Universidade de São Paulo

Instituto de Física de São Carlos

Análise e Reconhecimento de Padrões

Docente: Prof. Luciano Fontoura da Costa

Projeto 2: Atributos

Beatriz de Camargo Castex Ferreira
10728077
bcastex@usp.br

São Paulo - 22/05/2020

Índice

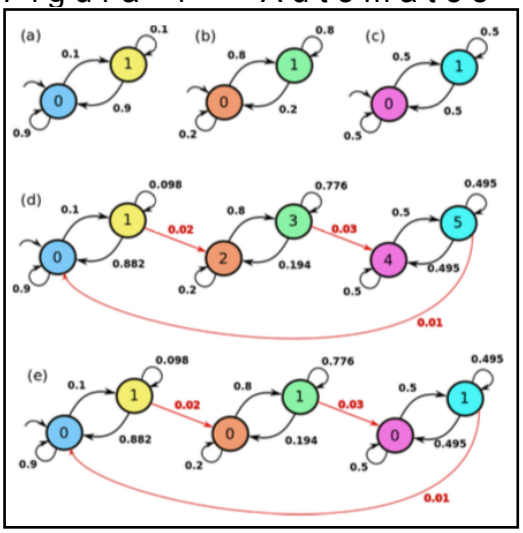
1. Resumo

Atributos são basicamente qualquer medida retirada de um certo objeto ou conjunto que permita com que reconheçamos padrões. Novamente utilizando o exemplo de uma banana, os atributos que usamos para reconhece-la são coisas como a cor amarela, a forma curvada, etc. É importante tomar cuidado ao escolher os atributos que serão utilizados para reconhecer qualquer padrão pré determinado, no geral deve-se tentar utilizar apenas atributos que permita a diferenciação entre os padrões. Nesse projeto serão analisados padrões gerados pelos autônomos da figura 1 e da figura 2 utilizando 4 conjuntos de atributos diferentes: bursts de split signals, distâncias intersinais de split-signals, espectro da potência da transformada de Fourier dos split signals e gratos dos sinais.

2. Split Signals

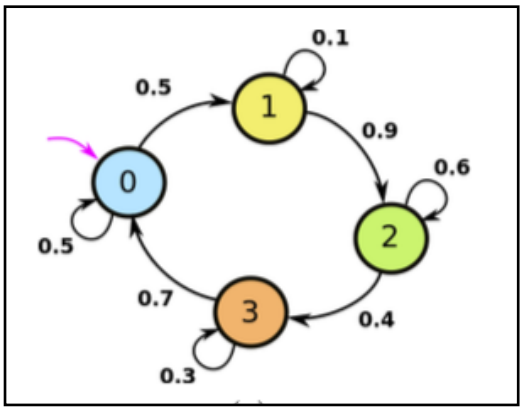
Dado um sinal gerado por um certo autômato pode-se separar esse em múltiplos padrões chamados split signals, um para cada estado. Isso é possível marcando um 1 toda vez que o agente estiver em no estado especificado e um 0 para todos os outros estados. A seguir está o programam feito para obter os split siglas dos autômatos das figuras 1 do CDT-22[1] e 2 do CDT-23[2] abaixo:

Figura 1 - Autômatos



Fonte: CDT-22[1]

Figura 2 - Autômatos



Fonte: CDT-23[2]

Por motivos de visualização, os programas escritos no projeto serão apresentados de forma simplificada, ou em imagens (ou seja, não seria possível copiar o código e implementar em seu próprio computador), portanto todos estarão disponíveis em sua integridade, juntamente com quaisquer dados utilizados para gerar os resultados em uma pasta do google drive linkada abaixo[3].

O programa que realiza split está disponível juntamente com os arquivos utilizados para obter os gráficos abaixo na pasta 2-split com o nome split_signal.py :

---

[1] Referência [1]. Disponível em: <https://www.researchgate.net/publication/339599069>

