Documentação sistema especialista

Funções que inserem dados tanto para treino quanto para teste:
insert_training()
insert_test()

Função insert_training() chama a função training(canais), esta função tem o papel de gerar de forma randomizada uma heurística a ser utilizada, após isto esta comprova se a heuristica utilizada neste caso conseguiu obter o resultado igual ao da função verify(), cujo papel é determinar o melhor canal de forma a calcular todas as interferências. Cajo a heurística utilizada fora comprovada, esta é armazena em um banco de dados não relacional Firebase, esta inserção se da pela função store(), esta recebe os valores dos canais de entrada e a heurística que foi utilizada para solucionar esta entrada de dados, esta função armazena no banco de dados a matriz de entrada e a heurística utilizada.

Função insert_test() chama função test(canais), que tem como objetivo calcular a distancia euclidiana entre a matriz de entrada e as matrizes no banco de dados, a distancia euclidiana é calculada da forma, cada linha da matriz de entrada é comparada as linhas das matrizes armazenadas no banco de dados, após isto é feito um calculo somando a distancia euclidiana das linhas, cada interação é calculado um valor de distância total este é comparado com o menor valor já armazenado, ou seja para uma entrada terei que calcular a distancia total para cada matriz armazena no banco de dados, após isto armazeno a menor distância total assim utilizando a heuristica que foi utilizada pela matriz com menor distância total, a partir disto é utilizada a heuristica para resolução com base na matriz de entrada.

Abaixo imagem da função insert_training():

```
[54]: import csv
      import random


      def insert_training():

          arquivo = open('teste.csv')

          redes = csv.reader(arquivo)
          # TREINOOOO
          canais = []
          ver = 0
          cont = 0
          #print(canais)
          for rede in redes:


              coluna = []
              if rede[0] != "SSID":
                  for j in range(0,3):
                      coluna.append(rede[j])

                  canais.append(coluna)
                  #print(cont)
                  cont += 1

              if cont == 15:
                  #print(canais)
                  canais = wifi_vetor(canais)
                  training(canais)
                  cont = 0
                  canais = []
              ver += 1
              #print(ver)
```

Abaixo imagem da função insert_test():

```
def insert_test():
    arquivo = open('REDES.csv')

    redes = csv.reader(arquivo)

    canais = []
    ver = 0
    cont = 0
    #print(canais)
    for rede in redes:


        coluna = []
        if rede[0] != "SSID":
            for j in range(0,3):
                coluna.append(rede[j])

            canais.append(coluna)
            #print(cont)
            cont += 1

        if cont == 16:
            #print(canais)
            canais = wifi_vetor(canais)
            test(canais)
            cont = 0
            canais = []
            ver += 1
```

Abaixo imagem da função store():

```
[58]: import firebase_admin
      from firebase_admin import firestore, db
      import json
      import numpy

      def store(canais, heu):
          db = firestore.client()
          dados = db.collection('dados')
          docs = dados.stream()

          canais = normalize(canais)

          #print(canais)

          enter_matriz = {
              '1' : canais[1],
              '2' : canais[2],
              '3' : canais[3],
              '4' : canais[4],
              '5' : canais[5],
              '6' : canais[6],
              '7' : canais[7],
              '8' : canais[8],
              '9' : canais[9],
              '10' : canais[10],
              '11' : canais[11],
              '12' : canais[12],
              '13' : canais[13],
              '14' : canais[14],
              '15' : heu
          }

          dados.document().set(enter_matriz)
```

Abaixo imagem da função training():

```
[59]: def training(canais):

    print("INICIANDO LEITURA DOS CANAIS")

    c1 = len(canais[1])
    c6 = len(canais[6])
    c11 = len(canais[11])

    comp = verify(canais)
    h = 0
    ver = True


    while(ver == True):
        heu = random.randint(1, 4)

        if heu == 1:
            h = heuristic_1(c1, c6, c11)

        elif heu == 2:
            h = heuristic_2(c1, c6, c11, canais)

        elif heu == 3:
            h = heuristic_3(c1, c6, c11, canais)

        elif heu == 4:
            h = heuristic_4(c1, c6, c11, canais)


        if h == comp:
            ver = False
            print("O Melhor canal é: {} utilizando a heuristica: {}".format(h, heu))

    store(canais, heu)
```

Abaixo imagem da função test():

```
def test(canais):

    canais_aux = canais
    c1 = len(canais[1])
    c6 = len(canais[6])
    c11 = len(canais[11])

    db = firestore.client()
    dados = db.collection('dados')
    docs = dados.stream()

    canais = normalize(canais)
    cont = 0
    heuristica = 0
    distan = 0
    for doc in docs:
        matriz = doc.to_dict()
        #print(matriz)
        euclidean_distance = 0
        for lin in range(1,15):
            #print(lin, matriz[str(lin)])
            euclidean_distance += euclidean(matriz[str(lin)], canais[int(lin)])
        if cont == 0:
            heuristica = matriz[str(15)]
            distan = euclidean_distance
            id = doc.id
        else:
            if distan > euclidean_distance:
                id = doc.id
                heuristica = matriz[str(15)]
                distan = euclidean_distance
        cont += 1
    print(id)
    print("A MENOR DISTANCIA É: {} A HEURISTICA A SER UTILIZADA É {}".format(distan, heuristica))

    if heuristica == 1:
        h = heuristic_1(c1, c6, c11)

    elif heuristica == 2:
        h = heuristic_2(c1, c6, c11, canais)

    elif heuristica == 3:
        h = heuristic_3(c1, c6, c11, canais)

    elif heuristica == 4:
        h = heuristic_3(c1, c6, c11, canais)

    print("O MELHOR CANAL É O CANAL: {}".format(h))
```