Introdução à Física Computacional II (4300318)

Prof. André Vieira
apvieira@if.usp.br
Sala 3120 – Edifício Principal

Aula 12

Recozimento simulado

A distribuição de Boltzmann

- Considere que um sistema físico seja mantido a uma temperatura absoluta constante T. Como se discute em Física 2, no equilíbrio térmico, a probabilidade de observar o sistema em um **estado** i caracterizado pela energia E_i é dada pela **distribuição de Boltzmann**

$$P_i = \frac{e^{-\beta E_i}}{\sum_j e^{-\beta E_j}} \equiv \frac{e^{-\beta E_i}}{Z}, \quad \beta = \left(k_B T\right)^{-1}$$

Nas unidades do SI, a constante de Boltzmann k_B vale $1{,}38\times10^{-23}$ J/K. A quantidade representada por Z é chamada de "função de partição".

A distribuição de Boltzmann

$$P_i = \frac{e^{-\beta E_i}}{\sum_j e^{-\beta E_j}} \equiv \frac{e^{-\beta E_i}}{Z}, \quad \beta = \left(k_B T\right)^{-1}$$

- Podemos escolher a escala de energia para que o estado de mais baixa energia corresponda a $E_0 = 0$. Se esse estado fundamental for único, temos, **no limite de temperatura nula**,

$$P_i = \begin{cases} 1, & \text{se } i = 0 \\ 0, & \text{se } i \neq 0 \end{cases}.$$

Encontrando o estado fundamental

- Para encontrar o estado fundamental, em princípio podemos então resfriar o sistema até uma temperatura muito baixa e observar qual estado é alcançado.
- Encontrar o estado fundamental é um exemplo de um problema de **otimização**, uma categoria que engloba problemas em diversas áreas, como engenharia e computação, além de física.
- Vamos ver como a abordagem da distribuição de Boltzmann pode ser estendida a outros problemas.

O algoritmo de Metropolis

- O método da cadeia de Markov faz uso de taxas de transição escolhidas para satisfazer o balanceamento detalhado. A mais simples escolha é o **algoritmo de Metropolis**:

$$W_{ij} = \begin{cases} 1, & \text{se } E_j < E_i \\ e^{-\beta(E_j - E_i)}, & \text{se } E_j \geq E_i \end{cases}.$$

- A partir de um certo estado, definimos o novo estado a partir de um conjunto pré-determinado de movimentos possíveis (por exemplo, mudar a configuração de uma única partícula).

Simulações de Monte Carlo

- Eis a receita para implementar uma simulação de Monte Carlo com uma cadeia de Markov.

  1. Escolha um estado inicial qualquer.
  2. Escolha aleatoriamente um movimento a partir do conjunto de possibilidades.
  3. Calcule a taxa de transição W_{ij} associada.
  4. Aceite o movimento com probabilidade W_{ij}; em particular, movimentos que diminuem a energia são sempre aceitos.
  5. Meça e acumule as quantidades de interesse.
  6. Repita a partir do passo 2.

Recozimento simulado

- Como o método trabalha com um conjunto de movimentos possíveis, há o risco de que, quando a temperatura é muito baixa, fiquemos presos em um estado que corresponda a um mínimo local da energia, ou seja, um estado que tenha energia mais baixa do que qualquer dos estados alcançáveis a partir dali.
- Isso ocorre porque, no limite $T \rightarrow 0$, a probabilidade $\exp\left(-\Delta E / k_B T\right)$ de aceitação de um movimento que eleve a energia do sistema tende a zero.

Recozimento simulado

- Para tentar contornar essa dificuldade, partimos de altas temperaturas e, buscando emular o que se faz em metalurgia, resfriamos lentamente o sistema. A isso se chama **recozimento simulado**.

Recozimento simulado

- Na prática, implementamos um protocolo de resfriamento, reduzindo a temperatura segundo alguma regra. Mais comumente, essa regra corresponde a uma função exponencial

$$T = T_0 \exp\left(-t/\tau\right),$$

em que T_0 é uma temperatura alta, t é o "tempo" (medido em tentativas de movimentos) e τ é um número suficientemente grande para garantir um resfriamento eficiente, mas não tão grande a ponto de tornar o tempo de execução impraticável.

Recozimento simulado

- Mas o que fazer para adaptar o método para outros problemas de otimização? Afinal, temperatura é um conceito da física.

Recozimento simulado

- Mas o que fazer para adaptar o método a outros problemas de otimização? Afinal, temperatura é um conceito da física.
- Nada nos impede de introduzir uma quantidade que, de maneira puramente formal, desempenhe o papel da temperatura!
- Por exemplo, se quereremos maximizar uma função $f(x)$, convertemos o problema na maximização da função

$$p(x) = \exp\left[f(x)/T\right], \quad T \to 0.$$

Note a inversão de sinal no argumento da exponencial, já que aqui queremos maximizar a função, e não minimizar a energia.

Exemplo 1: o problema do caixeiro viajante

- Um problema clássico de otimização é o de minimizar a distância percorrida por um caixeiro viajante que deve visitar N cidades passando por cada uma delas exatamente uma vez, voltando ao ponto de partida ao final da viagem. Esse problema pertence a uma classe (chamada *NP-difícil*) de problemas cuja solução levaria à solução de vários outros problemas de otimização. No entanto, não se sabe se a solução ótima requer um número de passos polinomial ou exponencial em N.

