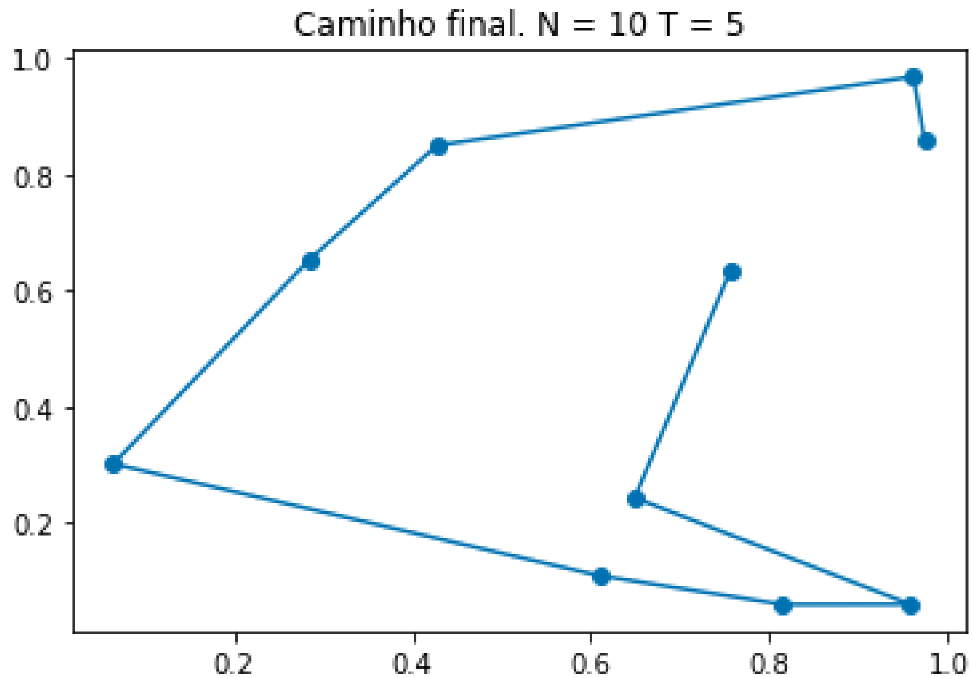
[10]:

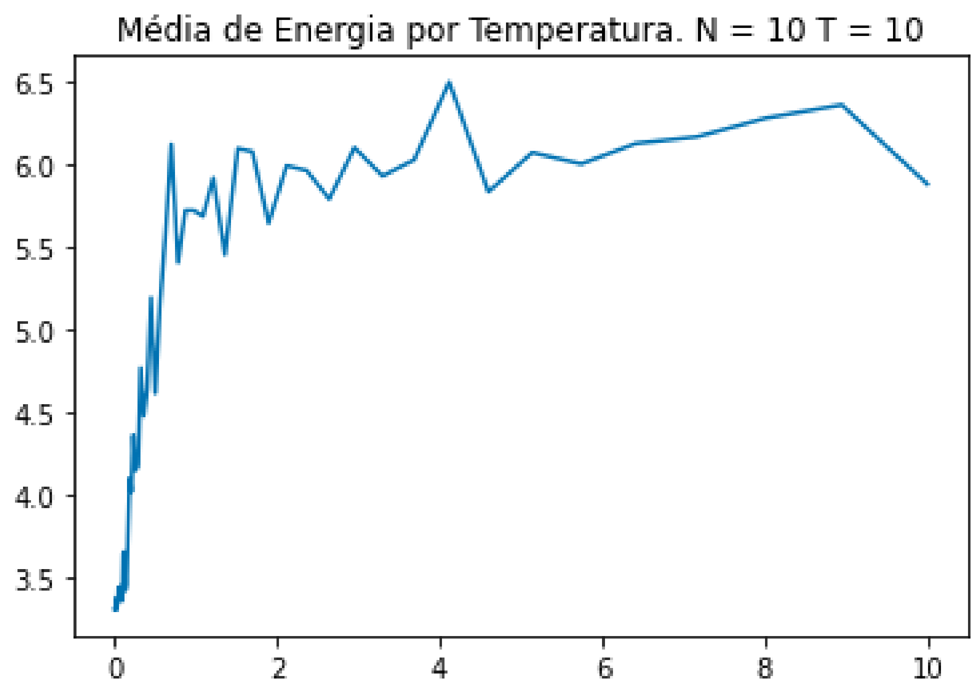
```python
plot(Ns, pairs, Ts, Ts_, eners, last_cams)
```

