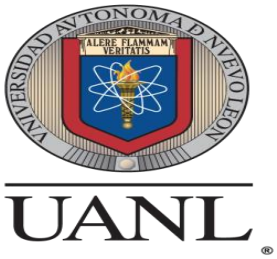


Universidad Autónoma de Nuevo León.

Facultad de Ciencias Físico Matemáticas.

Matemáticas Computacionales

Profesor José Anastacio Hernández Saldaña

Hernández Quintero Alexandra Valeria 1663507

Rivera Rodríguez Mayra Alejandra 1742899

Ruiz Fraser Guillermo Javier 1837506

Segovia Olay Isaac Emanuel 1748957

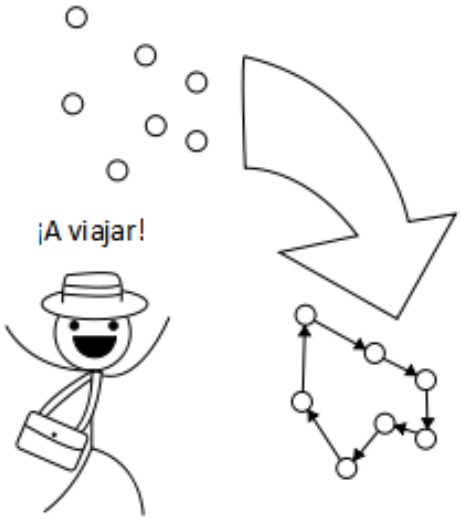
Villanueva Andrade Daniela Estefanía 1684742

27 noviembre 2017

El problema del agente viajero

Este problema es considerado como un conjunto de grafos cuyas aristas son vistas como las rutas a recorrer para visitar a todos los nodos.

El objetivo que nos plantea este problema es encontrar un recorrido que conecte todos los nodos de un conjunto de modo que los visite solo una vez, regresando a su nodo de inicio y además de esto minimice la distancia que se tiene que recorrer.

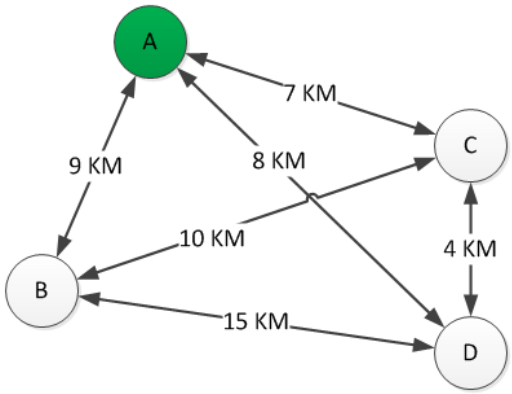


El PAV se clasifica como un problema de optimización combinatoria, es decir, es un problema en el que intervienen cierto número de variables donde cada variable puede tener n valores diferentes por el cual se hacen combinaciones entre ellos y esto da ligar a múltiples soluciones que se calculan en un tiempo finito.

Se considera un problema complicado de resolver por qué no se puede garantizar que se encontrara la mejor solución en un tiempo razonable para los sistemas de cómputo.

Lo más difícil de este problema se podría decir que es la variación que existe en cuanto a las distancias ya que el número de rutas dependerá si las rutas son simétricas o no.

La cantidad de rutas posibles están determinadas por la ecuación (n-1)! Si son simétricas y si no lo fueran se reduciría a la mitad es decir (n-1)! /2.



La complejidad del cálculo del problema del agente viajero ha despertado múltiples iniciativas para mejorar la eficiencia del cálculo de rutas.

Uno de los métodos más básicos que se conoce lleva el nombre de fuerza bruta y este consiste en calcular todos los recorridos posibles, lo cual es muy ineficiente; aunque

también existen otros heurísticos que se han desarrollado para facilitar el cálculo de las soluciones posibles al problema algunos ejemplos de ellos son métodos como el del vecino más cercano, la inserción más barata y el doble sentido.