Um problema semelhante consiste em determinar a rota mais curta ou mais rápida a ser seguida por um carro em uma cidade.

Exemplo 1: o problema do caixeiro viajante

- Concretamente, dadas as N cidades a visitar, queremos minimizar a distância

$$D = \sum_{i=0}^{N-1} \left| \vec{r}_{i+1} - \vec{r}_i \right|, \quad \vec{r}_N \equiv \vec{r}_0 .$$

- Vamos então dispor as cidades em um quadrado de lado 1 (em unidades arbitrárias), e introduzir uma "temperatura" T para maximizar

$$\exp\left(-D/T\right), \quad T \to 0,$$

reduzindo T segundo

$$T = T_0 \exp\left(-t/\tau\right).$$

Exemplo 1: o problema do caixeiro viajante

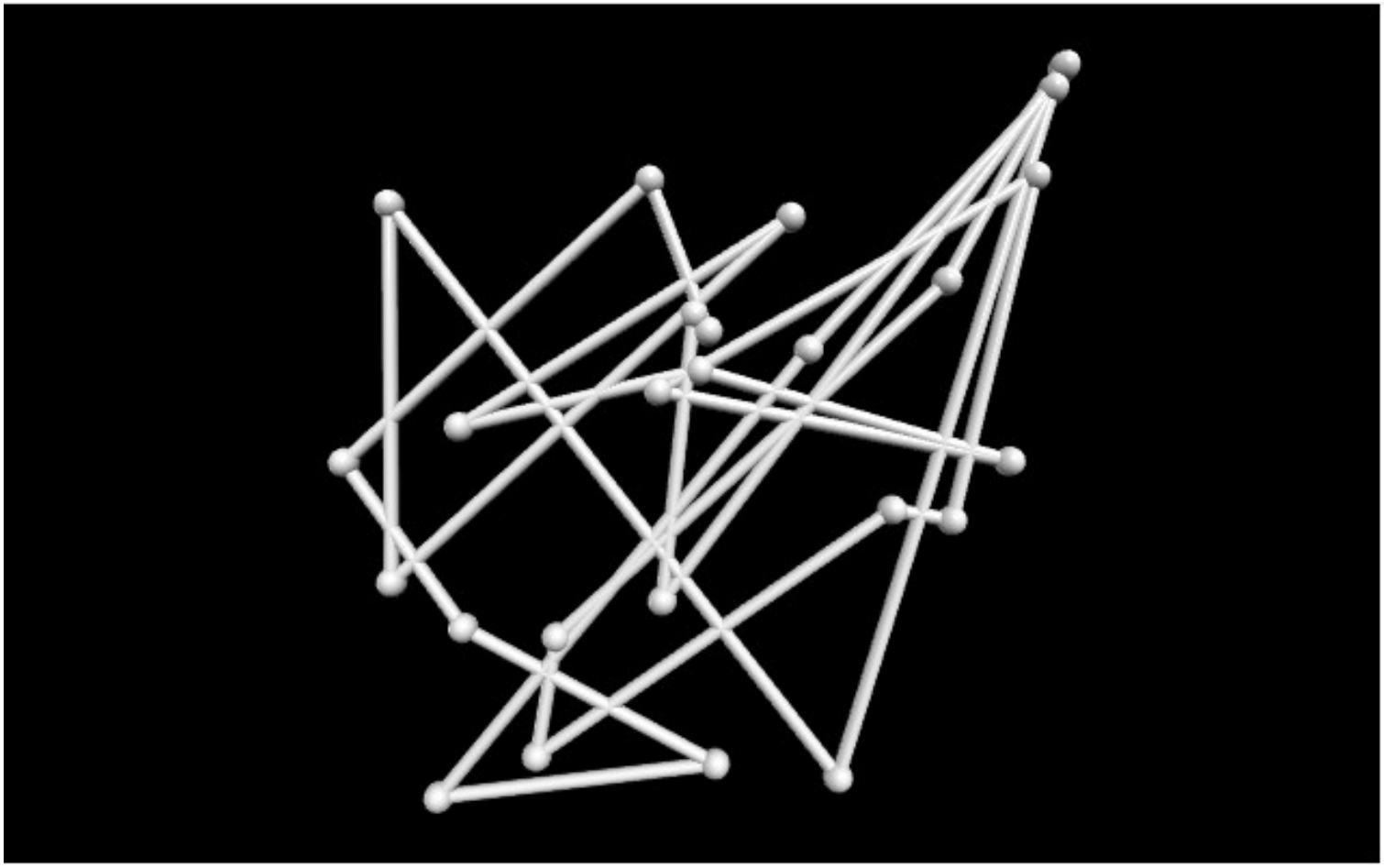
```
# Parâmetros do problema
N = 25          # Número de cidades a percorrer
T0 = float(N)   # Temperatura inicial do recozimento
Tmin = N/25000  # Temperatura final do recozimento
tau = 1e3       # Tempo característico da redução da temperatura

# Parâmetros da visualização
R = 0.02        # Raio das esferas que representam as cidades

# Função para calcular a distância percorrida durante a viagem
def distancia(r):
    s = 0.0
    for i in range(N):
        s += mag(r[i+1]-r[i])
    return s

# Escolhendo N posições para as cidades e calculando a distância inicial
r = [0]*(N+1) # Lista que armazena as posições das cidades
for i in range(N):
    r[i] = vector(random(), random(), 0.0)
r[N] = r[0]
D = distancia(r)

print("A estimativa original é",D)

# Inicializando os gráficos
canvas(center=vector(0.5,0.5,0))
for i in range(N):
    sphere(pos=r[i],radius=R)
l = curve(pos=r,radius=R/2)          # Criando a curva que liga as cidades
```