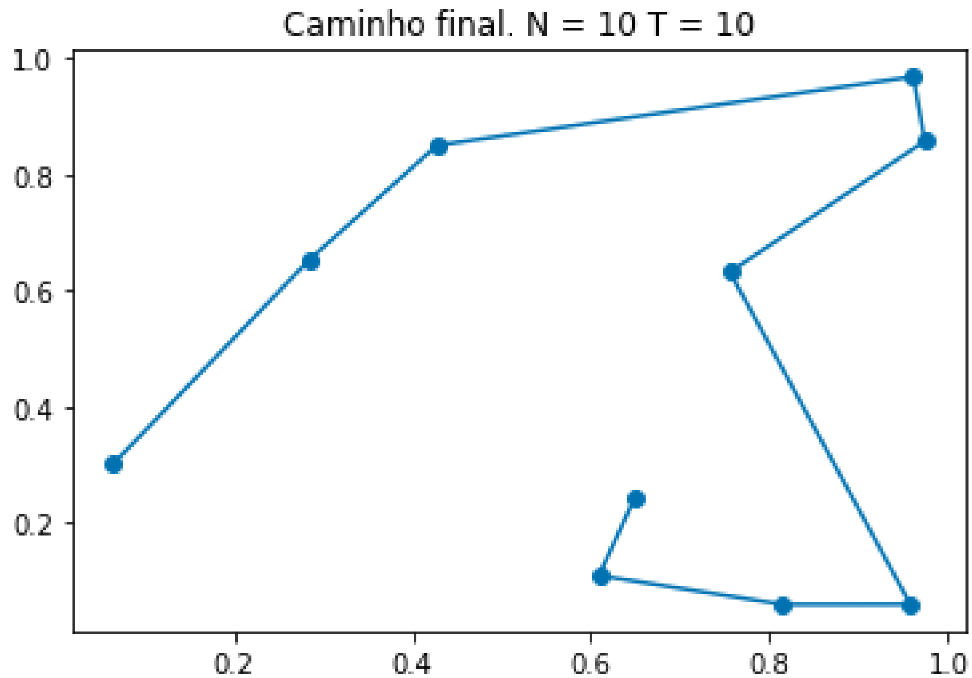


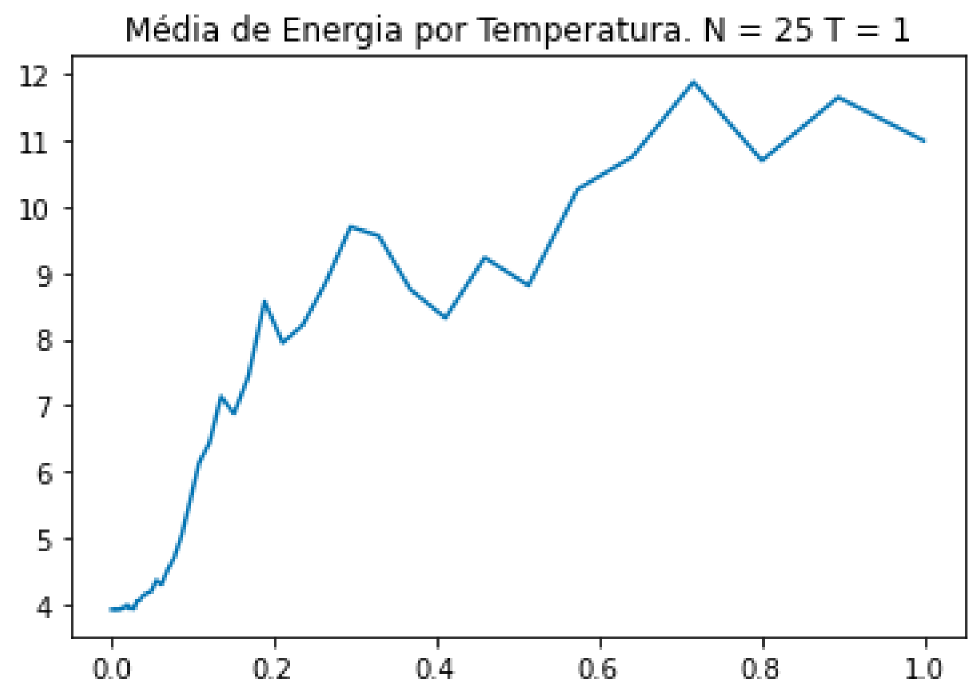

Caminho final. N = 10 T = 1
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0

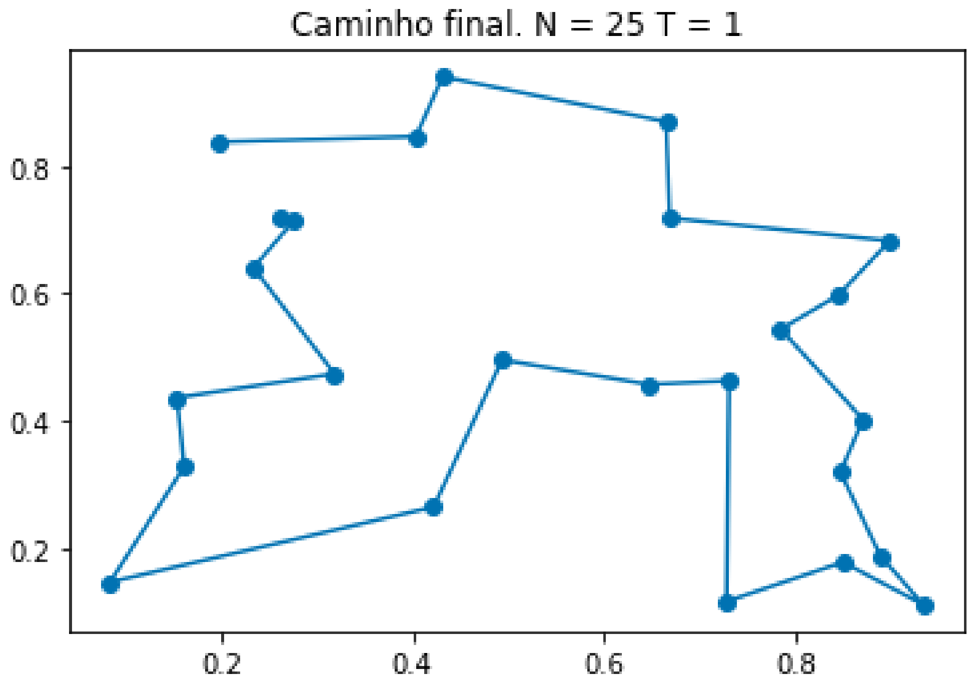

Média de Energia por Temperatura. N = 10 T = 5
6.0
5.5
5.0
4.5
4.0
3.5
0
1
2
3
4
5