Descripción del ejemplo:

Nuestro problema consiste en llevar a cabo la implementación de un algoritmo de aproximación usando el algoritmo de kruskal para obtener un árbol de expansión mínima y a partir de él crear una solución aproximada al problema del agente viajero. En primera instancia se eligieron 50 ciudades que representan los nodos y se calcularon las distancias existentes entre ellas representando las aristas de nuestro grafo. Partiendo de lo anterior se tuvo como objetivo encontrar un recorrido que conecte se todas las ciudades, visitando cada una de ellas para después volver a nuestro punto de partida, buscando minimizar la distancia de la ruta.

Las 50 ciudades son las siguientes:

1. Ciudad Mante, Tamaulipas.	19. Tlayacapan, Morelos.	37. Oaxaca, Oaxaca
2. Culiacán, Sinaloa.	20. Puebla, Puebla.	38. Querétaro, Querétaro
3. Ciudad Juárez, Chihuahua.	21. Chetumal, Quintana Roo	39. Cancún, Quintana Roo
4. Nayarit, Tepic.	22. Tuxtla Gutiérrez, Chiapas	40. Nuevo Laredo, Tamaulipas
5. Ciudad Valles, San Luis P.	23. Chilpancingo, Guerrero	41. León, Guanajuato
6. Veracruz, Veracruz.	24. Pachuca, Hidalgo	42. Guanajuato, Guanajuato
7. Los Cabos, Baja California S.	25. Cuernavaca, Morelos	43. Tijuana, Baja California
8. Acapulco, Guerrero.	26. Hermosillo, Sonora	44. Chihuahua, Chihuahua
9. San Miguel de Allende, Gto.	27. Mérida, Yucatán	45. Guadalupe, Nuevo León
10. Morelia, Michoacán.	28. Villahermosa, Tabasco	46. Colima, Colima
11. Cd. Victoria, Tamaulipas.	29. Xalapa, Veracruz	47. Tampico, Tamaulipas
12. China, Nuevo León.	30. Toluca, México	48. Los Mochis, Sinaloa
13. Ozuluama, Veracruz.	31. Aguascalientes, Aguascalientes	49. Ojo de Agua, México
14. Mazatlán, Sinaloa.	32. Mexicali, Baja California	50. La Paz, Baja California Sur
15. Ensenada, Baja California.	33. Ciudad del Carmen, Campeche	
16. Santa Lucía del Camino, Oaxaca.	34. Saltillo, Coahuila de Zaragoza	
17. Tamasopo, San Luis Potosí.	35. Tula, Hidalgo	
18. Guadalajara, Jalisco.	36. Monterrey, Nuevo León	

Heurística:

La heurística consistió en tomar una ciudad al azar, visitar a su vez a su ciudad vecina que no haya sido visitada anteriormente, básicamente tratamos de movernos de una ciudad a la siguiente de manera tal, que de todas las opciones, la ciudad elegida sea lo más cercana al punto en el que se encuentra el viaje seleccionando las distancias más cortas.

Implementación:

```
def vecinoMasCercano(self):
    ni = random.choice(list(self.V))
    result=[ni]
    while len(result) < len(self.V):
        ln = set(self.vecinos[ni])
        le = dict()
        res =(ln-set(result))
        for nv in res:
            le[nv]=self.E[(ni,nv)]
        menor = min(le, key=le.get)
        result.append(menor)
        ni=menor
    return result
```

Algoritmo de aproximación:

Se utilizó también un algoritmo de aproximación permitiendo maximizar la posibilidad de obtener un buen resultado en cierto periodo de tiempo reduciendo la complejidad, y encontrando así la solución más óptima.

Se implementó el algoritmo de **kruskal**, el cual se encarga de encontrar el valor de la suma de todas las distancias entre las ciudades de nuestro grafo.

Así mismo hicimos uso de un **árbol de expansión mínima**, determinado por nuestras distancias y ciudades, esto para buscar aquella suma mínima que existe las aristas (distancias) que conectan a los vértices o nodos (ciudades).

