

REPORTE TAREA #2

INTELIGENCIA

COMPUTACIONAL

línea horizontal

Jesús Rivera Flores - A00820643

Diego Arnoldo Azuela Rosas - A01208345



**CONTENIDO**

# 

[**1. DESCRIPCIÓN DEL PROBLEMA**](#_1d3blirkvx7b) **3**

[Instrucciones](#_albjccy15t9v) 3

[Interpretación del Problema](#_ciwlzli16y57) 3

[**2. DESCRIPCIÓN DE LA REPRESENTACIÓN**](#_tnnn92xp2u3i) **3**

[Describir el formato en el que se representan las soluciones del problema](#_wjw3hbkbzoul) 3

[Se presenta en el reporte los resultados obtenidos por los algoritmos.](#_h0oay9airwd1) 3

[**3. CONCLUSIÓN Y RETOS ENCONTRADOS**](#_3olafbe5mvvh) **6**

# 

# 1. INSTRUCCIONES

## Instrucciones

1. Programar el algoritmo de recocido simulado y resolver el problema del vendedor viajero para K vendedores.
2. El algoritmo puede tener cualquier criterio de terminación (tiempo, iteraciones, iteraciones sin mejora, temperatura cercana a cero, etc.) y deben especificar en el reporte cual utilizaron. El problema tiene las siguientes restricciones:
   1. Se deben visitar todas las ciudades.
   2. Cada ciudad debe ser visitada solamente por un vendedor

## Interpretación del Problema

* El alumno trata con el algoritmo de recocido simulado para resolver el problema del vendedor viajero. El algoritmo de recocido simulado tiene una gran variedad de aplicaciones, el equipo investigó y se pudo observar que es un algoritmo robusto y en donde la única parte en donde realmente se deben aplicar modificaciones es en la mutación de las soluciones actuales.

Se inicia con una temperatura ‘x’, el algoritmo va aceptando soluciones hasta que llega a un ‘t’ de temperatura en donde se encuentra debajo de un umbral aceptado. Se cuenta con estas condiciones dentro de un ciclo ‘while’.

Dentro de este ciclo while se cuenta con un ciclo ‘For’ que le permitirá al equipo determinar el número de iteraciones que se tendrán de acuerdo a esta condición. En este caso se le otorga una variable ‘L’ que determinará el fin de este ciclo.

Para sustituir la solución actual con una nueva solución propuesta, se debe comparar el resultado de la operación [y = solución\_actual-solución\_propuesta], en este caso si el valor de ‘y’ es negativo, se reemplaza automáticamente. En caso contrario, se compara una función exponencial con un valor aleatorio para ver si se acepta la solución propuesta o no.

Una vez que se cumple lo anterior, se hará la mutación de la temperatura multiplicando la temperatura actual por un factor de 0.9, siempre obteniendo el 90% del valor anterior. Esto se vuelve un proceso iterativo que sale de los valores de mínimo local.

# 2. DESCRIPCIÓN DE LA REPRESENTACIÓN

## Describir el formato en el que se representan las soluciones del problema

* El IDE utilizado es Colab de Google. Formato similar al presentado por Jupyter Notebooks, pero con la versatilidad de Google que permite acceder a la información de diferentes puntos y con la seguridad de que todo se ha guardado en la nube. Se presentará el código junto con una foto de la gráfica que produce el código.
* Valor de ‘L’, es un valor establecido numérico
* ‘T’ decrece con un valor lineal constante de 0.9\*T
* Se cuentan con 5 vendedores y un número de 20 ciudades.