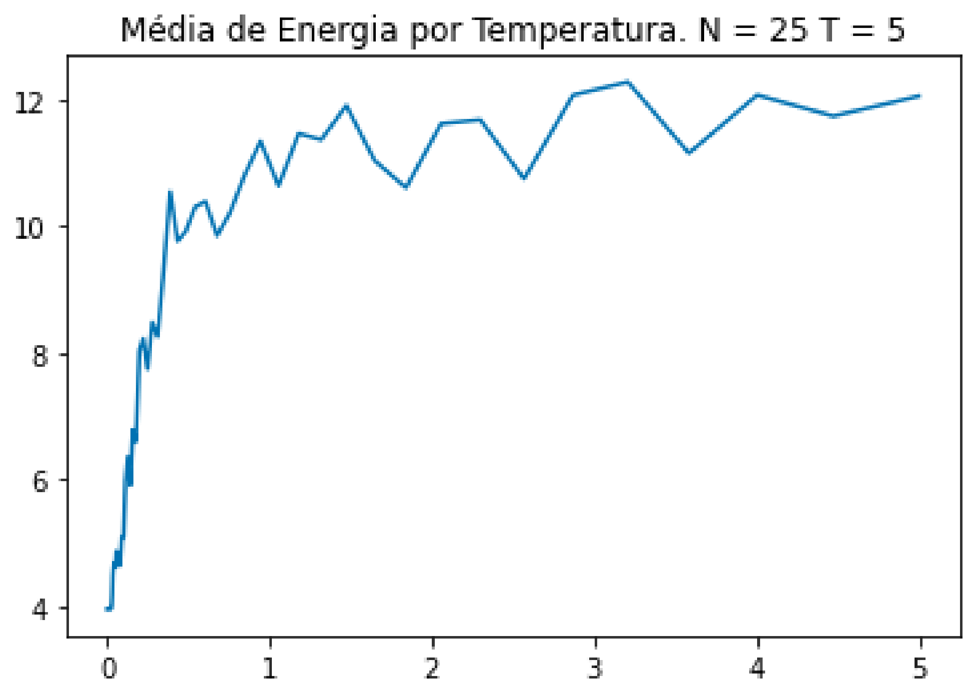

Caminho final. N = 10 T = 5
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0

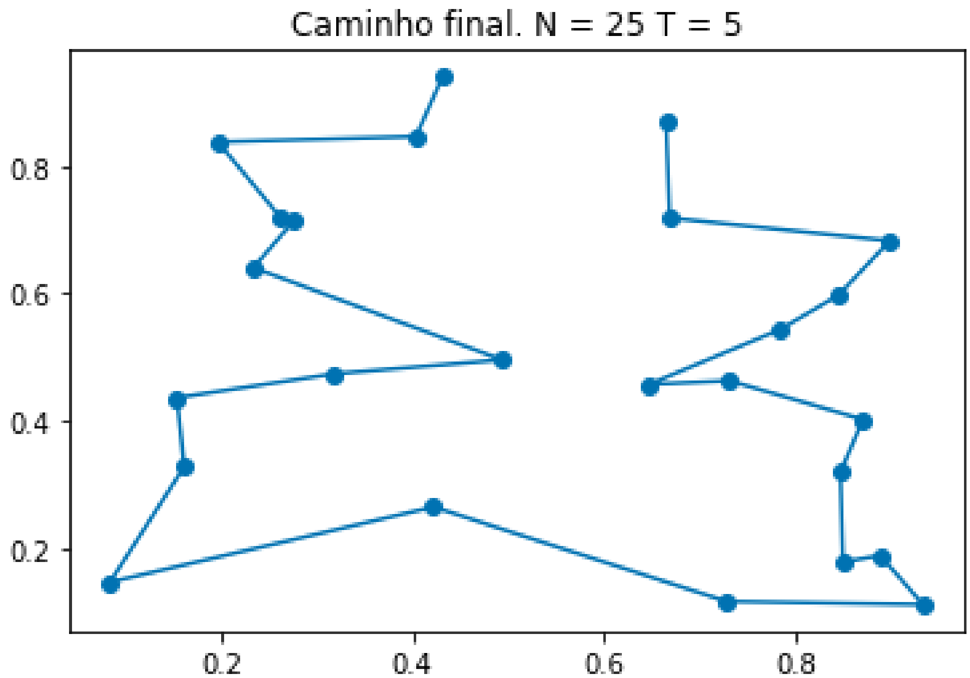

Média de Energia por Temperatura. N = 10 T = 10
6.5
6.0
5.5
5.0
4.5
4.0
3.5
0
2
4
6
8
10

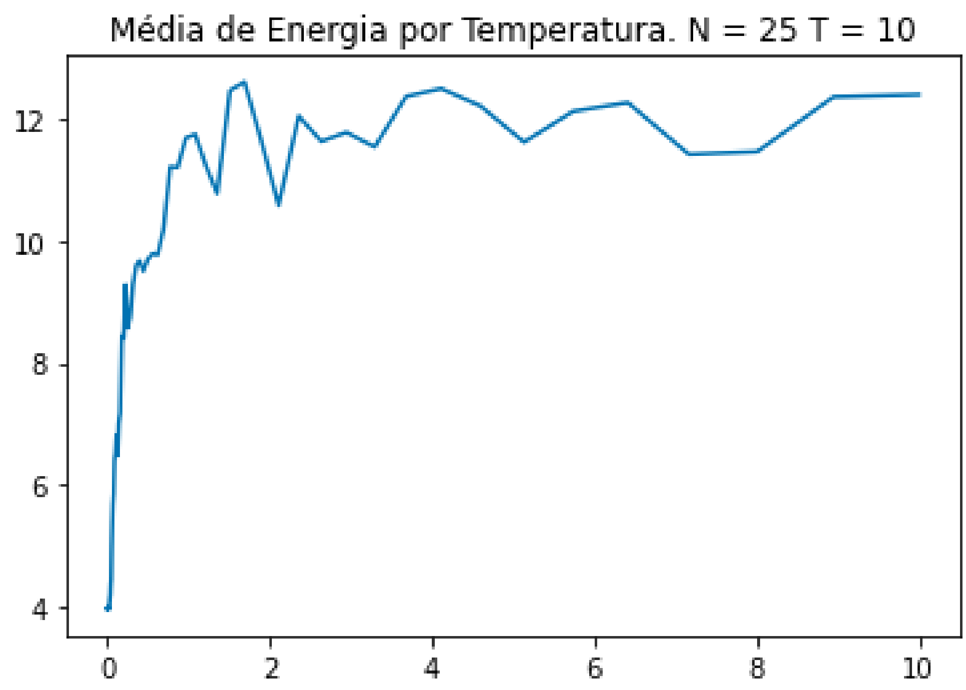

Caminho final. N = 10 T = 10
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0

