Reporte de algoritmo de Kruskal y PAV

José Manuel Tapia Avitia.
Matrícula: 1729372
jose.tapiaav@gmail.com
https://github.com/jose-tapia/1729372MC

3 de noviembre de 2017

El presente reporte tiene la finalidad de analizar y mostrar el algoritmo de Kruskal, su comportamiento en grafos y el como podríamos implementarlo en Python. Así como de exponer a grandes rasgos el problema del agente viajero, así cómo una aproximación no determinista, haciendo uso del algoritmo de Kruskal.

1. Problema del agente viajero

1.1. Definición

El **Problema del agente viajero** (PAV, por sus siglas) responde a la siguiente pregunta: "Dada una lista de ciudades y las distancias entre cada par de ellas,¿cuál es la ruta más corta posible que visite cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?".

Traduciendo esto a un problema de grafos: Dado un grafo completo, determinar el ciclo hamiltoneano con menor peso. Donde un ciclo hamiltoneano es un camino de un grafo, que visita todos los vértices del grafo una sola vez, además el último vértice visitado es adyacente al primero. El peso de un camino es la suma de las aristas que lo componen.

1.2. ¿Por qué es díficil el PAV?

El problema en si es $NP - Hard$, es decir, que por el momento no existe algoritmo con complejidad polinómica que resuelva dicho problema. Existe el problema del milenio ?‘$P = NP$?, el cual tiene una recompensa de un millón de dólares para aquel que lo resuelva.

Intentar todas las permutaciones nos permite resolver el problema en complejidad $\mathcal{O}(n!)$, con n el número de ciudades. Con programación dinámica, el algoritmo $Held - Karp$ resuelve el problema en $\mathcal{O}(n^2 2^n)$. La mejora de estos límites de tiempoes es dificil. Por ejemplo, no ha sido determinado si existe un algoritmo para el TSP que corra un tiempo de orden $\mathcal{O}(1{,}999^n)$.

1.3. ¿Qué es un algoritmo de aproximación

Un algoritmo de aproximación es un *algoritmo* usado para encontrar soluciones aproximadas a problemas de optimización, como es el caso del PAV. A diferencia de las herísticas, que usualmente encuentran soluciones razonablemente buenas en tiempos razonablemente rápidos, lo que se busca es encontrar soluciones que está demostrado son de calidad y cuyos tiempos de ejecución están acotadas por cotas conocidas. Idealmente, la aproximación mejora su calidad para factores constantes pequeños.

2. Algoritmo de Kruskal

2.1. Definiciones

Antes de profundizarnos en el algoritmo de Kruskal, veremos algunas definiciones.

- Árbol: Un árbol es un grafo conexo de n vértices, con $n-1$ aristas. Se cumple que al ser conexo, cualesquiera dos nodos del grafo están conectados, pero al ser un árbol, este camino es único.
- Árbol de expansión en un grafo: De un grafo G, se dice que G' es un árbol de expansión de G si: G' es un árbol, esta conformado por todos los vértices de G y además, las aristas de G' están en G.

2.2. Descripción del algoritmo

El algoritmo de Kruskal encuentra un árbol de expansión mínimo de un grafo G. Esto es, el árbol de expansión que al sumar todas las aristas, es lo mínima posible. El algoritmo permite encontrar dicho árbol en un tiempo bastante razonable, una complejidad de $\mathcal{O}(nlogn)$.

La idea base del algoritmo esta muy sencilla e intuitiva. He aqui el procedimiento:

- Sea G' un grafo vacío, en donde se irá construyendo el árbol de expansión mínima.
- Procesar las aristas del grafo G de acuerdo a sus pesos (En partícular, de menor a mayor).
- Para procesar una arista, se verifica que al agregar dicha arista al grafo G' no se forme algún ciclo, esto con ayuda de una estructura auxiliar que se verá más adelante. En caso de no formar ciclos, se añadirá al grafo G'.
- Se detiene el proceso hasta que se hayan procesado todas las aristas de G o la cantidad de aristas de G' sea igual a $n-1$, donde n es la cantidad de vértices de G.

2.3. Estructura auxiliar

2.3.1. Definición

La estructura auxiliar que nos permite verificar de manera eficiente si al agregar una arista al grafo G' se forma un ciclo es $Union\,Find$. También conocido como DSU (Disjoint set union, por sus siglas en inglés). Dicha estructura nos permite realizar operaciones como unir dos çonjuntos", preguntar si dos elementos pertenecen al mismo conjunto, preguntar el conjunto al que pertenece cierto elemento, entre otras operaciones.

