Documentación Técnica Pueblos Mágicos

Padilla Robles Artemio Santiago
05 de Febrero del 2021
Matemáticas Discretas y Algoritmos - IIMAS
Universidad Nacional Autónoma de México

En este archivo se presenta la documentación técnica de la aplicación para encontrar la ruta óptima para viajar a un conjunto de Pueblos Mágicos de interés del usuario, iniciando y terminando el viaje desde una ubicación arbitraria. En estas páginas se describe el proceso de obtención y procesamiento de los datos necesarios para implementar esta aplicación, el planteamiento matemático del problema, los algoritmos propuestos para encontrar la ruta óptima o su aproximación según sea el caso, de estos algoritmos se analiza su correctitud y sus complejidades en tiempo y en espacio. Finalmente se presenta la aplicación de estos algoritmos para solucionar el problema para 4 ejemplos de diferentes posibles entradas del usuario. Finalmente se discute como con esta aplicación se podrían proponer viajes turísticos que fomenten el desarrollo económico de los Pueblos Mágicos.

1. PLANTEAMIENTO DEL PROBLEMA A RESOLVER

Obtener la manera óptima de recorrer Pueblos Mágicos puede resultar en una propuesta atractiva para realizar turismo por diferentes partes de México, sin embargo, este problema forma parte del tipo de problemas del viajante (en inglés Travelling Salesman Problem o TSP). Obtener la manera de recorrer un ciclo de longitud mínima que pase por todas las ciudades que uno desea visitar es un problema que ha surgido desde los inicios del comercio y las rutas mercantes, y sin embargo, es uno de los problemas difíciles en las ciencias de la computación (NP-duro), por lo cual no se conocen maneras de solucionarlo en tiempo polinomial [1].

El problema de encontrar el ciclo óptimo puede resolverse mediante búsqueda exhaustiva, sin embargo, resolverlo por este método para más y más destinos rápidamente se vuelve inviable puesto que el número de combinaciones incrementa de manera exponencial, por lo que si quisiéramos conocer la ruta óptima para visitar los 132 Pueblos Mágicos necesitaríamos calcular 132! posibles rutas, esto equivale a explorar alrededor de 1×10^{224} opciones. Suponiendo que una computadora promedio actual puede calcular alrededor de $1,000,000$ de operaciones por segundo, esto nos da un tiempo de cómputo de alrededor de 3.54^{210} años, lo cual es alrededor de 200 ordenes de magnitud mayor que la vida del universo [2].

En este trabajo se busca solucionar el problema de recorrer el ciclo de longitud de mínima, partiendo de una la ubicación determinada por el usuario, pasando por todos los Pueblos Mágicos que este desea conocer y regresando al punto de partida.

Los Pueblos Mágicos son municipios de México que cuentan con una distinción particular por su belleza natural y su riqueza cultural y son puntos de referencia para los turistas que desean conocer el país, por esta razón, el conocer una ruta óptima puede fomentar viajes turísticos meramente centrados en conocer estos Pueblos Mágicos y contribuir con su desarrollo socio-económico a partir del fomento al turismo [3]. Por estas razones en este proyecto se propone una aplicación para resolver el problema de encontrar una ruta óptima para visitar Pueblos Mágicos de interés para el usuario partiendo y volviendo desde la ubicación que a este convenga. Debido a la imposibilidad de resolver este problema de manera exacta para gran cantidad de paradas, se proponen tanto algoritmos exactos para el caso donde se desea visitar 10 o menos Pueblos Mágicos, como aproximaciones a la ruta óptima cuando se trata con más.

2. BASE DE DATOS UTILIZADA

Para solucionar este problema se partió de una base de datos con todos los estados y municipios de todos Pueblos Mágicos de México hasta el año 2017 [4], esta base de datos está publicada por la Secretaría de Turismo en el Portal de Datos Abiertos del Gobierno de México y contiene información relativa a 112 localidades, sin embargo, entre ese año y la fecha de la publicación de este trabajo ya existen 132 localidades con el nombramiento de Pueblo Mágico, por lo que para completar este listado se cotejó la base de datos con una publicación de diciembre del 2020 [5] del Gobierno de México, donde se hace mención de todos los Pueblos Mágicos hasta esa fecha. De esta manera se obtuvo el estado y municipio de todos los Pueblos Mágicos de México.