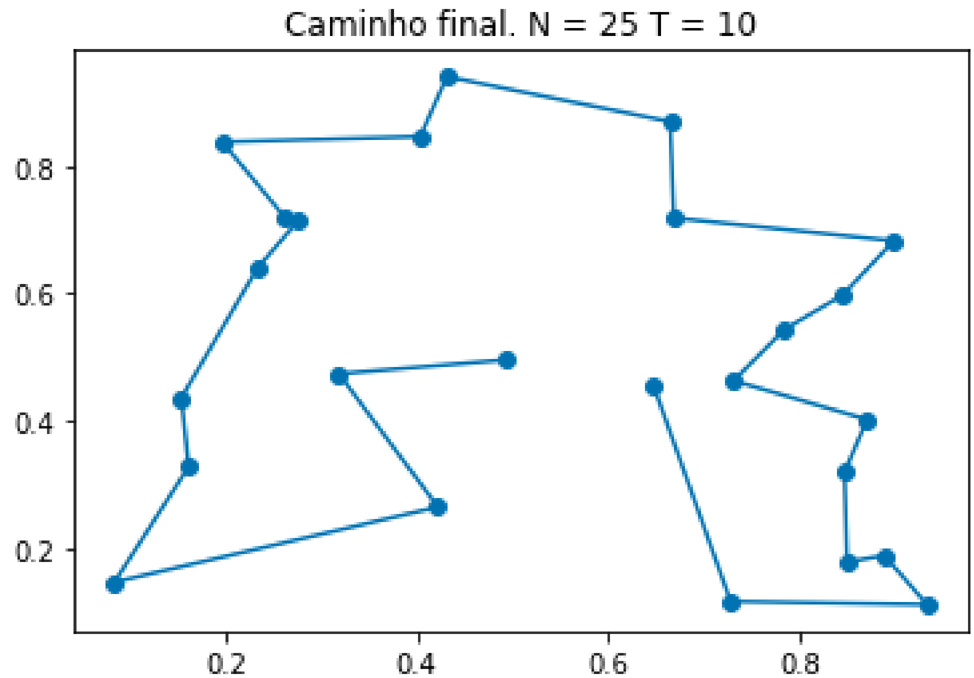

Média de Energia por Temperatura. N = 25 T = 1
12
11
10
9
8
7
6
5
4
0.0
0.2
0.4
0.6
0.8
1.0