Implementación del algoritmo de aproximación usando kruskal

```
k=g.kruskal()
print(k)#Imprime el arbol de expansion minima
tim=time.clock()
mejor=-1
camino=[]
for r in range(10):
    ni=random.choice(list(k.V))
    dfs=k.DFS(ni)
    peso=0
    for i in range(len(dfs)-1):
        peso+=g.E[(dfs[i],dfs[i+1])]
    peso+=g.E[(dfs[-1],dfs[0])]
```

```
    print("Nodo inicial ",ni)
    for i in range(len(dfs)-1):
        print("De ",dfs[i],"\ta",dfs[i+1],"\tla distancia es ",g.E[(dfs[i],dfs[i+1])])
    print("De ",dfs[-1],"\ta ",dfs[0],"\tla distancia es ",g.E[(dfs[-1],dfs[0])])

    print("Costo total: ",peso,"\n")
    if mejor==-1 or mejor>peso:
        mejor=peso
        camino=dfs
print("La mejor ruta es la siguiente: ")
for k in camino:
    print(k,'->')
print(camino[0])
print("\nCon un costo de ",mejor)
print("Tiempo de ejecucion: ",time.clock()-tim)
print("\n--------------------------------------------------\n\n")
```

Resultados de la heurística:

El mejor camino fue el siguiente:	El mejor camino fue el siguiente:	El mejor camino fue el siguiente:	El mejor camino fue el siguiente:	El mejor camino fue el siguiente:
7 ->	31 ->	47 ->	10 ->	9 ->
50 ->	41 ->	13 ->	25 ->	38 ->
44 ->	42 ->	17 ->	19 ->	42 ->
43 ->	9 ->	5 ->	20 ->	41 ->
15 ->	38 ->	1 ->	24 ->	31 ->
32 ->	35 ->	11 ->	49 ->	18 ->
26 ->	49 ->	12 ->	35 ->	4 ->
48 ->	24 ->	45 ->	30 ->	7 ->
2 ->	20 ->	36 ->	21 ->	50 ->
14 ->	25 ->	34 ->	27 ->	44 ->
4 ->	10 ->	23 ->	39 ->	43 ->
18 ->	30 ->	8 ->	33 ->	15 ->
41 ->	21 ->	25 ->	28 ->	32 ->
42 ->	27 ->	10 ->	22 ->	26 ->
9 ->	39 ->	42 ->	16 ->	48 ->
38 ->	33 ->	41 ->	37 ->	2 ->
35 ->	28 ->	31 ->	29 ->	14 ->
49 ->	22 ->	18 ->	6 ->	10 ->
24 ->	16 ->	4 ->	1 ->	25 ->
20 ->	37 ->	7 ->	5 ->	19 ->
25 ->	29 ->	50 ->	17 ->	20 ->
10 ->	6 ->	44 ->	13 ->	24 ->
30 ->	1 ->	43 ->	47 ->	49 ->
21 ->	5 ->	15 ->	11 ->	35 ->
27 ->	17 ->	32 ->	12 ->	30 ->
39 ->	13 ->	26 ->	45 ->	21 ->
33 ->	47 ->	48 ->	36 ->	27 ->
28 ->	11 ->	2 ->	34 ->	39 ->

22 ->	12 ->	14 ->	23 ->	33 ->
16 ->	45 ->	46 ->	8 ->	28 ->
37 ->	36 ->	30 ->	38 ->	22 ->
29 ->	34 ->	35 ->	9 ->	16 ->
6 ->	23 ->	49 ->	42 ->	37 ->
1 ->	8 ->	24 ->	41 ->	29 ->
5 ->	19 ->	20 ->	31 ->	6 ->
17 ->	18 ->	19 ->	18 ->	1 ->
13 ->	4 ->	29 ->	4 ->	5 ->
47 ->	7 ->	6 ->	7 ->	17 ->
11 ->	50 ->	37 ->	50 ->	13 ->
12 ->	44 ->	16 ->	44 ->	47 ->
45 ->	43 ->	22 ->	43 ->	11 ->
36 ->	15 ->	28 ->	15 ->	12 ->
34 ->	32 ->	33 ->	32 ->	45 ->
23 ->	26 ->	21 ->	26 ->	36 ->
8 ->	48 ->	27 ->	48 ->	34 ->
19 ->	2 ->	39 ->	2 ->	23 ->
31 ->	14 ->	38 ->	14 ->	8 ->
46 ->	46 ->	9 ->	46 ->	46 ->
40 ->	40 ->	40 ->	40 ->	40 ->
3 ->	3 ->	3 ->	3 ->	3 ->
7	31	47	10	9
Con un costo de: 12942.68	**Con un costo de:** 12851.1	**Con un costo de:** 13415.099999999999	**Con un costo de:** 12852.1	**Con un costo de:** 12819.1
Tiempo de ejecución: 0.1660189884105714	**Tiempo de ejecución:** 0.059384336948046984	**Tiempo de ejecución:** 0.05587450423227927	**Tiempo de ejecución:** 0.05460412751810479	**Tiempo de ejecución:** 0.07197004887999897