Este listado se corrigió en algunos errores ortográficos y para facilidad de interpretación de los datos por la compu-

tadora se eliminaron acentos. A partir de la lista de los Pueblos Mágicos generaron dos bases de datos, la principal corresponde a la matrices de distancias entre Pueblos Mágicos, y adicionalmente se creó una secundaria que es una base de datos que contiene las coordenadas de todos los Pueblos Mágicos y se puede utilizar para graficar rutas entre ellos en caso de ser deseado.

Tanto para obtener las distancias entre Pueblos Mágicos como sus coordenadas se completaron las bases de datos usadas con datos del Google Maps.

A continuación se explican estas bases de datos y su modelo matemático. Un mapa con las ubicaciones de la totalidad de los Pueblos Mágicos se muestra en la figura 1.

Fig. 1. Ubicaciones de los 132 Pueblos Mágicos de México en Febrero de 2021

3. FORMA DE MODELAR EL PROBLEMA MATEMÁTICAMENTE Y JUSTIFICACIÓN DE SUPOSICIONES

Dado que de cualquier Pueblo Mágico podemos ir a cualquier otro Pueblo Mágico, podemos modelar nuestro problema como una gráfica completa, donde cada Pueblo Mágico es un vértice y las aristas que unen los vértices tienen como peso la distancia entre Pueblos Mágicos. En este caso se decidió tomar distancias por ser una métrica que varía menos que el tiempo de viaje. Con este modelo de gráfica completa se puede generar una matriz de distancias la cual consiste en una matriz de 132 columnas y 132 renglones donde la entrada i, j indica la distancia de moverse del Pueblo Mágico i al Pueblo Mágico j. Para obtener las distancias entre Pueblos Mágicos se utilizó el API de Google Maps y se consideraron distancias en kilómetros de recorrido en automóvil.

En este paso se utilizaron varias suposiciones, la primera es que el usuario desea transportarse en automóvil. La segunda es que la distancia entre Pueblos Mágicos de ida es la misma que la de regreso. Es decir, sea $d(i, j)$ la distancia entre el Pueblo Mágico i al Pueblo Mágico j, entonces sucede que $d(i, j) = d(j, i)$. Esta suposición puede no ser estrictamente

cierta puesto que para viajar desde lugar hasta otro pueden existir pequeñas variaciones por los sentidos de las calles o porque las carreteras de un sentido toman un camino ligeramente diferente de un sentido que de otro, pero generalmente estas diferencias suelen ser despreciables. Además que al hacer esta suposición reducimos a la mitad la cantidad de consultas que tenemos que hacer a la API de Google Maps para conocer la distancia entre Pueblos Mágicos. Dado que estamos comparando 132 Pueblos Mágicos con 132 Pueblos Mágicos la matriz de distancia tienen $132 \times 132 = 17,424$ entradas.

Esta matriz de distancias representa la gráfica completa y será usada tanto para dar la solución exacta por búsqueda exhaustiva como la aproximación a la solución en casos con más de 10 Pueblos Mágicos.

Con esta matriz de distancias que representa la gráfica completa se puede obtener, dados los requerimientos del usuario, una subgráfica completa, con los Pueblos Mágicos que el usuario quiera visitar. De esta forma, de acuerdo a las necesidades el usuario se obtiene una matriz de distancias entre cada Pueblo Mágico de interés, y al momento se calcula la distancia entre la ubicación ingresada por el usuario como punto de partida y retorno y los Pueblos Mágicos que este quiere visitar, para de esta forma obtener una nueva matriz de distancias que representa una gráfica completa de los Pueblos Mágicos solicitados junto a la ubicación del usuario.