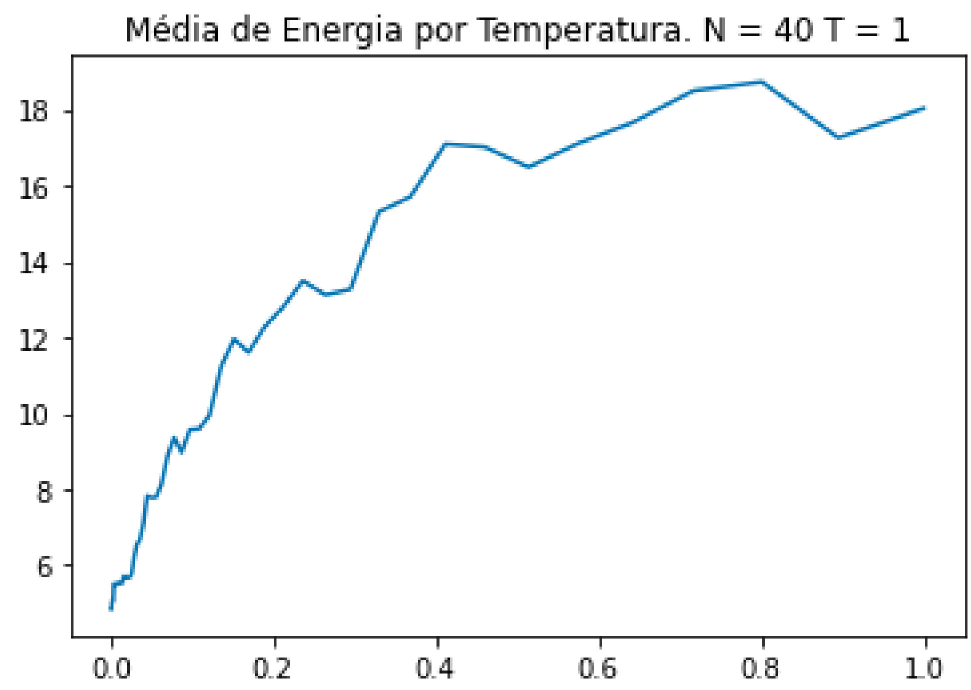

Caminho final. N = 25 T = 1
0.2
0.4
0.6
0.8
0.2
0.4
0.6
0.8

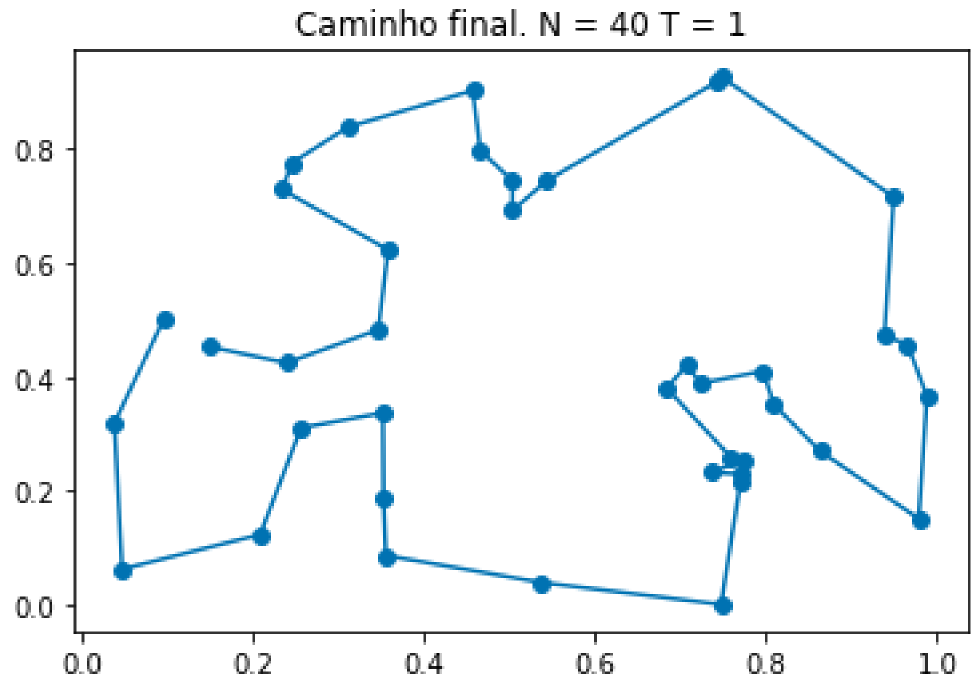

Média de Energia por Temperatura. N = 25 T = 5
4
6
8
10
12
0
1
2
3
4
5

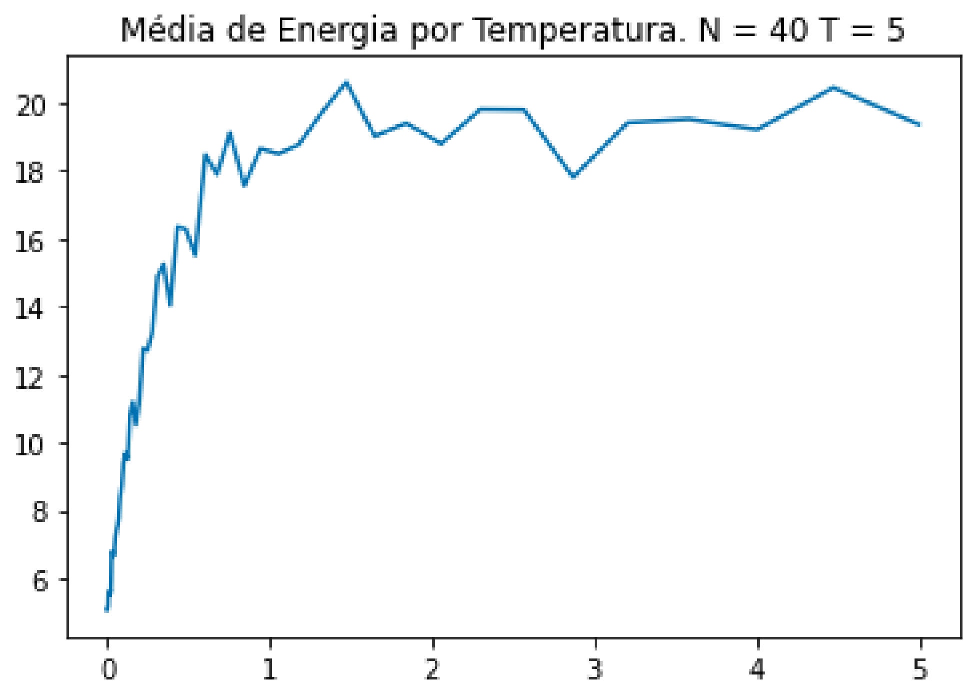

Caminho final. N = 25 T = 5
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8