El mejor camino fue el siguiente:	**El mejor camino fue el siguiente:**	**El mejor camino fue el siguiente:**	**El mejor camino fue el siguiente:**	**El mejor camino fue el siguiente:**
6 ->	46 ->	50 ->	14 ->	38 ->
29 ->	18 ->	44 ->	2 ->	9 ->
20 ->	4 ->	43 ->	48 ->	42 ->
25 ->	7 ->	15 ->	50 ->	41 ->
10 ->	50 ->	32 ->	44 ->	31 ->
42 ->	44 ->	26 ->	43 ->	18 ->
41 ->	43 ->	48 ->	15 ->	4 ->
31 ->	15 ->	2 ->	32 ->	7 ->
18 ->	32 ->	14 ->	26 ->	50 ->
4 ->	26 ->	4 ->	3 ->	44 ->
7 ->	48 ->	18 ->	40 ->	43 ->

50 ->	2 ->	31 ->	45 ->	15 ->
44 ->	14 ->	41 ->	36 ->	32 ->
43 ->	31 ->	42 ->	34 ->	26 ->
15 ->	41 ->	9 ->	23 ->	48 ->
32 ->	42 ->	38 ->	8 ->	2 ->
26 ->	9 ->	35 ->	25 ->	14 ->
48 ->	38 ->	49 ->	10 ->	10 ->
2 ->	35 ->	24 ->	42 ->	25 ->
14 ->	49 ->	20 ->	41 ->	19 ->
46 ->	24 ->	25 ->	31 ->	20 ->
30 ->	20 ->	10 ->	18 ->	24 ->
35 ->	25 ->	30 ->	4 ->	49 ->
49 ->	10 ->	21 ->	7 ->	35 ->
24 ->	30 ->	27 ->	1 ->	30 ->
19 ->	21 ->	39 ->	5 ->	21 ->
23 ->	27 ->	33 ->	17 ->	27 ->
8 ->	39 ->	28 ->	13 ->	39 ->
38 ->	33 ->	22 ->	47 ->	33 ->
9 ->	28 ->	16 ->	11 ->	28 ->
1 ->	22 ->	37 ->	12 ->	22 ->
5 ->	16 ->	29 ->	9 ->	16 ->
17 ->	37 ->	6 ->	38 ->	37 ->
13 ->	29 ->	1 ->	35 ->	29 ->
47 ->	6 ->	5 ->	49 ->	6 ->
11 ->	1 ->	17 ->	24 ->	1 ->
12 ->	5 ->	13 ->	20 ->	5 ->
45 ->	17 ->	47 ->	19 ->	17 ->
36 ->	13 ->	11 ->	30 ->	13 ->
34 ->	47 ->	12 ->	21 ->	47 ->
40 ->	11 ->	45 ->	27 ->	11 ->
3 ->	12 ->	36 ->	39 ->	12 ->
22 ->	45 ->	34 ->	33 ->	45 ->
28 ->	36 ->	23 ->	28 ->	36 ->
33 ->	34 ->	8 ->	22 ->	34 ->
21 ->	23 ->	19 ->	16 ->	23 ->
27 ->	8 ->	46 ->	37 ->	8 ->
39 ->	19 ->	40 ->	29 ->	46 ->
16 ->	40 ->	3 ->	6 ->	40 ->
37 ->	3 ->	7 ->	46 ->	3 ->
6	46	50	14	38
Con un costo de: 13763.1	**Con un costo de:** 13006.1	**Con un costo de:** 12919.68	**Con un costo de:** 13202.099999999999	**Con un costo de:** 12822.1
Tiempo de ejecución:	**Tiempo de ejecución:**	**Tiempo de ejecución:**	**Tiempo de ejecución:**	**Tiempo de ejecución:**