La idea para encontrar si hay ciclo o no, es preguntarnos si los extremos que estamos por juntar ya se unieron anteriormente, en caso de que si, ambos elementos pertenecerán al mismo conjunto. Si no, ahora se unirán estos conjuntos.

2.3.2. Métodos

Resumiendo, las operaciones que realiza la estructura son las siguientes:

- Inicializar: En el caso en el que el grafo G' este vacio, todos los nodos se encuentran separados.
- get(n): Retorna el conjunto al que pertenece.
- union(n,m): Comprueba si n,m se encuentran en el mismo conjunto o no. En caso de que no, los uné y retorna verdadero, mientras que en el caso contrario, falso.

2.3.3. Implementación de DSU

```
class DSU(object):
        def __init__(self,nodos):
                self.padre={}
                for w in nodos:
                        self.padre[w]=w
        def get(self, n):
                x=n
                while x!=self.padre[x]:
                        x=self.padre[x]
                while x!=n:
                        n,self.padre[n]=self.padre[n],x
                return n
        def union(self, a, b):
                a,b=self.get(a),self.get(b)
                if a!=b:
                        self.padre[a]=b
                        return True
                else: return False
```

2.4. Implementación de Kruskal

Se realizó como método de la clase Grafo.

```
    def kruskal(self):
        aristas_ordenadasw=deepcopy(self.aristas)
        aristas_ordenadas=sorted(aristas_ordenadasw.keys(), key= lambda k:

        peso,arbol,n=0,Grafo(),len(self.vertices)
        dsu=DSU(self.vertices)

        while len(aristas_ordenadas)>0 and len(arbol.aristas)<2*(n-1):
            (x,y)=aristas_ordenadas.pop()
            if dsu.union(x,y):
                arbol.conecta(x,y,self.aristas[(x,y)])
                peso+=self.aristas[(x,y)]
        print("Arbol de expansion minima: ",peso)
        return arbol
```

3. Aproximación del PAV

3.1. Descripción del algoritmo

Retomando el tema del PAV y algoritmos de aproximación, daremos un algoritmo para dicho problema.

Del grafo G, sea G_k el árbol de expansión mínima que encuentra kruskal en G.

El algoritmo consiste en lo siguiente:

- De algún vértice inicial (Tomado de manera aleatoria), se realizá una búsqueda en profundidad. Dicho recorrido que realiza (Es decir, el que esta determinado por como fueron visitados cada vértice del grafo) propondremos un ciclo.
- De dicho recorrido, $[x_1, x_2, ..., x_n]$, nos tomaremos el ciclo dado por $(x_1 \to x_2), (x_2 \to x_3), \ldots, (x_{n-1} \to x_n), (x_n \to x_1)$. El cuál podría ser el ciclo buscado, u aproximado.
- Repetir el proceso hasta tener la probabilidad de haber tenido un ciclo aproximado.

3.2. Implementación de algoritmo de aproximación

Dicho algoritmo, se implementó como sigue:

```
print("ALGORITMO DE APROXIMACION PARA PAV\n")
Gk=G.kruskal()
print(Gk)#Imprime el arbol de expansion minima
tim=time.clock()
mejor=-1
camino=[]
for repeat in range(5):
    inicial=random.choice(list(Gk.vertices))
    dfs=Gk.DFS(inicial)
    peso=0
    for i in range(len(dfs)-1):
        peso+=G.aristas[(dfs[i],dfs[i+1])]
    peso+=G.aristas[(dfs[-1],dfs[0])]

    print("Camino iniciando en ",inicial)
    for i in range(len(dfs)-1):
        print("De ",dfs[i],"\ta",dfs[i+1],"\tla distancia es ",G.aristas[(d
    print("De ",dfs[-1],"\ta ",dfs[0],"\tla distancia es ",G.aristas[(dfs[-

    print("Suma del camino: ",peso,"\n")
    if mejor==-1 or mejor>peso:
        mejor=peso
        camino=dfs
print("El mejor camino fue el 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")
```