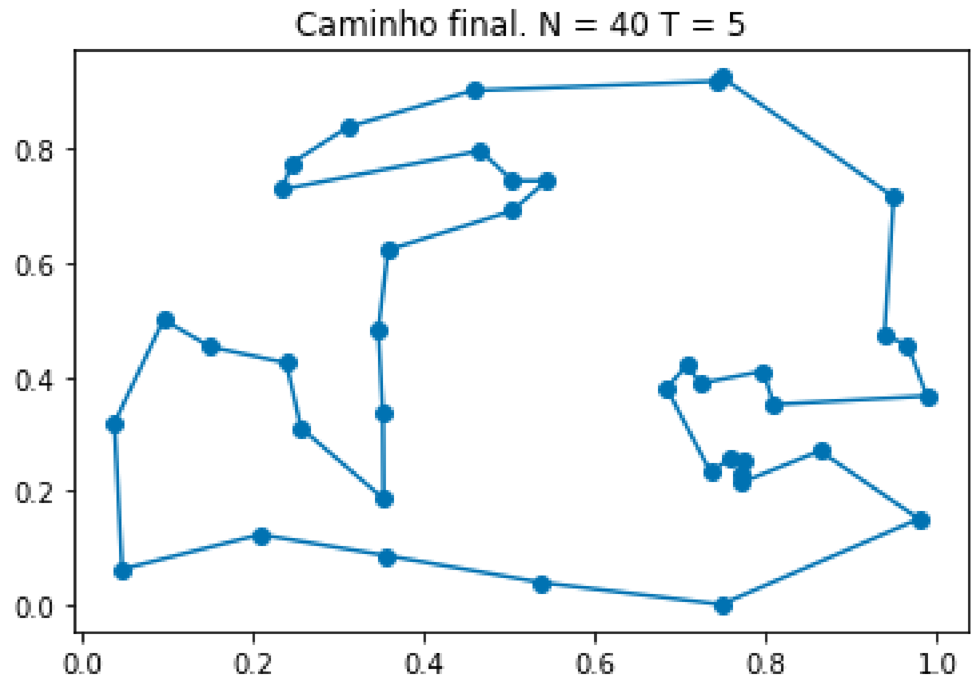

Média de Energia por Temperatura. N = 25 T = 10
12
10
8
6
4
0
2
4
6
8
10

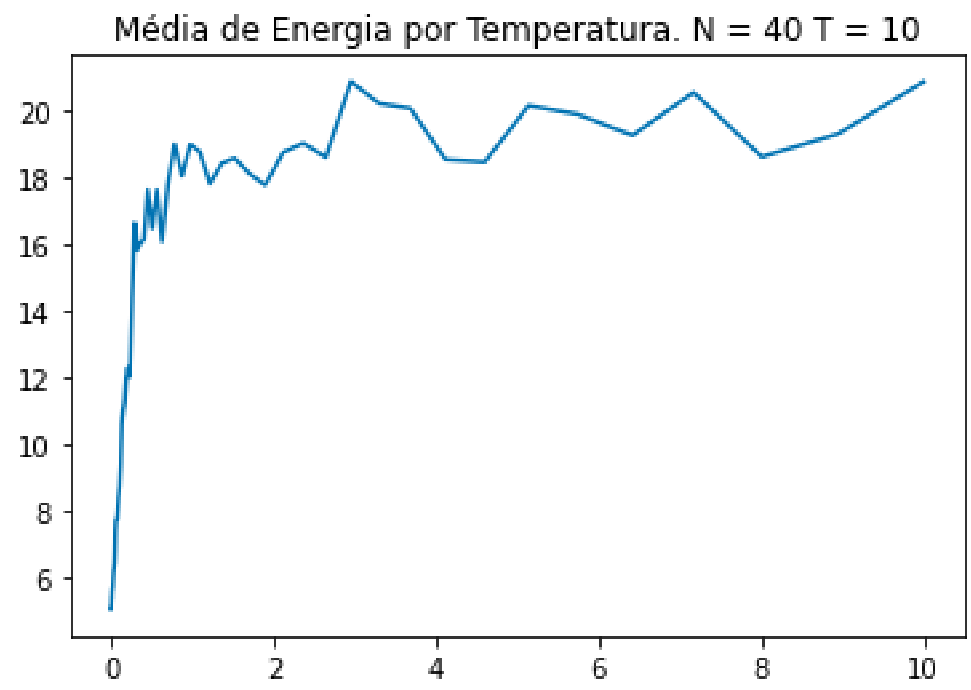

Caminho final. N = 25 T = 10
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8

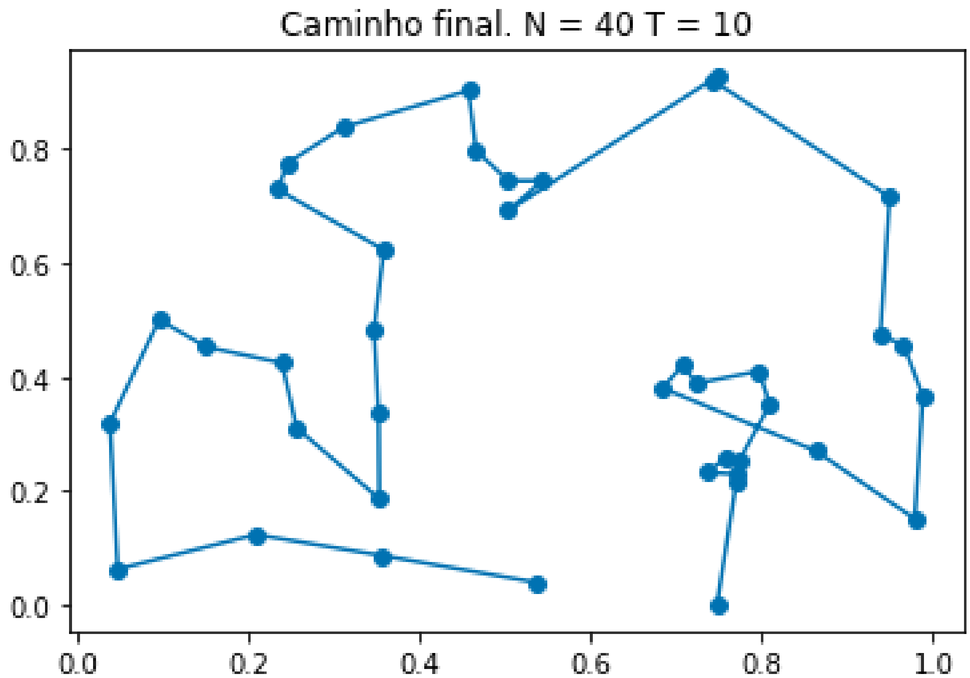

Média de Energia por Temperatura. N = 40 T = 1
18
16
14
12
10
8
6
0.0
0.2
0.4
0.6
0.8
1.0