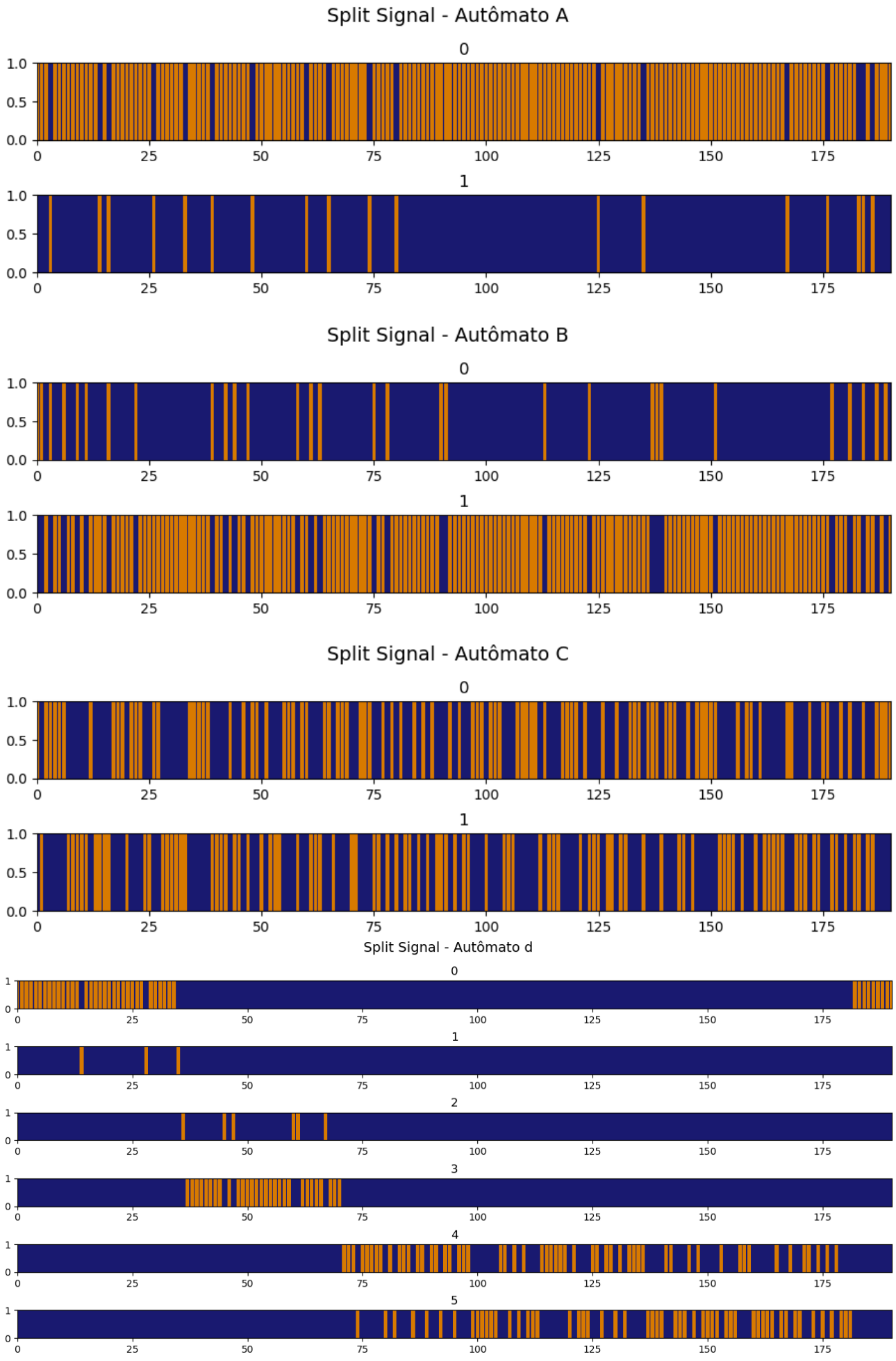
[2] Referência [2]. Disponível em:<https://www.researchgate.net/publication/339800429>