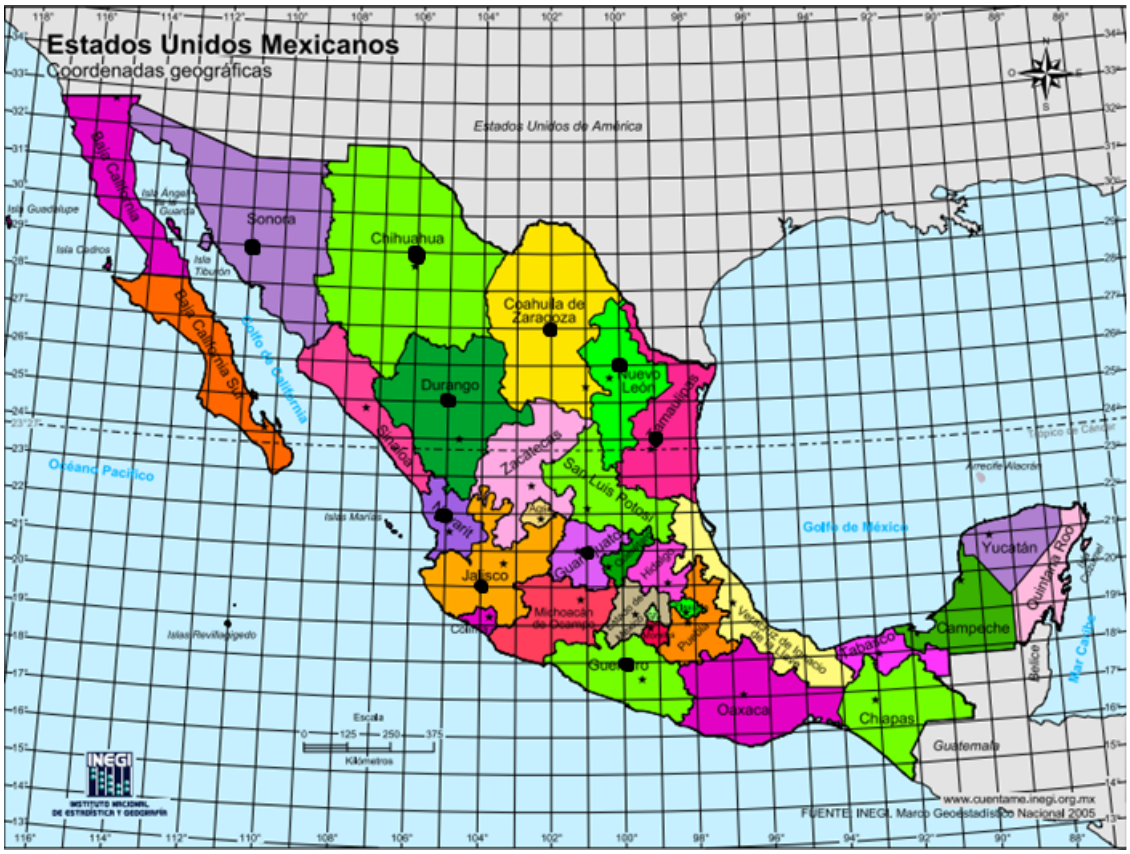
3.3. Caso de ejemplo

Aplicando dicho algoritmo en un ejemplo realista", tendremos resultados aproximados, en donde podremos ver de manera aplicada dicho algoritmo.

3.3.1. Descripción del caso

El caso propuesto es recorrer 10 estados de la Republica Mexicana, estos son Nayarit, Nuevo León, Durango, Chihuahua, Coahuila, Sonora, Tamaulipas, Jalisco, Guanajuato y Guerrero. Empezando en alguno de ellos, realizando un ciclo.

De acuerdo a la siguiente imagen, podemos determinar una aproximación de las distancias entre los puntos de interés que tenemos para cada estado. Para simplificar el proceso, se decidió que Nayarit correspondería al punto (0,0), sin perdida de generalidad.



En dicho mapa, se encuentran marcados con puntos negros los lugares de interés. A continuación se especifican sus coordenadas:

- Nayarit (0,0)
- Durango (0,3)
- Chihuahua (-1,7)
- Sonora (-6,7)
- Coahuila (3,5)
- Nuevo León (5,4)
- Tamaulipas (6,2)

- Jalisco (1,-2)
- Guanajuato (4,-1)
- Guerrero (-4,5)

Para que el programa lo interprete, se realizó el siguiente código:

```
#Crearemos el grafo, representando las coordenadas
mapa=[]
mapa.append(("Nayarit",(0,0)))
mapa.append(("Durango",(0,3)))
mapa.append(("Chihuahua",(-1,7)))
mapa.append(("Sonora",(-6,7)))
mapa.append(("Coahuila",(3,5)))
mapa.append(("Nuevo Leon",(5,4)))
mapa.append(("Tamaulipas",(6,2)))
mapa.append(("Jalisco",(1,-2)))
mapa.append(("Guanajuato",(4,-1)))
mapa.append(("Guerrero",(-4,5)))

G=Grafo()
for (u,p) in mapa:#Nodo u, Punto p
    for (v,q) in mapa:#Nodo v, Punto q
        if u==v:
            continue
        G.conecta(u,v,distancia(p,q))

print(G)
```