0.055707687087993736	0.0572203336240229	0.05126419164774789	0.074310621735196	0.05710227841422083

Resultados del algoritmo de aproximación:

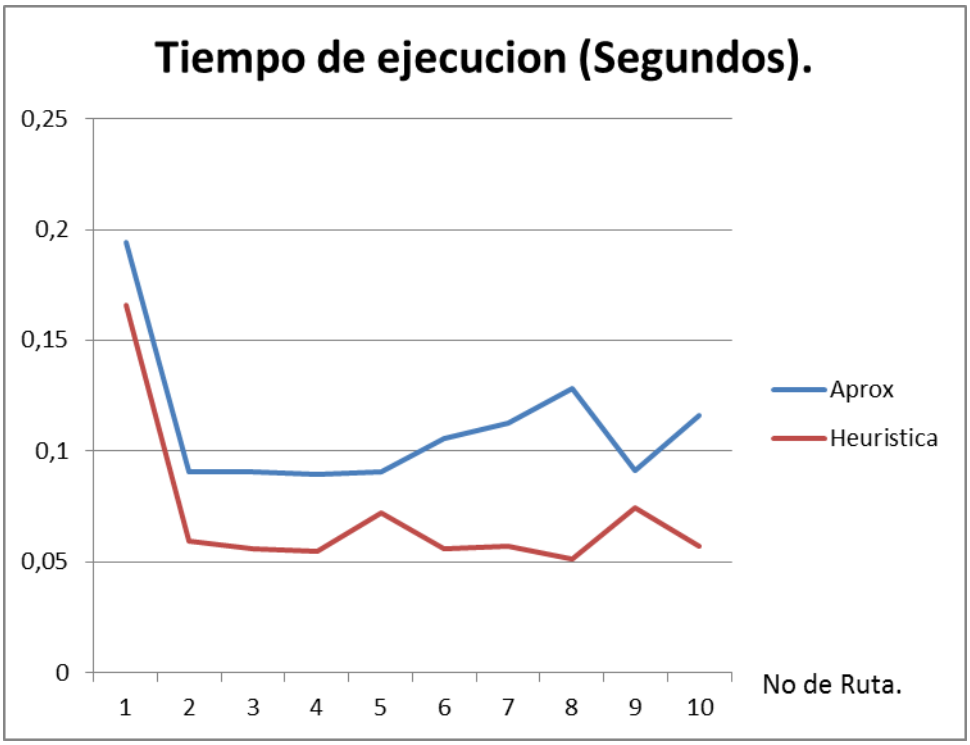
La mejor ruta es la siguiente:	La mejor ruta es la siguiente:	La mejor ruta es la siguiente:	La mejor ruta es la siguiente:	La mejor ruta es la siguiente:
32 ->	12 ->	1 ->	2 ->	23 ->
43 ->	45 ->	47 ->	14 ->	34 ->
15 ->	36 ->	13 ->	4 ->	36 ->
44 ->	34 ->	5 ->	7 ->	45 ->
38 ->	23 ->	17 ->	50 ->	40 ->
9 ->	8 ->	11 ->	44 ->	12 ->
42 ->	25 ->	12 ->	38 ->	11 ->
41 ->	19 ->	45 ->	35 ->	1 ->
31 ->	20 ->	36 ->	49 ->	47 ->
35 ->	37 ->	34 ->	25 ->	13 ->
30 ->	16 ->	23 ->	19 ->	5 ->
21 ->	29 ->	8 ->	10 ->	17 ->
39 ->	6 ->	25 ->	20 ->	25 ->
27 ->	49 ->	49 ->	29 ->	20 ->
33 ->	24 ->	24 ->	6 ->	37 ->
28 ->	35 ->	35 ->	37 ->	16 ->
22 ->	38 ->	30 ->	16 ->	29 ->
49 ->	44 ->	21 ->	23 ->	6 ->
24 ->	43 ->	33 ->	8 ->	19 ->
25 ->	32 ->	28 ->	34 ->	10 ->
19 ->	15 ->	22 ->	36 ->	49 ->
23 ->	50 ->	27 ->	45 ->	24 ->
34 ->	7 ->	39 ->	12 ->	35 ->
36 ->	4 ->	38 ->	11 ->	38 ->
45 ->	14 ->	9 ->	1 ->	44 ->
40 ->	2 ->	42 ->	5 ->	3 ->
12 ->	48 ->	41 ->	17 ->	43 ->
11 ->	26 ->	31 ->	47 ->	32 ->
1 ->	18 ->	44 ->	13 ->	15 ->
47 ->	46 ->	43 ->	40 ->	50 ->
13 ->	3 ->	32 ->	24 ->	7 ->
5 ->	9 ->	15 ->	30 ->	4 ->
17 ->	42 ->	3 ->	21 ->	18 ->
8 ->	41 ->	50 ->	39 ->	46 ->
20 ->	31 ->	7 ->	33 ->	14 ->
37 ->	30 ->	4 ->	28 ->	2 ->
16 ->	21 ->	18 ->	22 ->	48 ->