Exemplo 1: o problema do caixeiro viajante

```
# Laço principal
t = 0    # Zerando o tempo inicial
T = T0   # Inicializando a temperatura
while T>Tmin:
    # Resfriamento
    t += 1
    T = T0*exp(-t/tau)
    # Atualizando a visualização a cada 100 tentativas de movimento
    if t%100==0:
        l.clear()                       # Limpando a curva
        l = curve(pos=r,radius=R/2)     # Recriando a curva
    # Escolhendo duas cidades distintas e trocando a ordem de visitação
    i,j = randrange(1,N),randrange(1,N)
    while i==j:
        i,j = randrange(1,N),randrange(1,N)
    D_antigo = D
    r[i],r[j] = r[j],r[i]
    D = distancia(r)          # Nova distância total da viagem
    deltaD = D - D_antigo     # Variação na distância
    # Se o movimento é rejeitado, trocamos as cidades de volta
    if random()>exp(-deltaD/T):
        r[i],r[j] = r[j],r[i]
        D = D_antigo

print("A estimativa para a distância otimizada é",D)
```

Exemplo 1: o problema do caixeiro viajante

$T_0 = N$

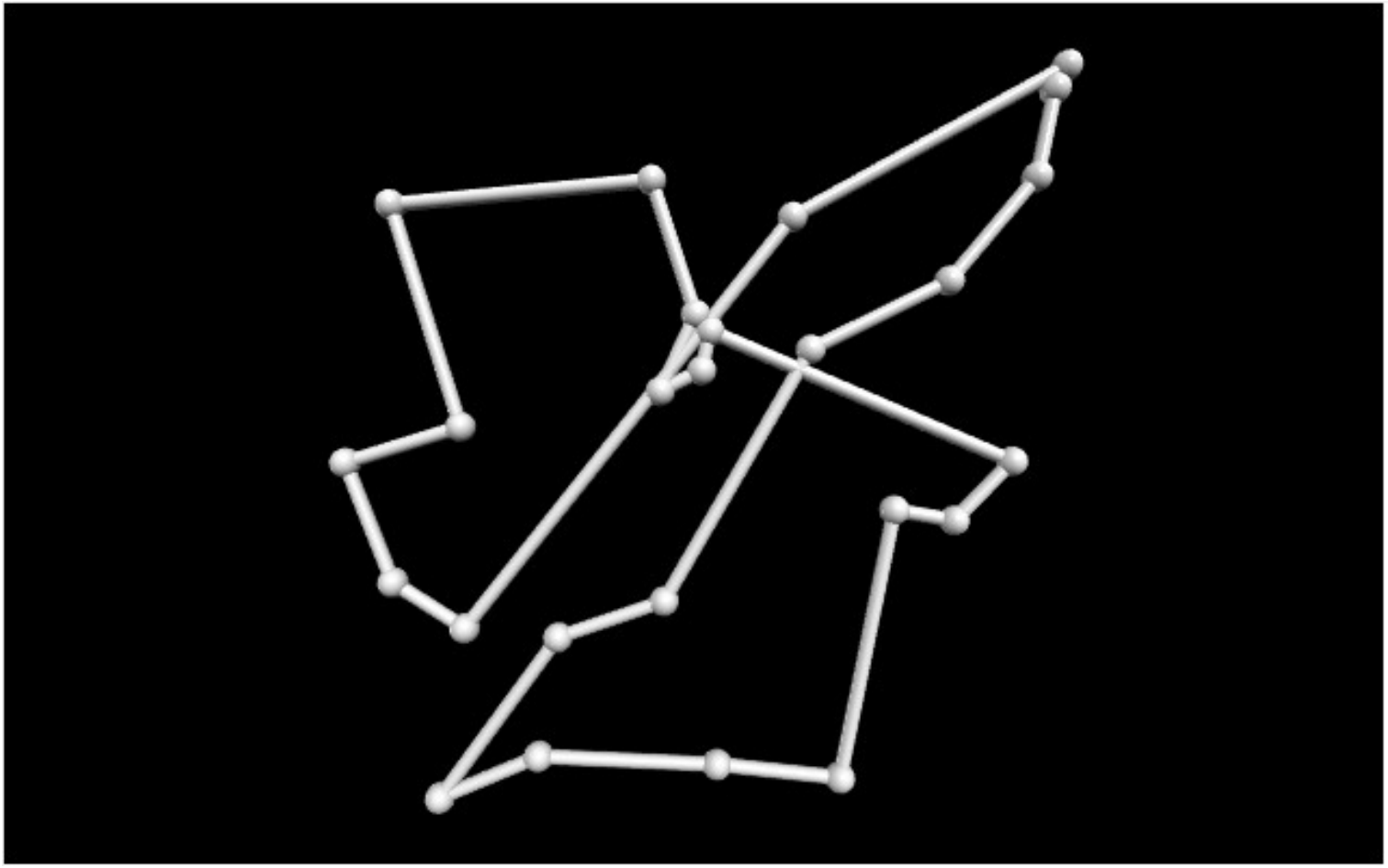


A distância original era de 12,2.