## Se presenta en el reporte los resultados obtenidos por los algoritmos.

| **CÓDIGO** | |
| --- | --- |
| **LIBRERÍAS UTILIZADAS** | import math  import numpy as np  from functools import partial  import matplotlib.pyplot as plt  from scipy.spatial import distance\_matrix  from multiprocessing.dummy import Pool as ThreadPool |
| **INICIALIZAR VARIABLES** | # Definir el número de ciudades a visitar.  n\_cities = 20  # Generar las coordenadas para n ciudades.  cities = np.random.randint(1, 100, size=(n\_cities, 2))  # Se toman los valores del eje 'x' y eje 'y'  cx = cities[:, 0]  cy = cities[:, 1]  # Se plotean las ciudades  fig, ax = plt.subplots()  ax.scatter(cx, cy) |
| **DEFINIR FUNCIONES** | def recocido (T0, a, L, Tf):  T = T0  soluciones = []  Sa = genera\_solucion\_inicial()  soluciones.append(Sa)  while T >= Tf:  for count in range(L):  Sc = genera\_vecino\_aleatorio(Sa)  soluciones.append(Sc)  d = cost(Sc)-cost(Sa)  if np.random.uniform(0,1) < math.exp(-d/T) or d < 0:  Sa = Sc  T = T\*a  return Sa |
| def cost(ruta):  cost = 0  for vend in ruta:  for x in range(-1,3):  cost += math.sqrt((vend[x][0] - vend[x+1][0])\*\*2 + abs(vend[x][1] - vend[x+1][1])\*\*2)  return cost |
| def genera\_vecino\_aleatorio(ruta):  aux = np.copy(ruta)  Xa = np.random.randint(0,5)  Xb = np.random.randint(0,5)  Ya = np.random.randint(0,4)  Yb = np.random.randint(0,4)  aux2 = np.copy(aux[Xa][Ya])  aux[Xa][Ya] = np.copy(aux[Xb][Yb])  aux[Xb][Yb] = aux2  return aux |
| def genera\_solucion\_inicial():  np.random.shuffle(cities)  arr = np.reshape(cities, (5,4,2))  return arr |
| def graficarMejorSolucion(caminos, ciudades):  aux = np.append(caminos[0],caminos[0][0])  print(aux)  print(np.reshape(aux, (5,2)))  caminos = [np.reshape(np.append(cam,cam[0]), (5,2)) for cam in caminos]  aux = caminos  cx = ciudades[:, 0]  cy = ciudades[:, 1]    s = plt.scatter(cx, cy, c='r')  fig, ax = plt.subplots()  ax.plot(aux[0][:,0], aux[0][:,1], c='r')  ax.plot(aux[1][:,0], aux[1][:,1], c='g')  ax.plot(aux[2][:,0], aux[2][:,1], c='b')  ax.plot(aux[3][:,0], aux[3][:,1], c='y')  ax.plot(aux[4][:,0], aux[4][:,1], c='m')  graficarMejorSolucion(recocido(40,.9,10, .1),cities) |
|  |  |

# 3. CONCLUSIÓN Y RETOS ENCONTRADOS

La realización de este código fue interesante y gratificante. A través de la investigación preliminar realizada el equipo indaga y descubre que el algoritmo de recocido simulado es uno de los más utilizados en la industria y con una gran variedad de aplicaciones. Uno de los más grandes atributos y de las razones por las cuales es tan usado es que es un algoritmo robusto que evita los mínimos locales dado que acepta nuevas soluciones basado en valores aleatorios cuando se rechaza ciertos valores.

Las barreras más importantes fueron las de entender la manera en que se debían tratar las variables iniciales, la manera en que se iba disminuyendo la temperatura y determinar un valor apropiado para ‘L’. Esto con el fin de tener un algoritmo funcional, pero que realmente arroja soluciones posibles para cada uno de los vendedores.

Estas fueron las primeras barreras, sin embargo una de las más complicadas fue la parte de graficar las respuestas. El equipo tuvo que tener varias iteraciones para determinar la viabilidad de las soluciones.

Al final, se presenta una solución para cada uno de los 5 vendedores viajeros, se presenta una lista con las ciudades que han visitado y se presenta una gráfica que denota las diferentes localidades que se han visitado.

| **SOLUCIONES ÓPTIMAS ENCONTRADAS** | |
| --- | --- |
|  |  |

# 

# 4. BIBLIOGRAFÍAS

Galán, José Manuel (24 Mar,2020) Recocido simulado (1 de 8). Algoritmo Básico II.

*YOUTUBE.* Recuperado el 06 de Febrero del 2021, de: https://www.youtube.com/watch?v=Xe63l4kpuMg