Además, para poder calcular la distancia entre cada estado utilizamos el siguiente código, que calcula la distancia entre dos puntos del plano cartesiano.

```
def distancia(p,q):
    (x,y),(a,b)=p,q
    return math.sqrt((x-a)**2+(y-b)**2)
```

Siendo el grafo final el siguiente:

```
Vertices:
 Chihuahua  Tamaulipas  Durango  Sonora  Guerrero  Guanajuato  Nayarit
Coahuila  Jalisco  Nuevo Leon
Aristas:
 ('Nayarit', 'Durango'):        3.0
 ('Nayarit', 'Chihuahua'):      7.0710678118654755
 ('Nayarit', 'Sonora'):         9.219544457292887
 ('Nayarit', 'Coahuila'):       5.830951894845301
 ('Nayarit', 'Nuevo Leon'):     6.4031242374328485
 ('Nayarit', 'Tamaulipas'):     6.324555320336759
 ('Nayarit', 'Jalisco'):        2.23606797749979
 ('Nayarit', 'Guanajuato'):     4.123105625617661
 ('Nayarit', 'Guerrero'):       6.4031242374328485
 ('Durango', 'Chihuahua'):      4.123105625617661
 ('Durango', 'Sonora'):         7.211102550927978
```

```
('Durango', 'Coahuila'):        3.605551275463989
('Durango', 'Nuevo Leon'):      5.0990195135927845
('Durango', 'Tamaulipas'):      6.082762530298219
('Durango', 'Jalisco'):         5.0990195135927845
('Durango', 'Guanajuato'):      5.656854249492381
('Durango', 'Guerrero'):        4.47213595499958
('Chihuahua', 'Sonora'):        5.0
('Chihuahua', 'Coahuila'):      4.47213595499958
('Chihuahua', 'Nuevo Leon'):    6.708203932499369
('Chihuahua', 'Tamaulipas'):    8.602325267042627
('Chihuahua', 'Jalisco'):       9.219544457292887
('Chihuahua', 'Guanajuato'):    9.433981132056603
('Chihuahua', 'Guerrero'):      3.605551275463989
('Sonora', 'Coahuila'):         9.219544457292887
('Sonora', 'Nuevo Leon'):       11.40175425099138
('Sonora', 'Tamaulipas'):       13.0
('Sonora', 'Jalisco'):          11.40175425099138
('Sonora', 'Guanajuato'):       12.806248474865697
('Sonora', 'Guerrero'):         2.8284271247461903
('Coahuila', 'Nuevo Leon'):     2.23606797749979
('Coahuila', 'Tamaulipas'):     4.242640687119285
('Coahuila', 'Jalisco'):        7.280109889280518
('Coahuila', 'Guanajuato'):     6.082762530298219
('Coahuila', 'Guerrero'):       7.0
('Nuevo Leon', 'Tamaulipas'):   2.23606797749979
('Nuevo Leon', 'Jalisco'):      7.211102550927978
('Nuevo Leon', 'Guanajuato'):   5.0990195135927845
('Nuevo Leon', 'Guerrero'):     9.055385138137417
('Tamaulipas', 'Jalisco'):      6.4031242374328485
('Tamaulipas', 'Guanajuato'):   3.605551275463989
('Tamaulipas', 'Guerrero'):     10.44030650891055
('Jalisco', 'Guanajuato'):      3.1622776601683795
('Jalisco', 'Guerrero'):        8.602325267042627
('Guanajuato', 'Guerrero'):     10.0
```