Exemplo 1: o problema do caixeiro viajante

$T_0 = N$

$\tau = 10^3$

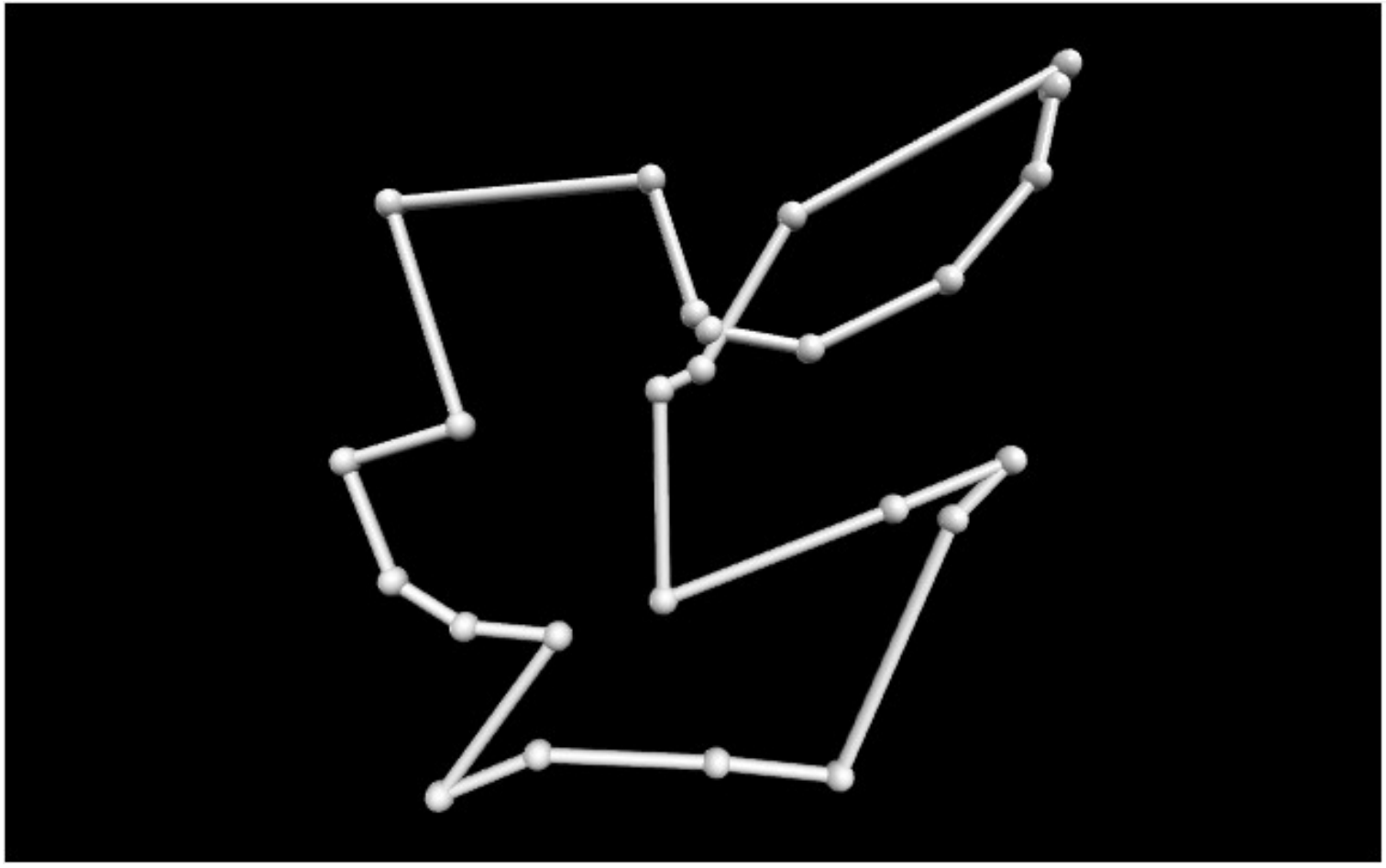


A estimativa para a distância total otimizada aqui foi de 5,56.