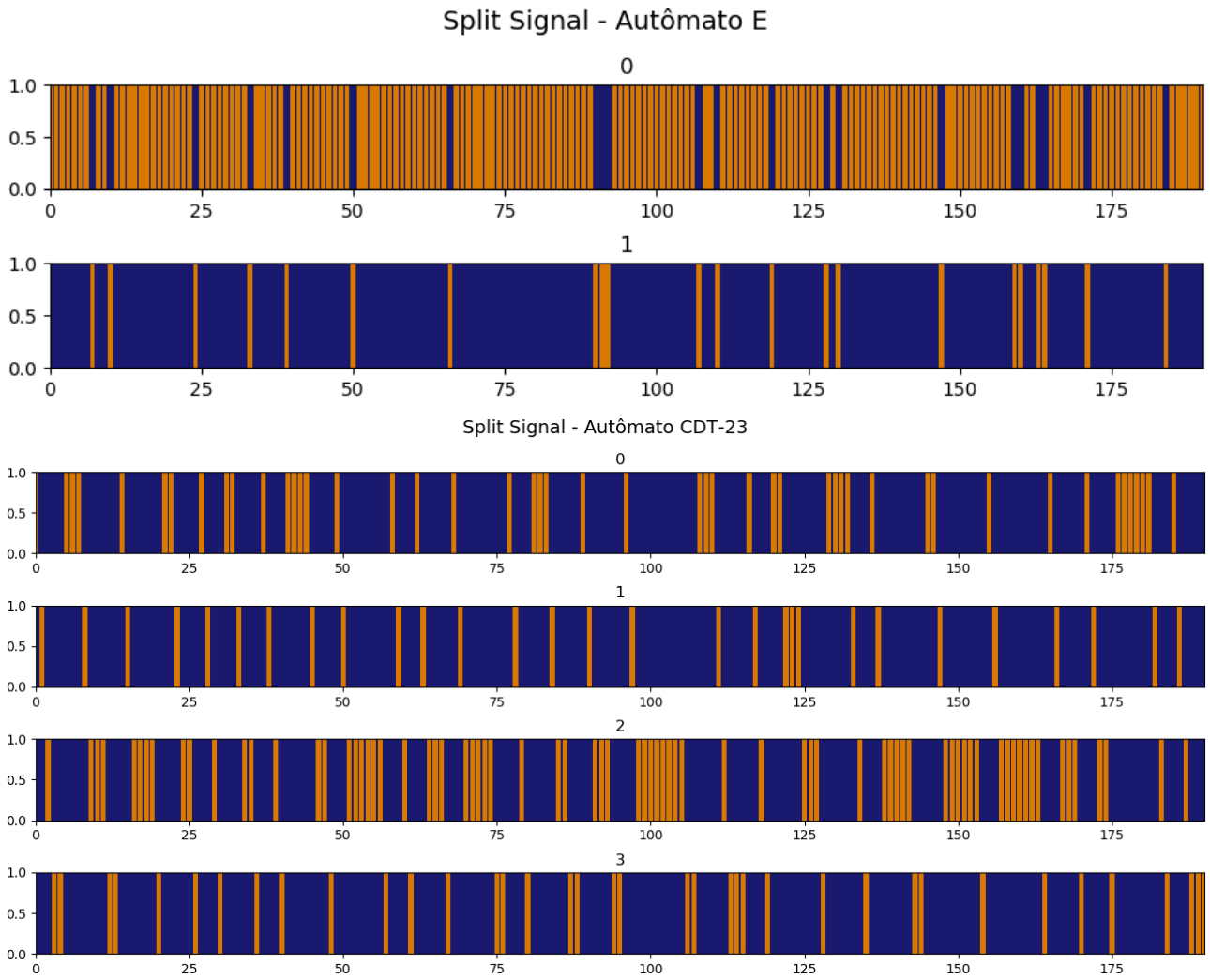
[3] Disponível em: <https://drive.google.com/drive/folders/1JRwXtCoVFILRGxWCBixfDLn2gHGZ4pQH?usp=sharing>

```python
import csv
import numpy as np
import matplotlib.pyplot as plt

# Arrays onde serão armazenados os padrões de frequência dos sinais
ss_0 = []
ss_1 = []

 # Abrir o arquivo contendo o padrão
 with open('aut_a.csv', 'r') as csvfile:
     reader = csv.reader(csvfile, delimiter=',', quotechar='|')
     # Depois organizamos os padrões em arrays
     for row in reader:
         walk = np.array([int(s) for s in row])

# Então buscamos o array para cada um dos sinais do padrão:
for i in range(len(walk)):
    if ( walk[i] == 0):
        ss_0.append(1)
        ss_1.append(0)
    else:
        ss_0.append(0)
        ss_1.append(1)

 # Finalmente plotamos os gráficos de barra contínua:
 x = list(range(0, len(walk))) # Cria uma lista com todos os índices de walk
 # Criamos uma figura formada de dois gráficos em uma coluna:
 fig, ax= plt.subplots(nrows=2, ncols=1,figsize=(9,3), constrained_layout=True)
 fig.suptitle('Split Signal - Autômato A', fontsize=14)

ax[0].bar(x,ss_0, color="#d87a00")
ax[0].set_title('0')
ax[0].set_ylim(0,1) # Limitamos o tamanho do eixo y
ax[0].set_xlim(0,190) # Limitamos o tamanho do eixo x
ax[0].set_facecolor('#191970') # mudamos o plano de fundo

ax[1].bar(x,ss_1, color="#d87a00")
ax[1].set_title('1')
ax[1].set_ylim(0,1) # Limitamos o tamanho do eixo y
ax[1].set_xlim(0,190) # Limitamos o tamanho do eixo x
ax[1].set_facecolor('#191970') # mudamos o plano de fundo

plt.show()
```

2-split/split_signal.py [3] - Fonte: Elaborado pelo compilador.

Aqui estão as representações em split signal dos autômatos das figuras 1 e 2:


Split Signal - Autômato A
0
1
Split Signal - Autômato B
0
1
Split Signal - Autômato C
0
1
Split Signal - Autômato d
0
1
2
3
4
5

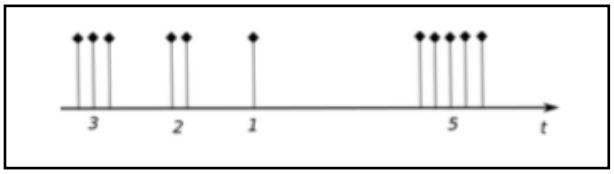


Fonte: Elaborado pelo compilador.

3. Burst

Bursts acontecem quando um sinal produz uma sequência do mesmo símbolo rápidamente. No caso de um split signal do estado 1 um burst seria como uma sequência de dois três ou mais uns, por exemplo.