3.3.2. Resultados del algoritmo

Al realizar el algoritmo descrito anteriormente, nos arrojó los siguientes resultados:

```
Arbol de expansion minima:  27.033116893959576
Vertices:
 Coahuila  Durango  Chihuahua  Tamaulipas  Guerrero  Guanajuato  Jalisco
Sonora  Nayarit  Nuevo Leon
Aristas:
 ('Tamaulipas', 'Nuevo Leon'):  2.23606797749979
 ('Nuevo Leon', 'Coahuila'):    2.23606797749979
 ('Jalisco', 'Nayarit'):        2.23606797749979
 ('Guerrero', 'Sonora'):        2.8284271247461903
 ('Durango', 'Nayarit'):        3.0
 ('Guanajuato', 'Jalisco'):     3.1622776601683795
 ('Guanajuato', 'Tamaulipas'):  3.605551275463989
 ('Guerrero', 'Chihuahua'):     3.605551275463989
 ('Chihuahua', 'Durango'):      4.123105625617661

Camino iniciando en  Jalisco
De  Jalisco     a Guanajuato    la distancia es  3.1622776601683795
De  Guanajuato  a Tamaulipas    la distancia es  3.605551275463989
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a Nayarit       la distancia es  5.830951894845301
De  Nayarit     a Durango       la distancia es  3.0
De  Durango     a Chihuahua     la distancia es  4.123105625617661
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a  Jalisco      la distancia es  11.40175425099138
Suma del camino:  42.02975506229647

Camino iniciando en  Nayarit
De  Nayarit     a Durango       la distancia es  3.0
De  Durango     a Chihuahua     la distancia es  4.123105625617661
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a Jalisco       la distancia es  11.40175425099138
De  Jalisco     a Guanajuato    la distancia es  3.1622776601683795
De  Guanajuato  a Tamaulipas    la distancia es  3.605551275463989
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a  Nayarit      la distancia es  5.830951894845301
Suma del camino:  42.02975506229646

Camino iniciando en  Guanajuato
De  Guanajuato  a Jalisco       la distancia es  3.1622776601683795
De  Jalisco     a Nayarit       la distancia es  2.23606797749979
De  Nayarit     a Durango       la distancia es  3.0
De  Durango     a Chihuahua     la distancia es  4.123105625617661
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
```

```
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a Tamaulipas    la distancia es  13.0
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a  Guanajuato   la distancia es  6.082762530298219
Suma del camino:  42.51032814879381

Camino iniciando en  Coahuila
De  Coahuila    a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Tamaulipas    la distancia es  2.23606797749979
De  Tamaulipas  a Guanajuato    la distancia es  3.605551275463989
De  Guanajuato  a Jalisco       la distancia es  3.1622776601683795
De  Jalisco     a Nayarit       la distancia es  2.23606797749979
De  Nayarit     a Durango       la distancia es  3.0
De  Durango     a Chihuahua     la distancia es  4.123105625617661
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a  Coahuila     la distancia es  9.219544457292887
Suma del camino:  36.25266135125247

Camino iniciando en  Chihuahua
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a Durango       la distancia es  7.211102550927978
De  Durango     a Nayarit       la distancia es  3.0
De  Nayarit     a Jalisco       la distancia es  2.23606797749979
De  Jalisco     a Guanajuato    la distancia es  3.1622776601683795
De  Guanajuato  a Tamaulipas    la distancia es  3.605551275463989
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a  Chihuahua    la distancia es  4.47213595499958
Suma del camino:  34.593249774269474

El mejor camino fue el siguiente:
Chihuahua -> Guerrero -> Sonora -> Durango -> Nayarit -> Jalisco
-> Guanajuato -> Tamaulipas -> Nuevo Leon -> Coahuila -> Chihuahua

Con un costo de  34.593249774269474
Tiempo de ejecucion:  0.14658418977896664
```

Al principio, se muestra el árbol de expansión mínima de nuestro grafo G.

Resultando el mejor ciclo, el conformado por Chihuahua, Guerrero, Sonora, Durango, Nayarit, Jalisco, Guanajuato, Tamaulipas, Nuevo León y Coahuila en ese orden, con un costo de 34.593249774269474

4. Heuristica del PAV (Vecino más cercano)

4.1. Descripción del algoritmo

De manera intuitiva, podemos pensar que si nos movemos a el lugar más cercano llegaremos a nuestro destino antes. El algoritmo se comporta de manera similar a dicho razonamiento. El procedimiento es el que sigue:

- Iniciamos en algún vértice (Elegido de manera aleatoria).
- Nos movemos a el vértice más cercano que no haya sido visitado.
- Repetimos el proceso hasta haber visitado todos los nodos.

Lo cuál no siempre nos genera la respuesta más óptima.