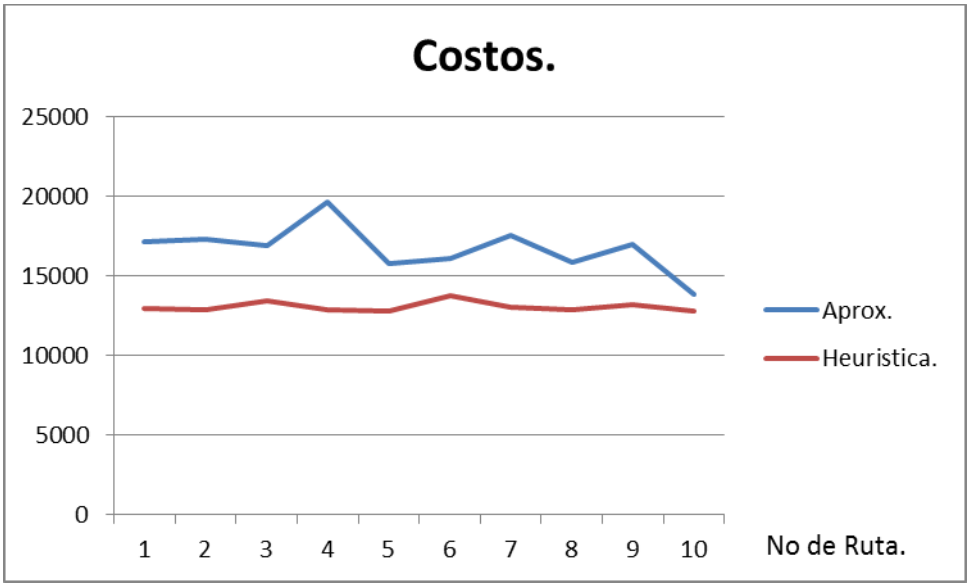
29 ->	33 ->	46 ->	27 ->	26 ->
6 ->	28 ->	14 ->	9 ->	9 ->
10 ->	22 ->	2 ->	42 ->	42 ->
50 ->	39 ->	48 ->	41 ->	41 ->
7 ->	27 ->	26 ->	31 ->	31 ->
4 ->	10 ->	20 ->	3 ->	30 ->
14 ->	40 ->	37 ->	43 ->	21 ->
2 ->	11 ->	16 ->	32 ->	27 ->
48 ->	1 ->	29 ->	15 ->	39 ->
26 ->	5 ->	6 ->	18 ->	33 ->
18 ->	17 ->	10 ->	46 ->	28 ->
46 ->	47 ->	19 ->	48 ->	22 ->
3 ->	13 ->	40 ->	26 ->	8 ->
32	12	1	2	23
Con un costo de: 17122.02	**Con un costo de:** 17300.92	**Con un costo de:** 16894.0	**Con un costo de:** 19638.22	**Con un costo de:** 15784.0
Tiempo de ejecución: 0.19444668293091052	**Tiempo de ejecución:** 0.09050163711778984	**Tiempo de ejecución:** 0.09045595488443164	**Tiempo de ejecución:** 0.08949303499926343	**Tiempo de ejecución:** 0.09038563504207128