Figura 3 - Bursts



Fonte: CDT-23[2]

Possíveis atributos que podem vir disso são os tamanhos de bursts (quantos símbolos estão dentro de um burst) e o número de bursts (quantos bursts existem em um sinal). Pode-se então calcular a média, desvio padrão e evenness desses valores, assim como no programam abaixo, baseado no algoritmo 1 do CDT-23[2]:

```python
import csv
import numpy as np
import matplotlib.pyplot as plt

# Arquivos que vamos abrir:
file_list = ['split_a.csv', 'split_b.csv','split_c.csv','split_e.csv',
             'split_fg2.csv']

# Arquivos que vamos escrever:
with open('length_burst.csv', 'w') as csvfile:
    with open('number_burst.csv', 'w') as other_csvfile:
        file = csv.writer(csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)
        other_file = csv.writer(other_csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)

        for f in file_list:

            length_burst = [] # Vetor onde serão guardados os comprimentos
            number_burst = [] #vetor onde será guardada a quantidade de bursts

            # Abrir o arquivo contendo o padrão
            with open(f, 'r') as csvfile:
                reader = csv.reader(csvfile, delimiter=',', quotechar='|')

                # Depois organizamos os padrões em arrays
                for row in reader:
                    walk = np.array([int(s) for s in row])

                    burst = [] # Vetor onde serão guardados os índices
                    j = 0

                    # Parseamos todo o padrão para procurar 1s:
                    for i in range(len(walk)):
                        # Quando achamos um 1 contamos:
                        if ( walk[i] == 1 ):
                            j += 1
                        # Para não dar erro de índice pulamos o primeiro e o último:
                        elif ( i == 0 ):
                            pass
                        elif ( i == 1000 ):
                            # Mas antes vemos se o último é o fim de um burst
                            if ( walk[i-1] == 1 ):
                                burst.append(i-1)
                                length_burst.append(j)
                                j = 0
                            else:
                                pass
                        # Se estiver entre dois 1s o zero faz parte do burst
                        elif ( (walk[i-1] == 1 ) and (walk[i+1] == 1) ):
                            j += 1
                        # Se não reiniciamos o contador e marcamos  os valores
                        elif ( walk[i-1] == 1 ):
                            length_burst.append(j)
                            burst.append(i-1)
                            j = 0

                    number_burst.append(len(burst))

            file.writerow(length_burst)
            other_file.writerow(number_burst)
```

3-burst/burst.py [3] - Fonte: Elaborado pelo compilador.

4. Distância Intersinais

De forma similar aos burst distâncias intencionais é o número de passos (ou zeros) entre em sinal e outro. Podemos obter a distância intersinais marcando o primeiro 1 de um split signal, e contando todos os itens dos conjunto até que se encontre outro 1, como no programa abaixo. Através disso é possível obter também outros atributos como o número de distâncias intersímbolos em um conjunto e split signals.

```python
import csv
import numpy as np
import matplotlib.pyplot as plt

# Arquivos que vamos abrir:
file_list = ['split_a.csv', 'split_b.csv','split_c.csv','split_e.csv',
            'split_fg2.csv']

# Arquivos que vamos escrever:
with open('inter_dist.csv', 'w') as csvfile:
    with open('number_dist.csv', 'w') as other_csvfile:
        file = csv.writer(csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)
        other_file = csv.writer(other_csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)

        for f in file_list:

            inter_dist = [] # Vetor onde serão guardados os valores das distâncias
            number_dist = [] #vetor onde será guardada a quantidade de distâncias

            # Abrir o arquivo contendo o padrão
            with open(f, 'r') as csvfile:
                reader = csv.reader(csvfile, delimiter=',', quotechar='|')

                # Depois organizamos os padrões em arrays
                for row in reader:
                    walk = np.array([int(s) for s in row])

                    j = 0
                    start = 0
                    dist = 0

                    # Parseamos todo o padrão para procurar 1s:
                    for i in range(len(walk)):
                        # Quando achamos um 1 começamos a contagem de 0s:
                        if ( (walk[i] == 1) and (start == 0) ):
                            start = i
                        # Quando achamos o próximo 1 calculamos a distância e
                        # recomeçamos a contagem
                        elif ( (walk[i] == 1) ):
                            dist = (i) - start - 1
                            inter_dist.append(dist)
                            start = i
                            j += 1

                    number_dist.append(j)

            file.writerow(inter_dist)
            other_file.writerow(number_dist)
```

4-interdistancet/interdis.py [3] - Fonte: Elaborado pelo compilador.