Para la solución aproximada de este problema se supone que la matriz de distancias obedece la desigualdad del triangulo, es decir que la distancia del punto a, al punto b y después al punto c es menor o igual a la distancia del punto a al punto c, i.e. $d(a,b) + d(b,c) \leq d(a,c)$. Esta es una suposición razonable, dado que Google Maps da como distancia entre puntos la menor distancia posible, por lo que en el peor de los casos si vamos de a a c pasaremos de nuevo por el punto b, en cuyo caso se da la igualdad $d(a,b) + d(b,c) = d(a,c)$.

Bajo estos supuestos el problema de la ruta para visitar los Pueblos Mágicos partiendo de un punto y regresando al mismo se convierte en un problema de encontrar el ciclo hamiltoneano de peso mínimo sobre la gráfica representada por la matriz de distancias generada a partir de la ubicación que introduce usuario y los pueblos que quiere visitar.

Ahora tenemos un modelo matemático del problema a resolver expresado con teoría de gráficas, el cual se resolverá para el caso exacto por búsqueda exhaustiva y para el caso aproximado por la llamada 2-aproximación al problema del viajero, la cual nos da una solución máximo dos veces peor que la ruta óptima, sin embargo, esta solución suele ser 15 % a 20 % peor que la solución óptima [6].

Más adelante discutiremos los algoritmos para solucionar el problema tanto de manera exacta como por aproximación, por ahora, presentamos el enunciado algorítmico del problema.

4. ENUNCIADO ALGORÍTMICO GENERAL DEL PROBLEMA A RESOLVER

Con lo mencionado anteriormente podemos formular el problema a resolver algorítmicamente de la siguiente manera:

```
pueblos_magicos = Entrada de Usuario

origen = Entrada de usuario

Seleccionar la submatriz de distancias
entre los Pueblos Mágicos introducidos
por el usuario

Calcular las distancias del punto de
origen a todos los Pueblos Mágicos
seleccionados

Hacer una matriz de distancia que
contenga tanto las distancias entre
Pueblos Mágicos como la distancia
entre el punto de origen y los
Pueblos Mágicos

Si el tamaño de la matriz de distancias
a resolver corresponde a un número de
ubicaciones menor o igual que 10:

     Resolver Problema exacto

Si el tamaño es mayor:

     Resolver Solución Aproximada

Imprimir la ruta recomendada y la
distancia total para completarla
```

Procedamos ahora a analizar los algoritmos para obtener tanto la solución exacta como la solución aproximada.

5. PROPUESTA DE SOLUCIÓN MEDIANTE ALGORITMOS COMBINATORIOS

Analicemos por separado los algoritmos propuestos para la búsqueda exhaustiva y la aproximación a la solución.

1. Búsqueda Exhaustiva

Analicemos primero el algoritmo para encontrar la solución exacta, este algoritmo se basa en encontrar dicha solución mediante la exploración exhaustiva del espacio de estados de las posibles soluciones. Este algoritmo se aplica a las instancias del problema donde la ruta tiene 10 o menos paradas en su recorrido.

Para esto este algoritmo explora con backtrack todos los posibles caminos entre paradas (nodos en la gráfica) y va calculando sumando las distancias $d(i, j)$ de cada opción que toma. Para no repetir paradas lleva un registro de que nodos ha visitado, además de llevar registro de cuantas paradas ha realizado, de tal forma que en el momento en que ha visitado todos los nodos verifica si el camino encontrado es el de peso mínimo hasta el momento, de ser así lo guarda. En el algoritmo el origen corresponde al nodo 0.

Este algoritmo puede escribirse como:

```
costo_minimo = algún número muy grande
count = 1
vis = Lista en 0s para marcar nodos
                          visitados
actual = 0

TSP(vis, actual, count, km, path):

  si count == numero de paradas:
   si km + d(actual, origen) < costominimo:
     costominimo = km + d(actual, origen)
     mejor_camino = path
     agregar al mejor_camino el regreso
                            al origen
     regresar costo_minimo, mejor_camino

  En otro caso:
    para cada nodo en la lista de paradas:
      Si no hemos visitado el nodo:
        agregar el nodo a path
        marcamos el nodo como visitado
        TSP(vis, nodo, count + 1,
                 km + d(actual,nodo), path)
        quitamos el nodo de path
        quitamos la marca de visitado
                              del nodo
    regresar costo_minimo, mejor_camino
```

Procedamos ahora a analizar el algoritmo propuesto para encontrar una aproximación a la solución óptima para rutas con más de 10 Pueblos Mágicos.