Exemplo 1: o problema do caixeiro viajante

$T_0 = N$

$\tau = 10^4$

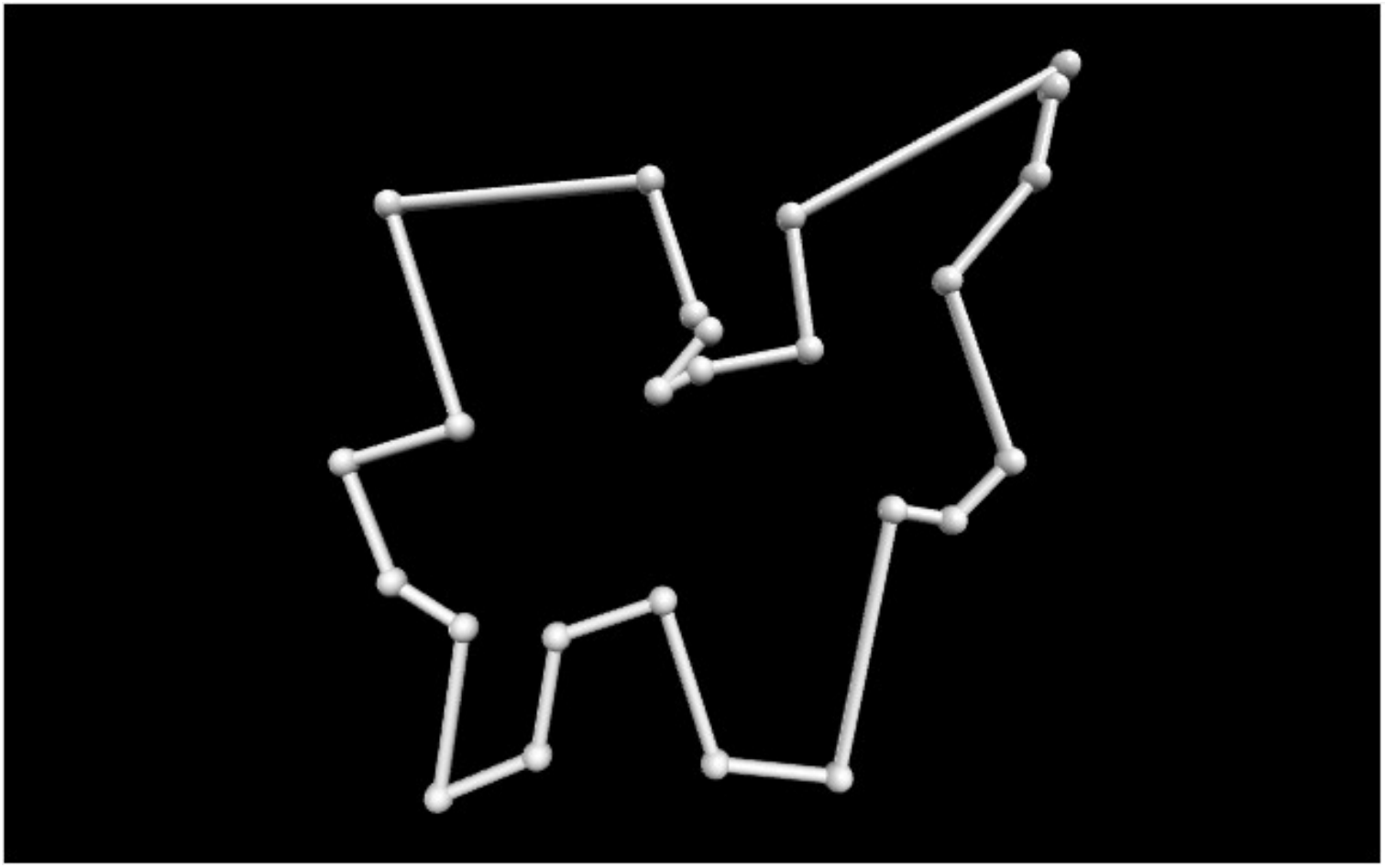


A estimativa para a distância total otimizada aqui foi de 4,94.