5. FFT

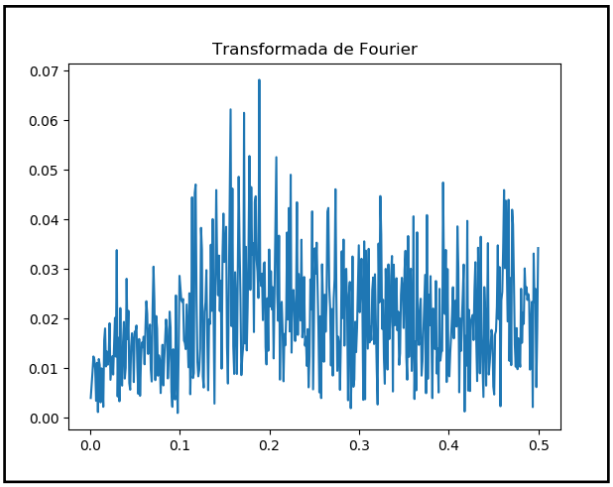
Existe uma transformação que pode ser feita em dados chamada transformada de Fourier. Através dela podemos obter um novo sinal em termos da amplitude e frequência do sinal anterior e este sinal pode ser novamente transformado em um outro atributo chamado espectro de potência, em que o pico será equivalente a distância média entre os sinais.

Abaixo está uma imagem que foi obtida rodando o programam que será apresentado para o split signal do autônomo da figura 2:

Observe que na figura 5 o pico está indicando uma frequência de aproximadamente 0,18 Hz, o que indica um período de 5,33 - exatamente a média da distância intestinais deste autônomo, como será mostrado depois.

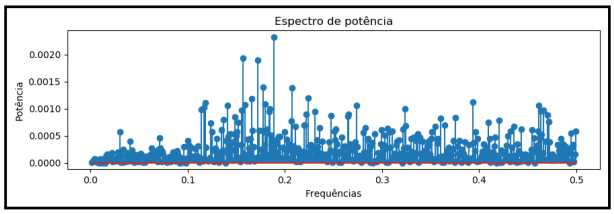
Para obter estes valores foi feito o programa abaixo, que utiliza a função fft do NumPy[4] para realizar a transformada de Fourier:

Figura 4 - FFT do autônomo da figura 2



Fonte: Elaborada pelo compilador.

Figura 5 - Espectro de potência



Fonte: Elaborada pelo compilador.

[4] Referência [5]. Disponível em: <https://numpy.org/doc/1.18/reference/generated/numpy.fft.fft.html>

```python
import numpy as np
import matplotlib.pyplot as plt
import csv

# Primeiro obtemos nosso sinal:

# Lista de arquivos a serem abertos:
file_list = ['split_a.csv', 'split_b.csv','split_c.csv','split_e.csv',
            'split_fg2.csv']

# Arquivo que vamos escrever:
with open('pow_spec.csv', 'w') as csvfile:
    file = csv.writer(csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)

    for f in file_list:

        power_spectrum = []

        # Abrir o arquivo contendo o padrão
        with open(f, 'r') as csvfile:
            reader = csv.reader(csvfile, delimiter=',', quotechar='|')

            # Depois organizamos os padrões em arrays
            for row in reader:
                split = np.array([int(s) for s in row])

                # Depois obtemos o parâmetro de frequências ( ou bins de frequência )
                freq = np.fft.fftfreq(len(split))

                # Para não termos valores duplicados no negativo (por ser uma função com
                # números imaginários) escolhemos apenas os bins com frequências positivas.
                mask = freq > 0

                 # Agora realizamos a FFT usando a função da biblioteca numpy:
                 fourier = np.fft.fft(split)
                 # Então ajustamos os valores para termos apenas representações positivas
                 n_fourier = 2.*abs(fourier/len(split))

                 # Finalmente calculamos o espectro de potência
                 power_spec = (2.*(abs(fourier/len(split))**2))
                 for i in power_spec:
                     power_spectrum.append(i)

         file.writerow(power_spectrum)
```

5-fourier/power_spectrum.py [3] - Fonte: Elaborado pelo compilador.