4.2. Implementación del algoritmo

La implementación se realizó como un método de la clase Grafo:

```
def heuristica(self,ini):
    vis=[ini]
    act=ini
    for k in range(len(self.vertices)-1):
        sig=0
        peso=1000000000
        for w in self.vecinos[act]:
            if w not in vis and self.aristas[(w,act)]<peso:
                peso=self.aristas[(w,act)]
                sig=w
        act=sig
        vis.append(act)
    return vis
```

4.3. Resultado del algoritmo

```
Camino iniciando en  Coahuila
De  Coahuila    a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Tamaulipas    la distancia es  2.23606797749979
De  Tamaulipas  a Guanajuato    la distancia es  3.605551275463989
De  Guanajuato  a Jalisco       la distancia es  3.1622776601683795
De  Jalisco     a Nayarit       la distancia es  2.23606797749979
De  Nayarit     a Durango       la distancia es  3.0
De  Durango     a Chihuahua     la distancia es  4.123105625617661
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a  Coahuila     la distancia es  9.219544457292887
Suma del camino:  36.25266135125247

Camino iniciando en  Guanajuato
De  Guanajuato  a Jalisco       la distancia es  3.1622776601683795
```

```
De  Jalisco     a Nayarit       la distancia es  2.23606797749979
De  Nayarit     a Durango       la distancia es  3.0
De  Durango     a Coahuila      la distancia es  3.605551275463989
De  Coahuila    a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Tamaulipas    la distancia es  2.23606797749979
De  Tamaulipas  a Chihuahua     la distancia es  8.602325267042627
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a  Guanajuato   la distancia es  12.806248474865697
Suma del camino:  44.318585010250246

Camino iniciando en  Nayarit
De  Nayarit     a Jalisco       la distancia es  2.23606797749979
De  Jalisco     a Guanajuato    la distancia es  3.1622776601683795
De  Guanajuato  a Tamaulipas    la distancia es  3.605551275463989
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a Durango       la distancia es  3.605551275463989
De  Durango     a Chihuahua     la distancia es  4.123105625617661
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a  Nayarit      la distancia es  9.219544457292887
Suma del camino:  36.85821262671645

Camino iniciando en  Guerrero
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a Chihuahua     la distancia es  5.0
De  Chihuahua   a Durango       la distancia es  4.123105625617661
De  Durango     a Nayarit       la distancia es  3.0
De  Nayarit     a Jalisco       la distancia es  2.23606797749979
De  Jalisco     a Guanajuato    la distancia es  3.1622776601683795
De  Guanajuato  a Tamaulipas    la distancia es  3.605551275463989
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a  Guerrero     la distancia es  7.0
Suma del camino:  35.427565618495585

Camino iniciando en  Tamaulipas
De  Tamaulipas  a Nuevo Leon    la distancia es  2.23606797749979
De  Nuevo Leon  a Coahuila      la distancia es  2.23606797749979
De  Coahuila    a Durango       la distancia es  3.605551275463989
De  Durango     a Nayarit       la distancia es  3.0
De  Nayarit     a Jalisco       la distancia es  2.23606797749979
De  Jalisco     a Guanajuato    la distancia es  3.1622776601683795
De  Guanajuato  a Chihuahua     la distancia es  9.433981132056603
De  Chihuahua   a Guerrero      la distancia es  3.605551275463989
De  Guerrero    a Sonora        la distancia es  2.8284271247461903
De  Sonora      a  Tamaulipas   la distancia es  13.0
Suma del camino:  45.34399240039852
```

```
El mejor camino fue el siguiente:
Guerrero ->
Sonora ->
Chihuahua ->
Durango ->
Nayarit ->
Jalisco ->
Guanajuato ->
Tamaulipas ->
Nuevo Leon ->
Coahuila ->
Guerrero

Con un costo de  35.427565618495585
Tiempo de ejecucion:  0.14108841921253928
```

En donde nos genera una respuesta muy buena en un tiempo razonable, lo cual es un buen indicio.

5. Solución exacta

Para resolver el PAV de manera exacta se puede realizar de diversas formas, al tener nuestro grafo de 10 vértices, debería ser suficiente el algoritmo de fuerza bruta $\mathcal{O}(n!)$, probar todas los ciclos posibles.

5.1. Descripción del algoritmo

Básicamente es probar todos los ciclos, esto se puede realizar de la siguiente manera:

- Realizar un conjunto de todas las permutaciones de los números del 1 al n, donde n es la cantidad de vértices de G.
- Realizar cada permutación, obteniendo su suma e ir actualizando nuestra mejor respuesta

Aunque, de manera alternativa, se puede realizar lo siguiente:

- Realizar una DFS, solo que de manera recursiva y sin marcar "visitados"(más si el cuidar la condición de no pasar por dos vértices en el ciclo).
- Realizar todos los posibles caminos e ir actualizando nuestra respuesta global.