Exemplo 1: o problema do caixeiro viajante

$T_0 = N$

$\tau = 10^5$

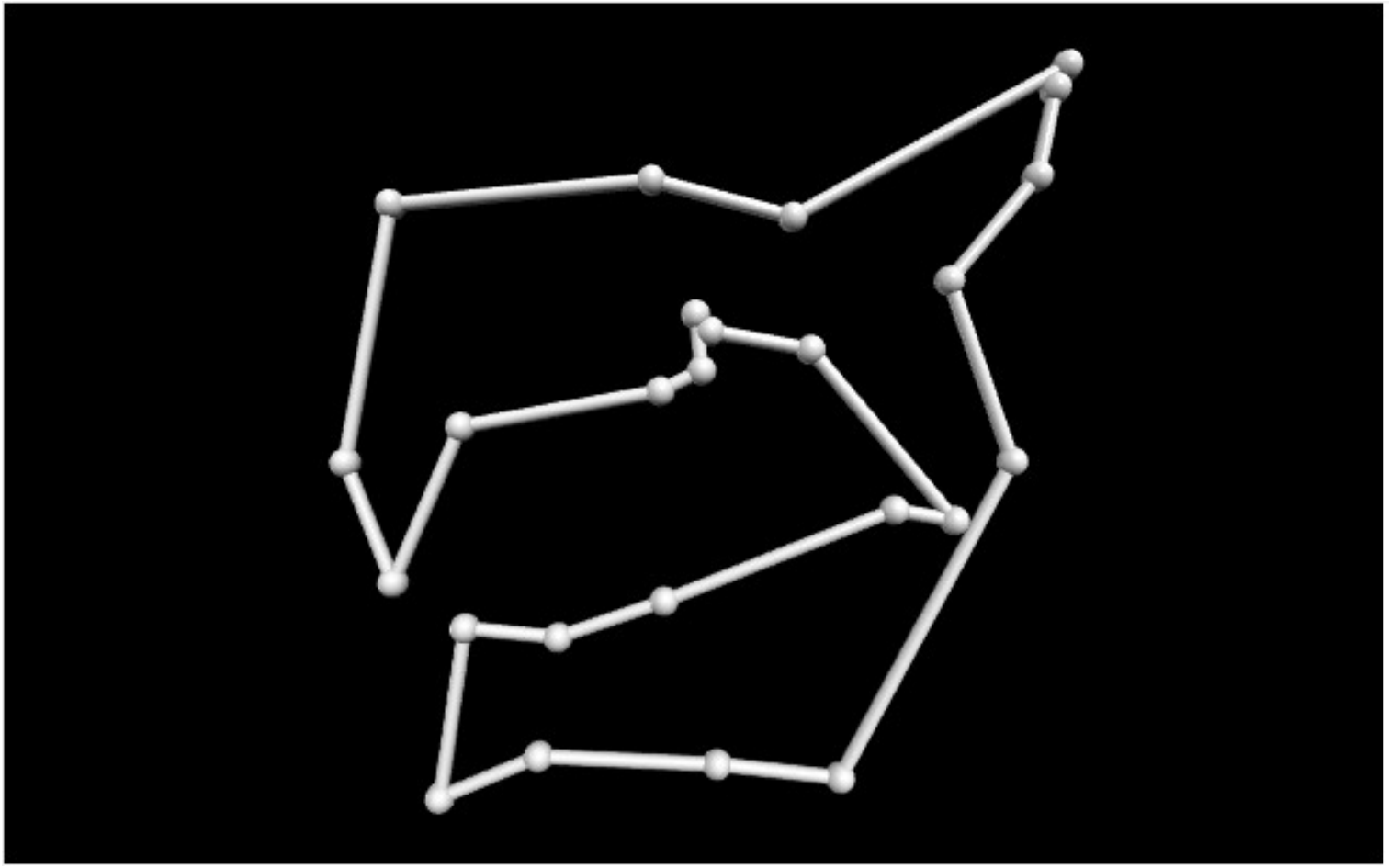


A estimativa para a distância total otimizada aqui foi de 4,69.

Exemplo 1: o problema do caixeiro viajante

$T_0 = 1/N$

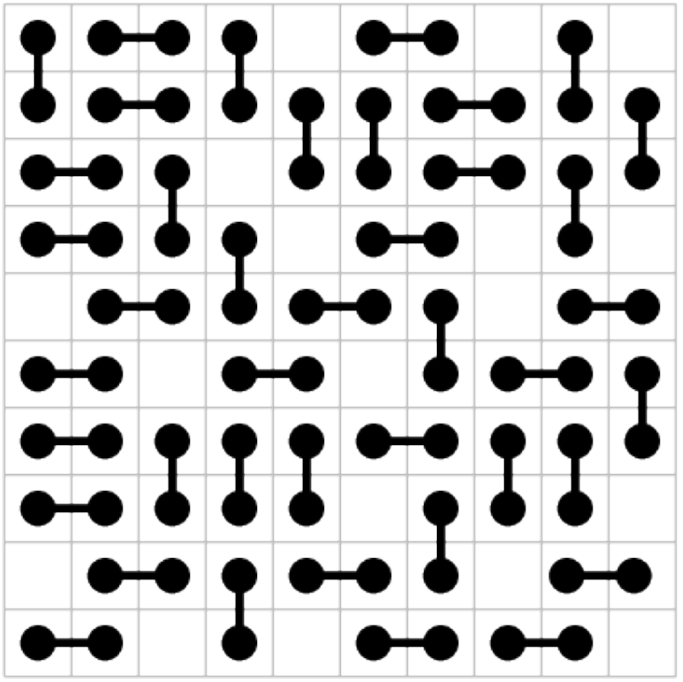
$\tau = 10^5$



A estimativa para a distância total otimizada aqui foi de 5,14.

Exemplo 2: o problema da cobertura por dímeros

- Dímeros (polímeros com 2 átomos) devem ser depositados em uma rede de modo a acomodar o maior número possível.



Exemplo 2: o problema da cobertura por dímeros

- Dímeros (polímeros com 2 átomos) devem ser depositados em uma rede de modo a acomodar o maior número possível.
- A tarefa portanto é maximizar o número de dímeros. Para isso, recorremos ao recozimento simulado com as etapas a seguir.

1) Escolhemos um par de sítios vizinhos.
   - Se estiverem ambos vazios, colocamos um dímero.
   - Se ambos os sítios estiverem ocupados pelo mesmo dímero, o removemos com probabilidade $\exp(\Delta N/T)$, $\Delta N = -1$
   - Em qualquer outra situação, nada ocorre.
2) Retornamos ao passo 1, diminuindo a temperatura.