2. Aproximación a la solución

El algoritmo descrito se basa en la aproximación mencionada en el libro de Cormen [1]. Para la aproximación a la solución realiza la creación de un árbol generador de peso mínimo T, el cuál tiene una distancia/costo en las aristas que lo forman dado por $c(T)$. Sea $c(H^*)$ la distancia total del ciclo de la solución óptima, notemos que $c(T) \leq c(H^*)$ debido a que un árbol generador tiene $n-1$ aristas, mientras el ciclo hamiltoneano tiene n y ambos tocan todos los vértices. Notemos también que si hacemos una caminata completa W en T, de tal forma que recorremos T tanto para llegar a los nodos hoja como de regreso al nodo raíz el costo de esta caminata completa es $c(W) = 2c(T)$ debido a que tomamos cada arista tanto de ida como de vuelta.

Combinando $c(W) = 2c(T)$ con $c(T) \leq c(H^*)$, notemos que $c(W) \leq 2c(H^*)$. Ahora utilizamos la desigualdad del triangulo asumida para las distancias entre aristas para este problema, y notemos que si en la camina completa W en lugar de repetir vértices, pasamos directamente a los vértices por los que aún no hemos caminado podemos obtener un ciclo hamiltoneano H que será nuestra aproximación a la solución de tal forma que $c(H) \leq c(W)$, por lo que $c(H) \leq 2c(H^*)$ y por lo tanto esta solución aproximada tiene menos de dos veces el costo de la solución óptima.

Un diagrama de tipo de aproximaciones se muestra en la figura 2.

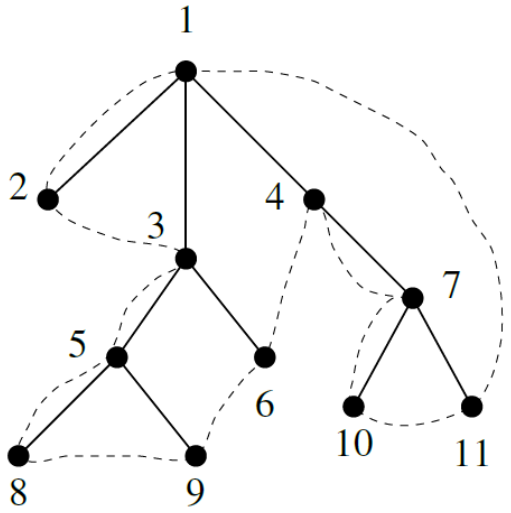

Fig. 2. Árbol generador mínimo y ciclo aproximado respectivo a partir de no repetir vértices en su caminata completa [6], notemos que, por ejemplo, en este caso $d(8,5)+d(5,9) \leq d(8,9)$

Notemos ahora que para obtener esta solución aproximada podemos obtener un árbol generador de peso mínimo (MST por las siglas en inglés de Minimum Spanning Tree) y posteriormente obtener el ciclo de la solución aproximada haciendo una búsqueda a profundidad (DFS por Depth First Search) del MST.

De esta manera el algoritmo para esta solución se puede escribir como:

```
generar MST
Hacer un DFS donde guardemos los vértices
                    conforme los descubramos
agregar 0 al camino descubierto por el DFS
calcular y sumar el costo de cada arista
                      del ciclo propuesto
regresar el costo y el ciclo
```

Para generar el árbol generador de peso mínimo se utilizó el algoritmo de Prim para matrices de distancias, el cual selecciona uno a uno las aristas de peso mínimo que no forma parte de la componente conexa que va creando hasta que todos los vértices están conectados.