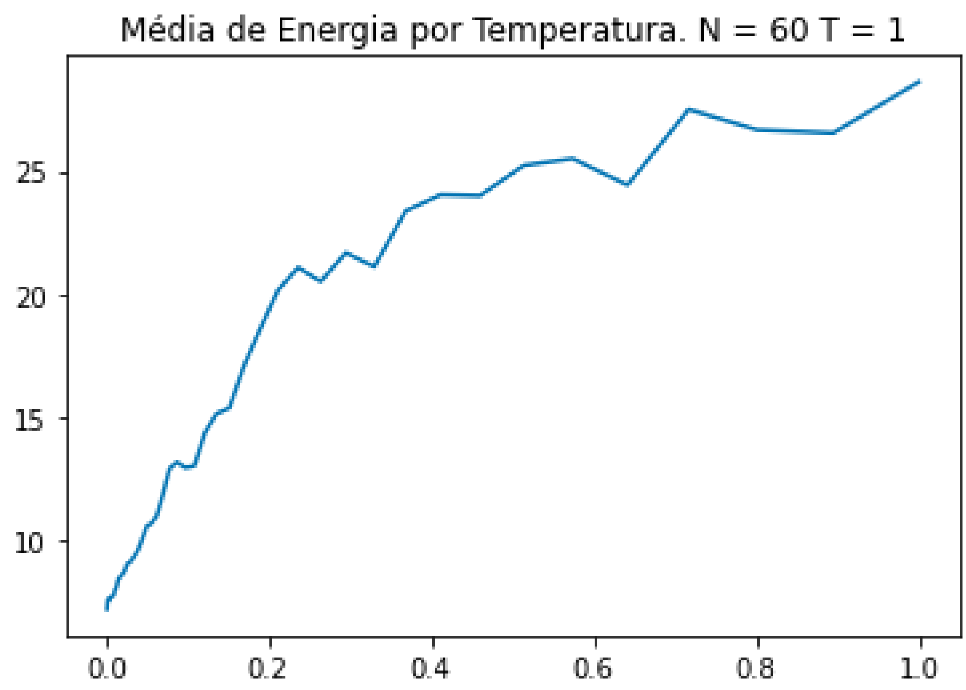

Caminho final. N = 40 T = 1
0.0
0.2
0.4
0.6
0.8
1.0

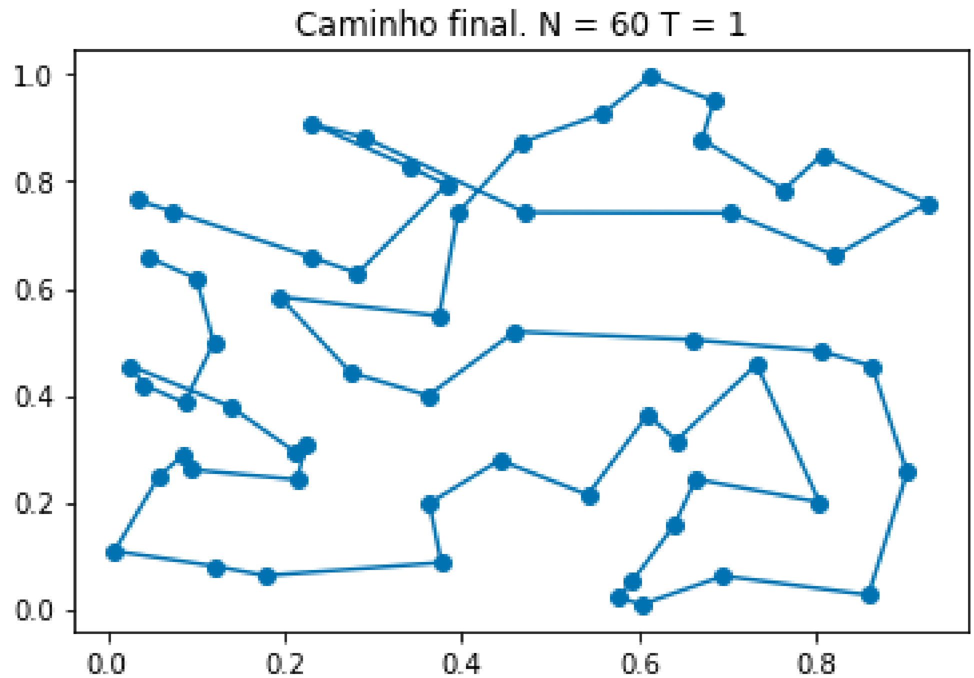

Média de Energia por Temperatura. N = 40 T = 5
6
8
10
12
14
16
18
20
0
1
2
3
4
5

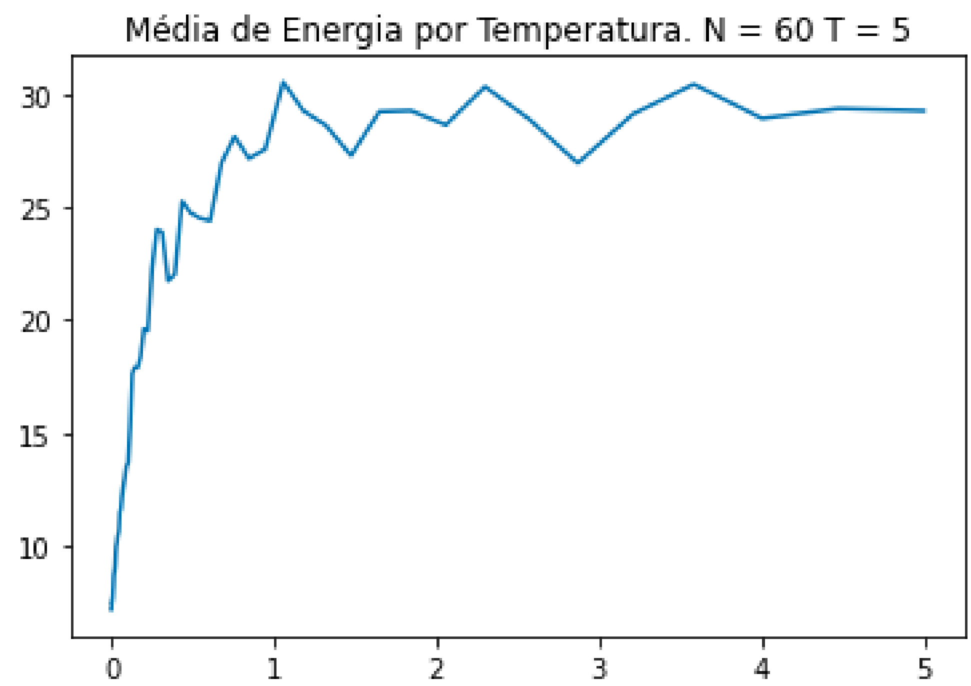

Caminho final. N = 40 T = 5
0.0
0.2
0.4
0.6
0.8
1.0

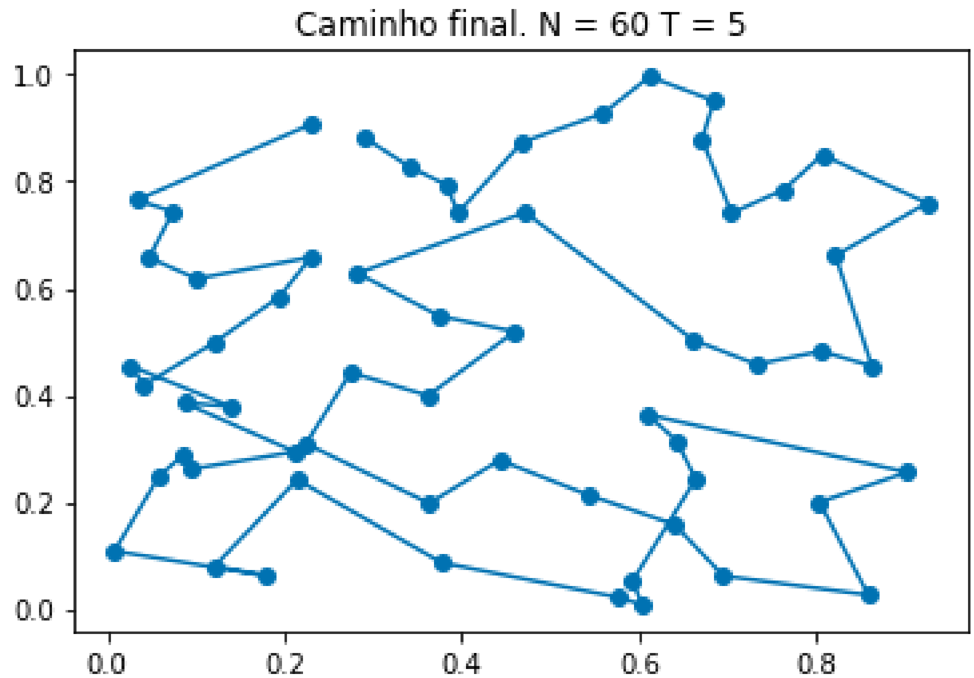

Média de Energia por Temperatura. N = 40 T = 10
6
8
10
12
14
16
18
20
0
2
4
6
8
10