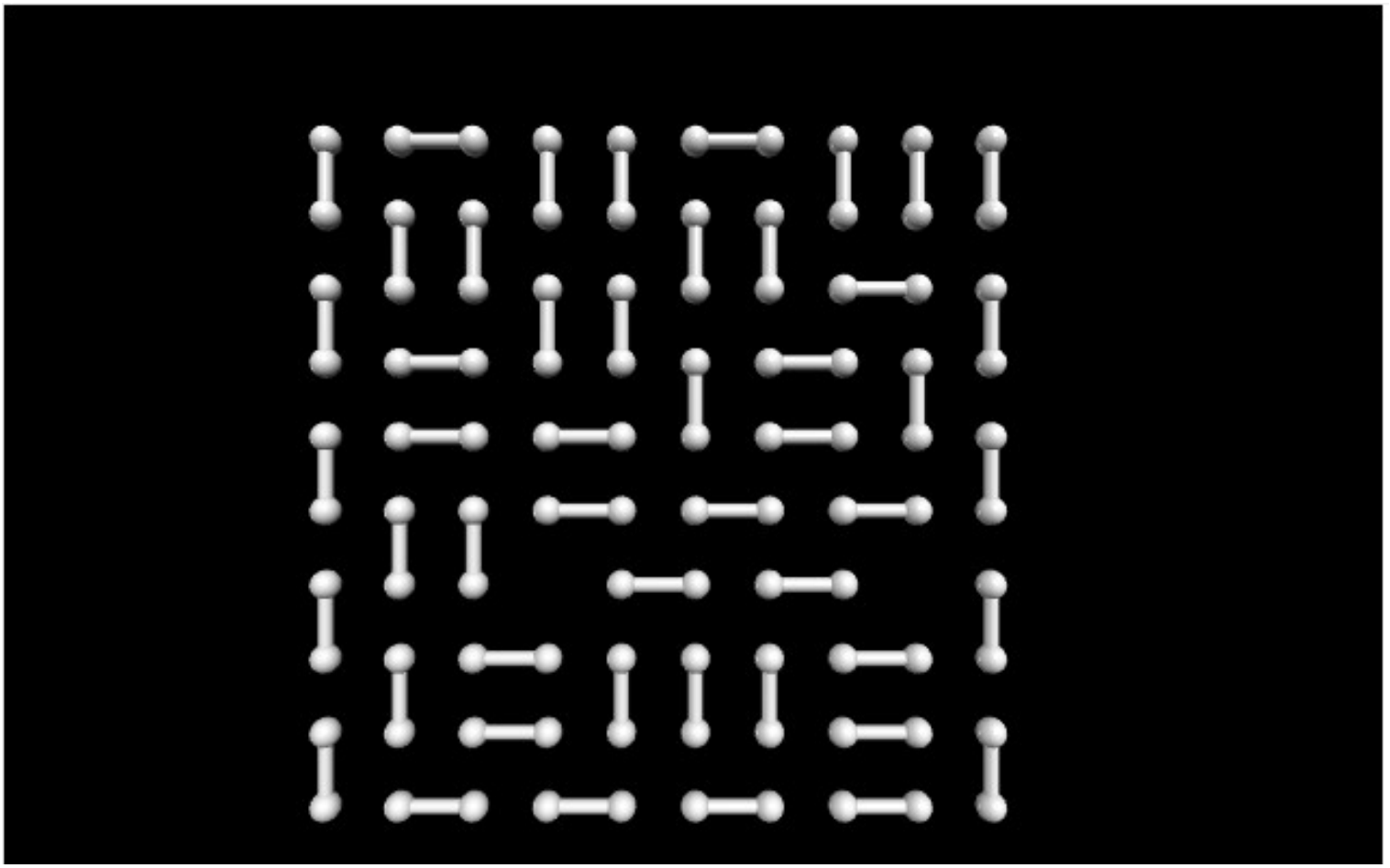
Exemplo 2: o problema da cobertura por dímeros

```
# Implementa a solução aproximada do problema da cobertura por dímeros
# utilizando recozimento simulado.

# Parâmetros do problema
L = 50          # Lado da rede quadrada
T0 = float(L)   # Temperatura inicial do recozimento
Tmin = 1e-5     # Temperatura final do recozimento
tau = 1e5       # Tempo característico da redução da temperatura

# Definindo uma matriz para armazenar a que dímero pertencem os
# sítios da rede. O estado -1 indica que o sítio não pertence a nenhum dímero.
d = full([L,L],-1)
# Definindo uma lista que armazena os sítios pertencentes a cada dímero
sitios = []

N = 0           # Inicialmente não há dímeros

# Laço principal
t = 0    # Zerando o tempo inicial
T = T0   # Inicializando a temperatura
# Vamos criar listas para indicar os deslocamentos até os 4 vizinhos de qualquer sítio
dx, dy = [1,0,-1,0], [0,1,0,-1]
```