Para hacer la implementación más eficiente de este algoritmo sobre matrices de distancia se utilizaron listas auxiliares para llevar registro de los padres de cada vértice agregado, de los vértices visitados y del costo mínimo de agregar cada uno de los vértices restantes en cada iteración. Este algoritmo consiste en iterar hasta encontrar $n-1$ aristas (las necesarias para formar un árbol generador de peso mínimo), en cada iteración se escoge la arista de menor peso que conecte a un vértice que no hayamos integrado al MST, marcamos el vértice elegido como visitado, guardamos su padre y actualizamos las distancias mínimas a los vértices restantes considerando que tenemos un vértice nuevo en nuestro MST.

El algoritmo de Prim puede describirse como:

```
edges = []
dist = 1er renglón de la matriz de dist.
costo = 0
n = numero de vértices
marcamos el vértice 0 como visitado
parrent = lista en 0s de tamaño n

mientras el tamaño de edges no sea n-1:

  para cada vértice:
     escoge el vértice con menor distancia
           en dist que no hayamos visitado

  marca el vértice elegido como visitado

  guarda el padre del vértice visitado

  para cada vértice:
     actualiza dist con los pesos mínimos
     para agregar los vértices restantes
                                 al MST

  guarda en edges la arista del padre al
                       vértice  elegido
  suma a costo agrega el peso de la arista

regresa costo, edges
```

Con esta implementación de Prim sobre matrices de distancia podemos obtener las aristas del árbol generador de peso mínimo y el peso de estas aristas.

Para hacer DFS se utiliza la lista de aristas que regresa el algoritmo de Prim mencionado anteriormente. Este algoritmo de DFS recursivamente obtiene los vértices correspondientes de las aristas y los recorre y hasta llegar a los vértices hojas guardando los vértices nuevos en la búsqueda en una lista, posteriormente repite el procedimiento para el resto de los vértices hijos en cada nivel. Cuando ha terminado de hacer el recorrido por el árbol generador de peso mínimo agrega el vértice inicial a la lista de visitados para cerrar el ciclo.

Este algoritmo puede describirse como:

```
DFS(aristas):
  v1 = primer vértice de la primer arista
                             de las aristas
  vis = lista de vértices visitados
  agregar v1 a vis

  DFS_recursivo(v_):
    para cada arista en aristas
      v1* = 1er vértice de la arista
      v2*  = 2do vértice de la arista
      si v1* == v_:
        si v2* no está en vis:
           agrega el v2* a vis
           DFS_recursivo(v2*)
  DFS_recursivo(v1)

  agregar v1 a vis para regresar al origen
  regresar vis
```

6. ANÁLISIS DE CORRECTITUD Y ANÁLISIS ASINTÓTICO DE TIEMPO Y ESPACIO

A continuación analizamos la correctitud y la complejidad de los algoritmos utilizados para solucionar el problema, comencemos con el análisis de correctitud.

1. Análisis de correctitud

Analicemos primero el algoritmo que realiza búsqueda exhaustiva:

1. Búsqueda exhaustiva

Para mostrar la correctitud de este algoritmo notemos que si X es el conjunto de todas las posibles rutas entonces eventualmente llegaremos a x la ruta de menor distancia. Para demostrarlo supongamos que el algoritmo regresa como mejor solución otra ruta z con peso mayor que x. Notemos que el algoritmo explora por completo el espacio de estados por lo que en algún punto determinó que z no es una mejor solución al problema, lo cual es una contradicción. Por lo tanto el algoritmo por búsqueda exhaustiva es correcto y regresa x con x la mejor solución.

2. Aproximación a la solución

De la discusión vertida en la sección de la propuesta de algoritmos para solucionar el problema podemos concluir

que si logramos proponer un árbol generador de peso mínimo para una gráfica dada y un recorrido por el MST de tal forma que solo se visiten los vértices una vez, podemos generar un ciclo que a lo más tiene un peso de dos veces la solución óptima. Procedamos ahora a probar que el algoritmo de Prim planteado crea un árbol generador de peso mínimo y que la búsqueda a profundidad visita todos los vértices una sola vez.