La mejor ruta es la siguiente:	**La mejor ruta es la siguiente:**	**La mejor ruta es la siguiente:**	**La mejor ruta es la siguiente:**	**La mejor ruta es la siguiente:**
47 ->	12 ->	17 ->	19 ->	18 ->
1 ->	11 ->	5 ->	25 ->	4 ->
5 ->	1 ->	1 ->	20 ->	14 ->
17 ->	5 ->	11 ->	37 ->	2 ->
11 ->	17 ->	12 ->	16 ->	48 ->
12 ->	47 ->	45 ->	29 ->	26 ->
45 ->	13 ->	40 ->	6 ->	7 ->
40 ->	45 ->	36 ->	49 ->	50 ->
36 ->	36 ->	34 ->	24 ->	44 ->
34 ->	34 ->	23 ->	35 ->	43 ->
23 ->	23 ->	25 ->	30 ->	15 ->
8 ->	25 ->	10 ->	21 ->	32 ->
25 ->	49 ->	49 ->	39 ->	3 ->
10 ->	35 ->	24 ->	33 ->	38 ->
19 ->	30 ->	35 ->	28 ->	35 ->
49 ->	21 ->	38 ->	22 ->	30 ->
35 ->	33 ->	44 ->	27 ->	21 ->
30 ->	28 ->	50 ->	38 ->	27 ->
21 ->	22 ->	7 ->	44 ->	39 ->
33 ->	27 ->	4 ->	43 ->	33 ->
28 ->	39 ->	18 ->	32 ->	28 ->

22 ->	38 ->	46 ->	15 ->	22 ->
39 ->	9 ->	14 ->	3 ->	49 ->
27 ->	42 ->	2 ->	50 ->	24 ->
38 ->	41 ->	48 ->	7 ->	25 ->
9 ->	31 ->	26 ->	4 ->	10 ->
42 ->	44 ->	43 ->	14 ->	23 ->
41 ->	3 ->	32 ->	2 ->	34 ->
31 ->	43 ->	15 ->	48 ->	36 ->
44 ->	32 ->	3 ->	26 ->	45 ->
43 ->	15 ->	9 ->	18 ->	40 ->
32 ->	50 ->	42 ->	46 ->	12 ->
15 ->	7 ->	41 ->	9 ->	11 ->
3 ->	4 ->	31 ->	42 ->	1 ->
50 ->	18 ->	30 ->	41 ->	5 ->
7 ->	46 ->	21 ->	31 ->	17 ->
4 ->	14 ->	33 ->	10 ->	47 ->
14 ->	2 ->	28 ->	23 ->	13 ->
2 ->	48 ->	22 ->	34 ->	8 ->
48 ->	26 ->	27 ->	36 ->	20 ->
26 ->	24 ->	39 ->	45 ->	29 ->
18 ->	20 ->	19 ->	12 ->	6 ->
46 ->	29 ->	20 ->	11 ->	37 ->
24 ->	6 ->	29 ->	1 ->	16 ->
20 ->	37 ->	6 ->	47 ->	19 ->
29 ->	16 ->	37 ->	13 ->	9 ->
6 ->	10 ->	16 ->	5 ->	42 ->
37 ->	19 ->	8 ->	17 ->	41 ->
16 ->	8 ->	47 ->	40 ->	31 ->
13 ->	40 ->	13 ->	8 ->	46 ->
47	12	17	19	18
Con un costo de: 16069.22	**Con un costo de:** 17594.3	**Con un costo de:** 15870.0	**Con un costo de:** 17017.92	**Con un costo de:** 13878.68
Tiempo de ejecución: 0.10596019691608996	**Tiempo de ejecución:** 0.11274734491266739	**Tiempo de ejecución:** 0.12813147860044008	**Tiempo de ejecución:** 0.09137319253624167	**Tiempo de ejecución:** 0.11595947318632552

Comparación:





Conclusiones:

Los resultados muestran que en las pruebas realizadas, el que obtuvo mejores soluciones fue el algoritmo de la heurística y en menos tiempo, desde nuestro punto de vista y para fines más prácticos usaríamos el algoritmo de la heurística, además de que es lo mas cercano a la lógica humana, ya que por lo regular si estamos en lugar y tenemos que ir a otro lugares, generalmente nos dirigimos hacia el lugar que esté más cerca de donde estamos actualmente. Y no pensamos que el algoritmo de aproximación este equivocado o sea ineficiente, solo que por simplicidad usaríamos el de la heurística.