Exemplo 2: o problema da cobertura por dímeros

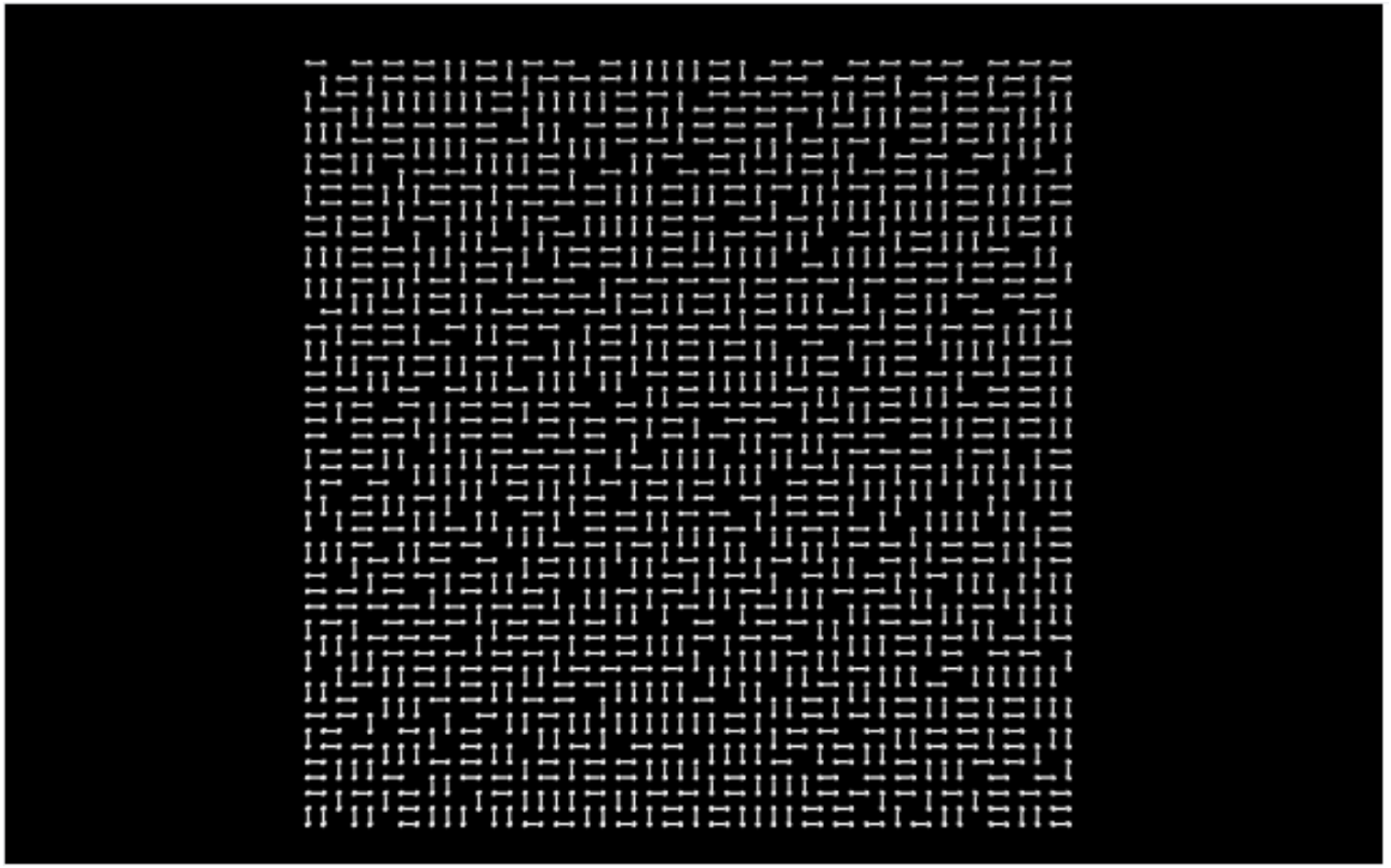
```
while T>Tmin:
    # Escolhendo dois sítios vizinhos ao acaso
    x, y = randrange(L), randrange(L)
    viz = randrange(4)
    xv, yv = x+dx[viz], y+dy[viz]
    if xv == -1 or xv == L or yv == -1 or yv == L:
        continue
    # Resfriamento
    t += 1
    T = T0*exp(-t/tau)
    if d[x,y] != d[xv,yv]:        # Sítios não pertencem ao mesmo dímero?
        continue
    if d[x,y] == -1 and d[xv,yv] == -1:  # Sítios estão ambos vazios?
        sitios.append([[x,y],[xv,yv]]) # Acrescentamos o par à lista de dímeros
        d[x,y] = d[xv,yv] = N
        N += 1
        continue
    if d[x,y] != -1 and d[x,y] == d[xv,yv]: # Sítios formam um dímero?
        if random()<exp(-1/T):
            # Se o movimento for aceito, e o dímero não ocupava a última
            # posição na lista, movemos na lista o último dímero para
            # a posição do dímero que será removido e trocamos o rótulo
            indice = d[x,y]
            d[x,y] = d[xv,yv] = -1
            if indice < N-1:
                sitios[indice] = sitios[N-1]
                x, y = sitios[indice][0][0], sitios[indice][0][1]
                xv, yv = sitios[indice][1][0], sitios[indice][1][1]
                d[x,y] = d[xv,yv] = indice
            sitios.pop()
            N -= 1
            continue
```

Exemplo 2: o problema da cobertura por dímeros



Um exemplo com $L = 10$. Notem a presença de dois sítios isolados não ocupados, indicando que a solução não é ótima.

Exemplo 2: o problema da cobertura por dímeros



Com $L = 50$, a estimativa do valor do número de dímeros fica em torno de 1150, contra 1250 do valor correto.

Exercício no moodle

Há um único exercício, explorando o conteúdo da aula de hoje, e que pode ser feito com base nos programas dos exemplos desta aula e da aula anterior, disponíveis no moodle. O exercício deve ser entregue até o dia **10 de junho**.