Para el algoritmo de Prim supongamos que existe un árbol generador de peso mínimo $T*$ con peso menor al árbol T obtenido por Prim. Notemos que esto significa que $T*$ y T tiene al menos una arista diferente, sin pérdida de generalidad asumamos que esta arista es (u,v), la cual se encuentra en $T*$ mientras en T el vértice v está conectado al árbol por la arista (w,v). Notemos que esto significa que el peso de (u,v) es menor que el peso de (w,v), pero en este caso el algoritmo de Prim hubiera escogido a la arista (u,v) sobre la arista (w,v), lo cual es una contradicción. Por lo tanto el algoritmo de Prim es correcto.

Para la búsqueda a profundidad notemos que estamos llevando una lista de vértices visitados y para hacer la llamada a la función DFS_recursivo sobre un vértice necesitamos no haber visitado este vértice anteriormente, por lo que DFS a lo más visitará cada vértice una vez. Notemos también que dado que se está haciendo DFS sobre un árbol generador, existen aristas que llegan a todos los vértices, por lo que la búsqueda a profundidad visitará todos los vértices. Por estas razones la implementación de búsqueda a profundidad es correcta.

Debido a que nuestra solución aproximada se basa en utilizar Prim y DFS y ambos son correctos, la solución aproximada es correcta, dentro del margen de aproximación mencionado anteriormente.

2. Análisis de complejidad en tiempo

De igual manera seccionemos el análisis de complejidad en tiempo para cuando se realiza búsqueda exhaustiva y cuando se hacer la solución por aproximación.

1. Búsqueda Exhaustiva

Para el caso donde se hace búsqueda exhaustiva notemos que el algoritmo, a pesar de realizar algunas operaciones en tiempo lineal en cada llamada recursiva, tiene su complejidad asintótica definida por el espacio de estados que explora.

El espacio de estados, suponiendo que estamos explorando n ciudades, empieza analizando que sucede si se visita cualquiera de las n ciudades inicialmente, para cada una de estas n ciudades explora que pasa si visita después cada una de las siguientes $n-1$ ciudades, y así sucesivamente hasta visitarlas todas, por lo cual la complejidad de la búsqueda exhaustiva es $O(n!)$.

2. Aproximación a la solución

Analicemos ahora los algoritmos necesarios para formular la aproximación a la solución.

Para el algoritmo de Prim notemos que el el ciclo while se ejecutará $n-1$ veces debido a que en cada uno iteración sobre este ciclo se agregará una arista al MST. Dentro del ciclo while existen varios ciclos no anidados entre ellos que revisan cada vértice, los cuales tienen complejidad $O(n)$. Por esta razón la complejidad neta en tiempo de esta implementación del algoritmo de Prim es $O(n^2)$, con n el número de vértices.

Para el algoritmo de DFS notemos que las llamadas recursivas tienen una complejidad en tiempo $O(n)$, mientras que en cada llamada recursiva se revisan cada una de las aristas, como se trata de un árbol el número de aristas es $n-1$, por lo la complejidad de esta implementación de DFS es $O(n^2)$.

De esta manera la complejidad en tiempo completa de encontrar la solución aproximada es $O(n^2)$.

3. Análisis de complejidad en espacio

Dividiendo el análisis de complejidad en espacio para ambos tipos de solución:

1. Búsqueda Exhaustiva

En el caso de la solución por búsqueda exhaustiva notemos a pesar de que exploramos un espacio de estados del tamaño de todas las combinaciones posibles en cada uno de los estados no estamos guardando todas las posibles soluciones y solo estamos comparando las rutas generadas con la mejor ruta hasta el momento, lo cual toma $O(n)$ en espacio con n el número de paradas en la ruta. También notemos que los caminos generados toman $O(n)$ en espacio. Por estas razones la complejidad en espacio está determinada por el tamaño de la matriz de distancias con la que estamos trabajando, la cual es $O(n^2)$.