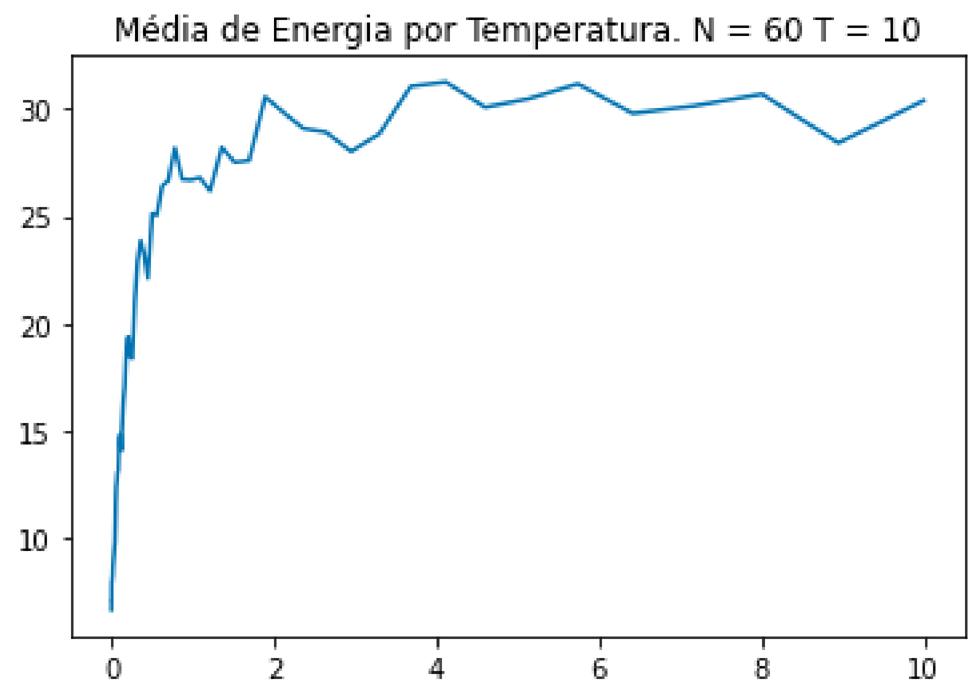

Caminho final. N = 40 T = 10
0.0
0.2
0.4
0.6
0.8
1.0

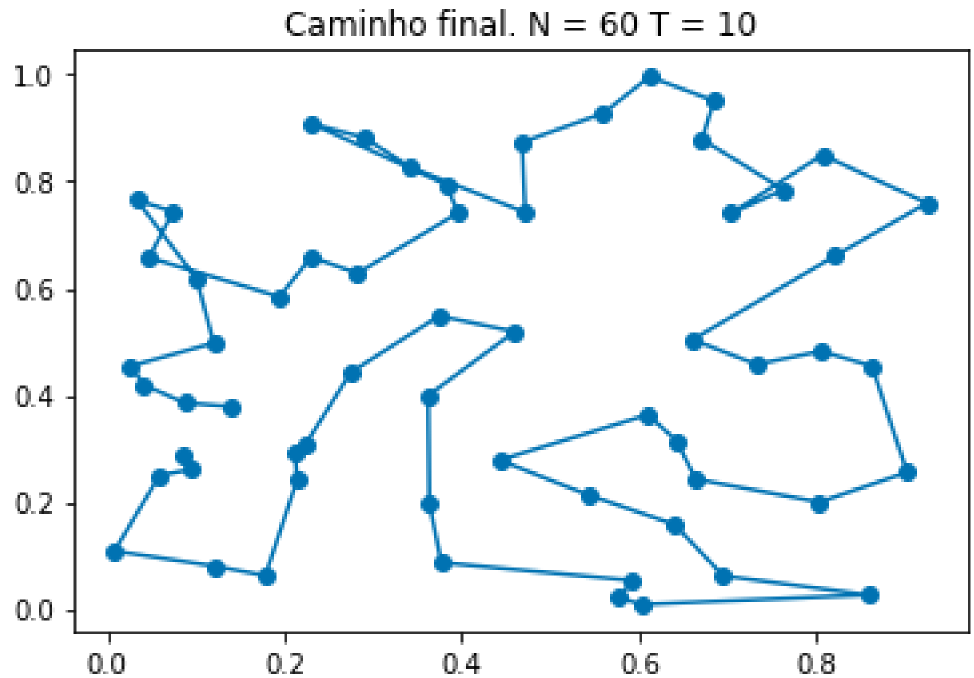

Média de Energia por Temperatura. N = 60 T = 1
10
15
20
25
0.0
0.2
0.4
0.6
0.8
1.0


Caminho final. N = 60 T = 1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8


Média de Energia por Temperatura. N = 60 T = 5
30
25
20
15
10
0
1
2
3
4
5


Caminho final. N = 60 T = 5
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8


Média de Energia por Temperatura. N = 60 T = 10
30
25
20
15
10
0
2
4
6
8
10



1.1.3 Análise dos Resultados Obtidos

Analisando a série de gráficos apresentados acima, nota-se alguns padrões, como o fato de que, quanto maior a temperatura, mais tempo o sistema leva para convergir. Além disso, quando a temperatura chega a valores abaixo de 1, rapidamente acontece essa convergência.

Outro ponto interessante a ser observado é que o sistema obteve menores energias para as temperaturas iniciais 5 e 10 quando comparadas à primeira temperatura.

Ao observar os caminhos finais gerados em diferentes execuções, vê-se que não são sempre iguais, ou seja, o modelo nem sempre atinge o resultado ótimo. Contudo, pos se tratar de uma execução bastante veloz, pode-se afirmar que é um modelo bastante adequado para análises do problema do Caixeiro Viajante.

```
[ ]:
```