6. Estátistica

Para ser possível obter informações relevantes a partir dos atributos obtidos pelos programas anteriores é preciso realizar análise estatística sobre eles, por isso foi feito um programa que calcula a média, desvio padrão, entropia e evenness de um conjunto de dados, utilizando as funções da biblioteca NumPy:

```
import numpy as np
import matplotlib.pyplot as plt
from math import log
import csv

i = 0

# Função gaussiana
def gaussian(x, mu, sig):
    return 1/(np.sqrt(2*np.pi)*sig)*np.exp(-(((x - mu)/sig)**2)/2)

# Bins que serão usados para plotar a gaussiana:
bins = np.linspace(-0.05, 0.05, 1000)

# Vetores onde iremos guardar os dados para fazer gráficos:
gaus = [] # Dados das gaussianas

# Primeiro abrimos o arquivo e lemos os dados coletados para cada automato:

file_name = 'inter_dist.csv' # !!! INSIRA AQUI NOME DO ARQUIVO !!!
print('\nDistâncias intersinais') # Dados sendo análisados

# Os arquivos estão organizados de forma que cada linha tem os dados de
# um dos nossos autômatos.
with open(file_name, 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',', quotechar='|')
    # Depois organizamos os padrões em arrays
    for row in reader:
        # Salvamos os dados
        dados = np.array([float(s) for s in row])

        # Identificamos o autômato em que estamos trabalhando
        if (i == 0): aut = 'a:'
        elif (i == 1): aut = 'b:'
         elif (i == 2): aut = 'c:'
         elif (i == 3): aut = 'e:'
         else: aut = 'do CDT-23:'

         print('\nDados do autômato ' + aut)

         ''' MÉDIA, DESVIO E GAUSSIANA '''

         # Calculando a média e o desvio padrão dos dados
         mean = np.mean(dados)
         sdev = np.std(dados)

         # Mapeando a densidade de probabilidade normal (gaussiana):
         gaus.append(gaussian(bins, mean, sdev))

         # Imprimindo valores:
         print('Média: ', mean)
         print('Desvio Padrão: ', sdev)
```

```
        ''' ENTROPIA E EVENNESS '''

        # Calculando a entropia dos dados:
        entp = 0
        for x in range(len(dados)):
            if (dados[x] != 0):
                entp -= dados[x] * log(dados[x],2)

        # Calculando a envenness dos dados:
        evns = 2**entp

        # Imprimindo valores:
        print('Entropia: ', entp)
        print('Evenness: ', evns)

        i = i + 1


# Montando o gráfico da gaussiana:
j = 0
for y in gaus:
    # Identificando o autômato:
    if ( j == 0 ): colors, labels = 'orange', 'Autômato A'
    elif ( j == 1 ): colors, labels = 'limegreen', 'Autômato B'
    elif ( j == 2 ): colors, labels = 'blue', 'Autômato C'
    elif ( j == 3 ): colors, labels = 'yellow', 'Autômato E'
    else: colors, labels = 'red', 'Autômato CDT-23'

    plt.plot(bins, y, label=labels, color=colors)

    j = j + 1

# Plotando o gráfico
plt.xlabel('Distâncias intersinais') # NOME DOS DADOS ANALISADOS
plt.ylabel('Densidade de probabilidade normal')
plt.legend()
plt.show()
```

6-statistics/stat.py [3] - Fonte: Elaborado pelo compilador.

Através dos quais obtiveram-se os seguintes resultados:

```
Comprimento de Bursts

Dados do autômato a:
Média:  1.3498663683261856
Desvio Padrão:  0.8276711940666525
Entropia:  -13658.354058623434
Evenness:  0.0

Dados do autômato b:
Média:  27.964021000617667
Desvio Padrão:  27.302592082407415
Entropia:  -975463.1954738899
Evenness:  0.0

Dados do autômato c:
Média:  4.967474158917668
Desvio Padrão:  4.630172769763478
Entropia:  -352221.54152365815
Evenness:  0.0

Dados do autômato e:
Média:  1.3374938210578349
Desvio Padrão:  0.8208467993206164
Entropia:  -13284.102100612386
Evenness:  0.0

Dados do autômato do CDT-23:
Média:  1.1102198768689533
Desvio Padrão:  0.3539788229020678
Entropia:  -6545.345413019512
Evenness:  0.0
```

```
Número de distâncias intersinais

Dados do autômato a:
Média:  98.68
Desvio Padrão:  9.072904716792742
Entropia:  -130865.76342441283
Evenness:  0.0

Dados do autômato b:
Média:  798.6
Desvio Padrão:  12.531560158256434
Entropia:  -1539941.4622058147
Evenness:  0.0

Dados do autômato c:
Média:  498.855
Desvio Padrão:  15.052042220243736
Entropia:  -894260.9592232688
Evenness:  0.0

Dados do autômato e:
Média:  98.36
Desvio Padrão:  9.33061627117952
Entropia:  -130356.30992574953
Evenness:  0.0

Dados do autômato do CDT-23:
Média:  156.96
Desvio Padrão:  6.315726403193856
Entropia:  -229017.71495063402
Evenness:  0.0
```

