UNIVERSIDAD AUTONOMA DE NUEVO LEON.

FACULTAD DE INGENIERIA MECÁNICA Y ELECTRICA.

TAREA 7: ALGORITMO HEURISTICO PARA PROBLEMA DE TRAVELING SALESMAN PROBLEM CON MULTIARRANQUE

PRESENTADO POR:

JESUS JAVIER MORENO VAZQUEZ 1619830

HORA: V4-V6

PROFESOR: DR. MARIA ANGELICA SALAZAR

SAN NICOLAS DE LOS GARZA, NUEVO LEON, A 03 DE ABRIL DEL 2017

**DESCRIPCION GENERAL DEL PROBLEMA**

Se tiene un conjunto de N nodos o ciudades, cada ciudad tiene una distancia o costo para llegar a otra ciudad, entonces, se requiere visitar cada una de las ciudades o nodos y regresar a la ciudad origen buscando siempre minimizar el costo o la distancia recorrida, dependiendo del caso.

Además, a la solución presentada se le tendrá que hacer un movimiento para intentar mejorar la solución.

**ALGORITMO PROPUESTO:**

1.- Lectura de Datos

2.- Calcular las distancias euclidianas.

3.- Calcular solución inicial con el método constructivo greedy o vecino más cercano, tomando como ciudad inicial siempre la primera.

\*\* Se había pensado sacar un promedio de las distancias de cada ciudad, y comenzar tomando la mejor distancia de la ciudad con el promedio más alto, sin embargo, los resultados no favorecían la utilización de este método. \*\*

4.- Calcular el tamaño de las aristas.

5.- Calcular la arista más pesada.

6.- Tomar dichos nodos (los de la arista más pesada)

7.- Realizar el movimiento: Se toman cadenas de dos de la solución inicial y se va recorriendo y evaluando, si encuentra una mejor, esa será la nueva solución inicial y de ahí sigue tomando cadenas para recorrer, se detiene cuando hayan pasado 100 iteraciones sin encontrar una mejor solución.

8.- Revisar posibles mejores soluciones utilizando método multiarranque GRISP.

8.- Reportar resultados.

**RESULTADOS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instancia | Distancia Calculada | Distancia Óptima | GAP Anterior | GAP Multi | Tiempo |
| Eil51 | **463** | **426** | **20.56%** | **8.6%** | **.095596** |
| Berlin52 | **8266** | **7542** | **19.07%** | **9.5%** | **.8980** |
| Eil76 | **592.14** | **538** | **28.93%** | **10 %** | **.9523** |
| Rat99 | **1372.66** | **1211** | **29.20%** | **13.34%** | **2.0963** |
| kroA100 | **23998.41** | **21282** | **26.19%** | **12.7 %** | **2.523** |
| Eil101 | **699.28** | **629** | **31.19%** | **11.17%** | **3.02** |
| Lin105 | **18285.27** | **14379** | **41.61%** | **27.16%** | **4.54** |
| Bier127 | **126415.06** | **118282** | **14.76%** | **6.87%** | **8.593606** |
| Ch130 | **7132.2862** | **6110** | **23.98%** | **16.73%** | **8.48** |
| kroB150 | **30573.33** | **26130** | **25.62%** | **17%** | **12.7156** |
| Ch150 | **7171.60** | **6528** | **25.53%** | **9.8%** | **13.007861** |
| kroB200 | **35633.03** | **29368** | **25.92%** | **21.33%** | **30.512505** |
| Ts225 | **136997.69** | **126643** | **20.41%** | **8.17%** | **39.638751** |
| Pr226 | **94052.51** | **80369** | **17.81%** | **16.98%** | **42.076144** |
| Gil262 | **2980.03** | **2378** | **36.31%** | **25.31%** | **47.235933** |
| A280 | **3138.7** | **2579** | **22.07%** | **21%** | **78.238727** |
| Lin318 | **52175.86** | **42029** | **28.56%** | **24.14%** | **101.1445** |
| Pr439 | **131072.02** | **107217** | **22.44%** | **22.2%** | **255.6998** |
| Dsj1000 | **24630960.1014** | **18659688** | **32.00%** | **32%** | **3326.69** |
| Pr1002 | **315596.58** | **259045** | **21.83%** | **21.83%** | **3259.5943** |

x = solución inicial

f(x) = Valor de función objetivo de x

N(x) = Vecindario

Repetir

x’ <- argmin { f(x´´)}

Si f(x´) <= f(x)

x🡨 x´

**CARACTERISTICAS DEL SISTEMA**

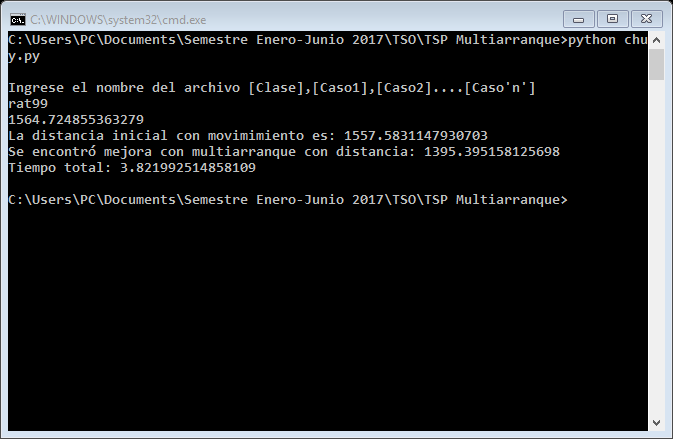
Procesador: Intel® Core i5-4590 CPU @ 3.30 GHz

Memoria Instalada (RAM): 8.00 GB (7.88 GB utilizable)

Sistema Operativo: Windows 10 Pro

Tipo de sistema: Sistema operativo de 64 bits, procesador x64

**Ejemplo**

****

**CONCLUSIONES**

En conclusión, con esta actividad pude percatarme del por qué la actividad anterior estaba equivocada, teniendo en cuenta que mi vecindario era más cerrado, yo igualaba dos arrays para poder hacer modificaciones, sin embargo, ocurría algo que tenía que ver con mutabilidad e inmutabilidad, los dos arrays a pesar de tener nombres diferentes tenían un ID idéntico, por lo que al modificar uno, automáticamente se modificaba el otro, dándome resultados erróneos, con esta tarea pude idéntica este problema, aumenté mi vecindario y encontré mejoras en los resultados.

**BIBLIOGRAFIA**

* https://www.python.org/doc/