5.2. Implementación del algoritmo

Para generar las permutaciones de un arreglo, realizamos el siguiente método:

```
def permutaciones(arr):
    if len(arr)==0: return [[]]
    perm=[]
    for i in range(len(arr)):
        wot=permutaciones(arr[:i]+arr[i+1:])
        for w in wot:
            perm.append([arr[i]]+w)
    return perm
```

Se realizó como un método de la clase Grafo:

```
def PAV(self):
    perm=permutaciones(list(self.vertices))
    mejor=-1
    camino=[]
    for w in perm:
        peso=0
        for i in range(len(w)-1):
            peso+=self.aristas[(w[i],w[i+1])]
        peso+=self.aristas[(w[-1],w[0])]
        if peso<mejor or mejor==-1:
            mejor=peso
            camino=w
    return camino
```

5.3. Resultado del algoritmo

El resultado fue el siguiente, consiguiendo un mejor camino a los anteriores obtenidos:

```
El mejor camino fue el siguiente:
Chihuahua ->
Sonora ->
Guerrero ->
Durango ->
Nayarit ->
Jalisco ->
Guanajuato ->
Tamaulipas ->
Nuevo Leon ->
Coahuila ->
Chihuahua

Con un costo de  33.24873190287708
Tiempo de ejecucion: 107.94603439274485
```

Por lo tanto, uno de los ciclos con la menor distancia posible es la siguiente: Chihuahua, Sonora, Guerrero, Durango, Nayarit, Jalisco, Guanajuato, Tamaulipas, Nuevo Leon, Coahuila, en ese orden. Obteniendo un resultado mejor, 33.24873190287708.

6. Conclusión

Con los resultados de la solución exacta, podemos darnos cuenta de que algoritmo fue el que se acerco más fue el de aproximación. De hecho, podemos apreciar el algoritmo de aproximación regularmente se acercaba, mientras que la heuristica consiguió el mismo resultado, más sin embargo, en la mayoría de los casos obtenía un resultado mayor respecto a los de la aproximación.

Al ser algoritmos no-deterministas (esto es, en el proceso del algoritmo se hace una elección aleatoria), nos dió un resultado satisfactoreo de acuerdo al tiempo en el que estuvieron ejecutandose.

Mientras que el algoritmo de solución exacta se tardó casi dos minutos, se consiguió una mejoría de 1.34451787139.

Para medir el tiempo que tardá en ejecutar un código, utilizaremos la siguiente función:

```
import time
tim=time.clock()
#Operaciones
print(time.clock()-tim)
```

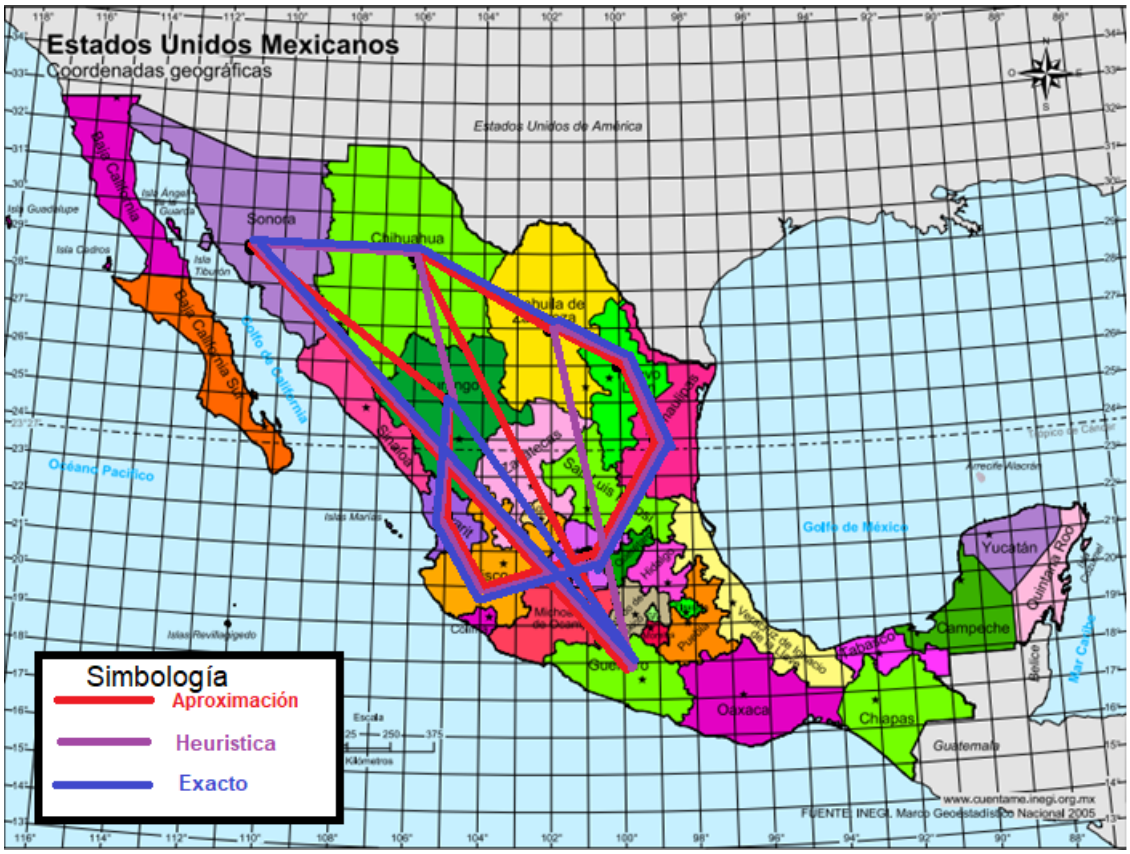
En la siguiente tabla podemos comparar los tiempos de ejecución de dichos algoritmos corriendo sobre el mismo grafo.