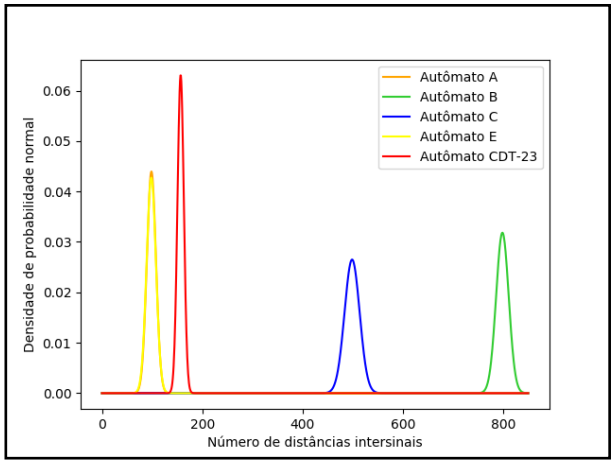
```
Distâncias intersinais

Dados do autômato a:
Média:  8.959616943656263
Desvio Padrão:  9.413212730562698
Entropia:  -681693.7777584483
Evenness:  0.0

Dados do autômato b:
Média:  0.25033183070373155
Desvio Padrão:  0.560088263661621
Entropia:  -17477.724016300268
Evenness:  0.0

Dados do autômato c:
Média:  0.9981457537761473
Desvio Padrão:  1.4112787038720211
Entropia:  -127984.42766511395
Evenness:  0.0

Dados do autômato e:
Média:  8.960349735664904
Desvio Padrão:  9.44966726972062
Entropia:  -678541.9645112578
Evenness:  0.0

Dados do autômato do CDT-23:
Média:  5.336232161060143
Desvio Padrão:  2.974257061762613
Entropia:  -449864.0325787347
Evenness:  0.0
```

```
Número de Bursts

Dados do autômato a:
Média:  80.445
Desvio Padrão:  6.46815081766033
Entropia:  -101917.90964574374
Evenness:  0.0

Dados do autômato b:
Média:  32.38
Desvio Padrão:  5.17741248115311
Entropia:  -32610.33629413783
Evenness:  0.0

Dados do autômato c:
Média:  125.285
Desvio Padrão:  7.071334739637207
Entropia:  -174681.42902702134
Evenness:  0.0

Dados do autômato e:
Média:  80.92
Desvio Padrão:  6.69728303120004
Entropia:  -102660.87104556175
Evenness:  0.0

Dados do autômato do CDT-23:
Média:  142.125
Desvio Padrão:  4.188003701048985
Entropia:  -203285.44690238844
Evenness:  0.0
```

```
Espectro de Potências

Dados do autômato a:
Média:  0.0001989618772835553
Desvio Padrão:  0.0006612991595617504

Dados do autômato b:
Média:  0.0015960063912111863
Desvio Padrão:  0.04033652153029265

Dados do autômato c:
Média:  0.0009977135751361524
Desvio Padrão:  0.015789728648776577

Dados do autômato e:
Média:  0.0001983231553661124
Desvio Padrão:  0.0006581205894740384

Dados do autômato do CDT-23:
Média:  0.0003152891064978977
Desvio Padrão:  0.0016015173190754046
```

Pode-se também obter o gráfico da Gaussiana desses atributos, o que pode as vezes ser interessante, por exemplo, percebe-se pelo gráfico abaixo que a distância intersinais do split signal de 1 pode ser útil para diferenciar entre os autônomos a), b), c) e o autônomo da figura 2, porém não ajudaria ao diferenciar entre a) e e):

Figura 6 - Dens. de prob. Das distâncias intersinais



Fonte: Elaborada pelo compilador.

7. Método de Visibilidade

É possível também obter um modo inteiramente novo de visualizar os sinais, transformando-os em gratos (ou redes) através do método de visibilidade. Abaixo está apresentado um programa implementando o algoritmo 3 para o método de visibilidade do CDT-23[2].

Ao obtermos o grafo dos sinais é possível calcular o grau de cada nódulo do grafo (ou seja quantas arestas esse nódulo tem) e o grau de conectividade dele (quantas prestar ele tem comparado a quantas poderia ter), o que foi feito utilizando a biblioteca NetworkKx[5]

[5] Referência [4]. Disponível em <https://networkx.github.io/documentation/stable/>

```python
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import csv

# Primeiro abrimos nosso sinal:
with open('aut_fg2.csv', 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',', quotechar='|')
    # Depois organizamos o sinal em arrays
    for row in reader:
        signal = np.array([int(s) for s in row])

# Então criamos um grafo vazio:
G = nx.Graph()
G.add_node( i for i in range(len(signal)))

# Em seguida implementamos o algorítimo do método de visibilidade
for j in range(2, len(signal)):
    for i in range(1, (j-1)):
        flag = 1
        k = i + 1
        while ( (k <= (j -1)) and (flag == 1) ):
            aux = signal[j] + (signal[i] - signal[j]) * (j-k)/(j-i)
            if (signal[k] >= aux):
                flag = 0
            k += 1
        if ( flag == 1 ):
            G.add_edges_from([ (i,j), (j,i) ])
# Vamos calcular a média e desvio padrão dos graus  e do coeficiente de
# aglomeração do grafo
deg = []
agl = []
  # Para cada nódulo
  for i in G.nodes():
      # Obter grau:
      deg.append(G.degree(i))
      # Obter coeficiente de alomeração
      agl.append(nx.clustering(G,i))
  # Calculando a média e o desvio padrão dos dados
  mean_deg = np.mean(deg)
  std_deg = np.std(deg)
  mean_agl = np.mean(agl)
  std_agl = np.std(agl)


# Desenhando o grafo gerado:
nx.draw_random(G, node_color='black', node_size=50, width=0.5)
plt.show()
```

7-visibility/vis_method.py [3] - Fonte: Elaborado pelo compilador.

