Open in Colab

AG3 - Actividad Guiada 3

Nombre: Diocles Germán Avendaño Ponce

Link: https://drive.google.com/file/d/1KRMfM_d-hUOm4DP9xiMZqSqsdvfj1LPh/view?usp=sharing

GitHub:
https://github.com/geharpz/algoritmos_optimizacion_miar2023/blob/main/src/AG3/Algoritmos(Colonia_de_Hormigas)_AG3_Diocles_German_Avendano_Ponce.ipynb

```
[ ] #Modulo de llamadas http para descargar ficheros
    !pip install requests

    #Libreria del problema TSP: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html
    !pip install tsplib95
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.27.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2023.5.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4)
Collecting tsplib95
  Downloading tsplib95-0.7.1-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.10/dist-packages (from tsplib95) (8.1.3)
Collecting Deprecated~=1.2.9 (from tsplib95)
  Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Collecting networkx~=2.1 (from tsplib95)
  Downloading networkx-2.8.8-py3-none-any.whl (2.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 26.0 MB/s eta 0:00:00
Requirement already satisfied: tabulate~=0.8.7 in /usr/local/lib/python3.10/dist-packages (from tsplib95) (0.8.10)
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.10/dist-packages (from Deprecated~=1.2.9->tsplib95) (1.14.1)
Installing collected packages: networkx, Deprecated, tsplib95
  Attempting uninstall: networkx
    Found existing installation: networkx 3.1
    Uninstalling networkx-3.1:
      Successfully uninstalled networkx-3.1
Successfully installed Deprecated-1.2.14 networkx-2.8.8 tsplib95-0.7.1
```

```
[ ] import tsplib95
    import random
    from math import e
    import urllib.request
```

```
    #DATOS DEL PROBLEMA
    file = "swiss42.tsp" ; urllib.request.urlretrieve("http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/swiss42.tsp.gz", file + '.gz')
    !gzip -d swiss42.tsp.gz     #Descomprimir el fichero de datos
    problem = tsplib95.load(file)

    #Nodos
    Nodos = list(problem.get_nodes())

    #Devuelve la distancia entre dos nodos
    def distancia(a,b, problem):
      return problem.get_weight(a,b)

    #Devuelve la distancia total de una trayectoria/solucion(lista de nodos)
    def distancia_total(solucion, problem):
      distancia_total = 0
      for i in range(len(solucion)-1):
        distancia_total += distancia(solucion[i] ,solucion[i+1] ,  problem)
      return distancia_total + distancia(solucion[len(solucion)-1] ,solucion[0], problem)
```

Algoritmo de colonia de hormigas

La función Add_Nodo selecciona al azar un nodo con probabilidad uniforme. Para ser mas eficiente debería seleccionar el próximo nodo siguiendo la probabilidad correspondiente a la ecuación:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\nu_{ij}]^\beta}{\sum_{k \in J_i^k} [\tau_{il}(t)]^\alpha [\nu_{il}]^\beta}, \text{si } j \in J_i^k$$

$$p_{ij}^k(t) = 0, \text{si } j \notin J_i^k$$

```
[ ] def Add_Nodo(problem, H ,T ) :
      #Mejora:Establecer una funcion de probabilidad para
      # añadir un nuevo nodo dependiendo de los nodos mas cercanos y de las feromonas depositadas
      Nodos = list(problem.get_nodes())
      return random.choice(   list(set(range(1,len(Nodos))) - set(H) )  )


    def Incrementa_Feromona(problem, T, H ) :
      #Incrementa segun la calidad de la solución. Añadir una cantidad inversamente proporcional a la distancia total
      for i in range(len(H)-1):
        T[H[i]][H[i+1]] += 1000/distancia_total(H, problem)
      return T

    def Evaporar_Feromonas(T ):
      #Evapora 0.3 el valor de la feromona, sin que baje de 1
      #Mejora:Podemos elegir diferentes funciones de evaporación dependiendo de la cantidad actual y de la suma total de feromonas depositadas,...
      T = [[ max(T[i][j] - 0.3 , 1) for i in range(len(Nodos)) ] for j in range(len(Nodos))]
      return T
```

```
[ ] def hormigas(problem, N) :
      #problem = datos del problema
      #N = Número de agentes(hormigas)

      #Nodos
      Nodos = list(problem.get_nodes())
      #Aristas
      Aristas = list(problem.get_edges())

      #Inicializa las aristas con una cantidad inicial de feromonas:1
      #Mejora: inicializar con valores diferentes dependiendo diferentes criterios
      T = [[ 1 for _ in range(len(Nodos)) ] for _ in range(len(Nodos))]

      #Se generan los agentes(hormigas) que serán estructuras de caminos desde 0
      Hormiga = [[0] for _ in range(N)]

      #Recorre cada agente construyendo la solución
      for h in range(N) :
        #Para cada agente se construye un camino
        for i in range(len(Nodos)-1) :

          #Elige el siguiente nodo
          Nuevo_Nodo = Add_Nodo(problem, Hormiga[h] ,T )
          Hormiga[h].append(Nuevo_Nodo)

        #Incrementa feromonas en esa arista
        T = Incrementa_Feromona(problem, T, Hormiga[h] )
        #print("Feromonas(1)", T)

        #Evapora Feromonas
        T = Evaporar_Feromonas(T)
        #print("Feromonas(2)", T)

        #Seleccionamos el mejor agente
      mejor_solucion = []
      mejor_distancia = 10e100
      for h in range(N) :
        distancia_actual = distancia_total(Hormiga[h], problem)
        if distancia_actual < mejor_distancia:
          mejor_solucion = Hormiga[h]
          mejor_distancia =distancia_actual


      print(mejor_solucion)
      print(mejor_distancia)


    hormigas(problem, 1000)
```

```
[0, 6, 14, 5, 13, 12, 33, 20, 3, 24, 11, 15, 19, 26, 4, 7, 36, 32, 16, 17, 18, 25, 31, 9, 10, 21, 23, 41, 8, 40, 27, 28, 22, 30, 34, 2, 39, 38, 29, 35, 37, 1]
3696
```