Problema del agente viajero			
Algoritmo	Costo resultante	Camino obtenido	Tiempo de ejecución (en segundos)
Aproximación	34.593249774269474	Chihuahua, Guerrero, Sonora, Durango, Nayarit, Jalisco, Guanajuato, Tamaulipas, Nuevo León, Coahuila.	0.14658418977896664
Heuristica	35.427565618495585	Guerrero, Sonora, Chihuahua, Durango, Nayarit, Jalisco, Guanajuato, Tamaulipas, Nuevo Leon, Coahuila.	0.14108841921253928
Exacta	33.24873190287708	Chihuahua, Sonora, Guerrero, Durango, Nayarit, Jalisco, Guanajuato, Tamaulipas, Nuevo Leon, Coahuila.	107.94603439274485

De la tabla podemos concluir que los algoritmos, tanto como el de aproximación como el de la heuristica, se aproximan al resultado óptimo, con una diferencia de 1.34451787139 y 2.17883371562, respectivamente. Lo cual equivale a que el margen de error aproximado de los algoritmos resulto ser de 4 % y 7 % respectivamente.

Observando el tiempo de ejecución, el obtener la solución exacta con solamente 10 vértices se tardó aproximadamente un minuto y medio. Mientras que en los otros algoritmos se tardaron menos de 0.15 segundos. Lo cuál nos indica que aún sin tener la solución más óptima, es posible obtener un resultado bueno mil veces mas rápido (Aproximado, realmente es cómo 765 veces).

Podemos ver de manera gráfica los caminos obtenidos en la siguiente imagen:



Observemos que podemos empezar desde cualquier camino, ya que de igual forma lo que buscamos es un ciclo.

Además, podemos ver lo extravagante que resulta ser la respuesta exacta, ya que se ve de forma muy rebuscada, díficil de encontrar.

7. Detalles de implementación

Realizé varias ejecuciones del programa, en donde me arrojaba distintos resultados en los algoritmos de aproximación y Heuristica, solían ambos obtener el resultado de 34.593249774269474. Regularmente el de aproximación lo obtenía, raramente el de heuristica.

Algo interesante es que ambos algoritmos tienen complejidad $\mathcal{O}(kn)$ donde n es la cantidad de vértices y k la cantidad de iteraciones que se desea realizar el algoritmo (Excepto el algoritmo de aproximación, el cuál necesita como precalculo el caluclar el árbol de expansión mínima, que al realizarlo con el algoritmo de kruskal se obteien complejidad total de $\mathcal{O}(n(logn + k))$). Esto nos lleva a que si deseamos hacerlo con $k = 5$, como se realizó, tiene complejidad los algoritmos de $\mathcal{O}(n)$, obteniendo un margen de error menor a $7\,\%$, mientras que el algoritmo de complejidad $\mathcal{O}(n!)$ nos da la solución exacta, pero la cantidad de tiempo que se invierte es considerablemente mayor.

Todo esto para concluir en que bajo a las necesidades que tengamos, son los algoritmos que podemos usar. Si permite tener un margen de error, se puede usar tanto el algoritmo de aproximación como el de la heuristica. Mientras que si deseamos ser más precisos, debemos de introducirnos a algoritmos más complejos o tardados.