Através desse programa obtemos os seguintes dados:

```
Método de visibilidade para o autômato A

Grau dos nódulos:
Média:  3.448692152917505
Desvio padrão: 7.253520847693647

Coeficiente de aglomeração dos nódulos:
Média:  0.7434061407706207
Desvio padrão: 0.4279220674953991
```

```
Método de visibilidade para o autômato B

Grau dos nódulos:
Média:  1.2814371257485O3
Desvio padrão: 0.5982632303204127

Coeficiente de aglomeração dos nódulos:
Média:  0.028942115768463072
Desvio padrão: 0.14359470659877469
```

```
Método de visibilidade para o autômato C

Grau dos nódulos:
Média:  2.082901554404145
Desvio padrão: 1.5765009709962083

Coeficiente de aglomeração dos nódulos:
Média:  0.24999383172958303
Desvio padrão: 0.3932768822580401
```

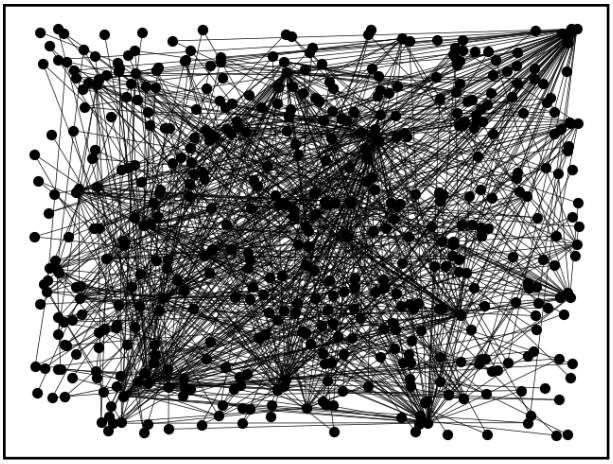
```
Método de visibilidade para o autômato D

Grau dos nódulos:
Média:  3.608173076923077
Desvio padrão: 7.53950050177054

Coeficiente de aglomeração dos nódulos:
Média:  0.4503693801003694
Desvio padrão: 0.45250681614087424
```

```
Método de visibilidade para o autômato E

Grau dos nódulos:
Média:  3.389452332657201
Desvio padrão: 6.098452760268692

Coeficiente de aglomeração dos nódulos:
Média:  0.71999361576417O8
Desvio padrão: 0.4385382484411338
```

```
Método de visibilidade para o autômato do CDT-23

Grau dos nódulos:
Média:  2.956906077348066
Desvio padrão: 2.5794168977920187

Coeficiente de aglomeração dos nódulos:
Média:  0.3281961269871572
Desvio padrão: 0.39552373245056055
```

Por curiosidade também foi obtida uma representação gráfica dos gratos obtidos pelos sinais, como as figuras abaixo:

Figura 7 - Grafo obtido pelo método de visibilidade sobre o autômato A



Fonte: Elaborada pelo compilador.

8. Referências:

[1] COSTA. L. da F. **Where do Patterns To Be Recognized Come From?** Researchgate. Março, 2020 Disponível em: <https://www.researchgate.net/publication/339599069>

[2] COSTA. L. da F. **Discrete One-Dimensional Signals: A Brief Catalogue of Features** Researchgate. Março, 2020 Disponível em: <https://www.researchgate.net/publication/339800429>

[3] **Clustering Coefficient in Graph Theory**. Disponível em: <https://www.geeksforgeeks.org/clustering-coefficient-graph-theory/>

[4] **Overview of NetworkX¶**. Disponível em: <https://networkx.github.io/documentation/stable/>

[5] **numpy.fft.fft¶**. Disponível em: <https://numpy.org/doc/1.18/reference/generated/numpy.fft.fft.html>

[6] **Transformada Rápida de Fourier (FFT) no Python.** Eletrônica e programação. Maio, 2019. Disponível em: <https://www.youtube.com/watch?v=uDz-KirOmw8>

[7] **NumPy Tutorials : 012 : Power Spectrum Analysis.** Fluidic Colors. Maio 2018. Disponível em: <https://www.youtube.com/watch?v=P571FXS33wg>

[8] **Fourier Analysis and Power Spectral Density.** Disponível em: <https://personal.egr.uri.edu/chelidz/documents/mce567_Chapter_4.pdf>

[9] WHITNEY, Daniel. **Network Models and Basic Network Operations.** Janeiro, 2008. Disponível em: <https://ocw.mit.edu/courses/engineering-systems-division/esd-342-network-representations-of-complex-engineering-systems-spring-2010/readings/MITESD_342S10_ntwk_models.pdf>