2. Aproximación a la solución

Notemos que en el algoritmo de Prim se utilizan varias listas de tamaño n, sin embargo, la complejidad en espacio está dominada por la matriz de distancias con la que se trabaja que tiene tamaño n^2, por lo que la complejidad en espacio de este algoritmo es $O(n^2)$.

Para el DFS se utilizan listas tamaño del número de aristas en el árbol generador de peso mínimo y del tamaño del número de vértices, por lo cual la complejidad en espacio del

algoritmo de DFS implementado es $O(n)$.

Por estas razones la complejidad en espacio de para calcular la aproximación a la solución es $O(n^2)$.

7. ESTRATEGIA DE DISEÑO DE ALGORITMOS Y ESTRUCTURA DE DATOS UTILIZADAS

Para la implementación de la solución al problema se decidió utilizar matrices de distancia para representar todas las distancias entre Pueblos Mágicos. La razón para utilizar matrices de distancia es porque es relativamente fácil obtener una submatriz de la matriz que tiene las distancias entre todos los Pueblos Mágicos de acuerdo a las necesidades del usuario, además de que es una estructura de datos conveniente para almacenar a largo plazo la información necesaria para esta aplicación sin tener que hacer consultas a la API de Google Maps cada vez que el usuario necesita encontrar una ruta óptima.

Otra razón para utilizar las matrices de distancia es que su aplicación tanto a la búsqueda exhaustiva y a la solución aproximada resulta sencilla de implementar con ubicaciones arbitrarias de orígenes que el usuario pueda introducir, para lo cual solo necesitamos aumentar la matriz de distancias entre los Pueblos Mágicos seleccionados y el punto de partida. Todos los algoritmos implementados suponen que la primera fila y la primera columna de la matriz de distancia corresponde al punto de partida. De esta forma, solo se necesitan hacer m consultas a la API de Google Maps si el usuario desea visitar m Pueblos Mágicos.

8. APLICACIÓN A LOS DATOS CONCRETOS

Utilizando los algoritmos anteriores y usando la matriz de distancias entre todos los Pueblos Mágicos implementó una función para que el usuario interactué con el programa y pueda seleccionar en primer lugar, los estados que desea visitar. Una vez ingresados los estados que desea visitar, para cada estado ingresado se le pide que ingrese los Pueblos Mágicos que está interesado en conocer pertenecientes al estado. Finalmente se le pide al usuario ingresar el punto de partida.

A partir de estos datos se obtiene la matriz de distancias entre los Pueblos Mágicos que el usuario desea conocer y al momento se calculan las distancias entre el punto de partida del usuario y sus Pueblos Mágicos de interés para crear una matriz de distancias que representa la gráfica completa donde los vértices son los Pueblos Mágicos de interés del usuario y su ubicación de partida y el peso de las aristas es la distancia entre cada par de vértices.

Para la entrada de los estados y de los Pueblos Mágicos de cada uno de los estados seleccionados se utilizaron diccionarios de datos, donde las llaves son números enteros y los valores son los nombres de los estados y de los Pueblos Mágicos. Al ingresar los Pueblos Mágicos el usuario puede ingresar 1"para indicar que desea conocer todos los Pueblos Mágicos de un estado dado. Esto se hizo con fin de agilizar la entrada de datos del usuario. Para la entrada de la ubicación de partida el usuario introduce una cadena de texto.

Con esta información se determina si la cantidad de ciudades a recorrer son 10 o menos, si lo son se calcula el ciclo hamiltoneano de peso mínimo con búsqueda combinatoria. En cambio, si el usuario desea conocer más de 10 Pueblos Mágicos se procede a calcular una aproximación a la solución óptima por los métodos descritos en secciones anteriores.

A continuación se muestran algunos ejemplos de la aplicación al problema de los Pueblos Mágicos:

Ejemplos de aplicación

Ejemplo 1:

Entradas:

- Estados a visitar: Oaxaca

- Pueblos Mágicos a visitar en Oaxaca: Todos

- Punto de Partida: San Pablo Etla Oaxaca

Salida:

La ruta a seguir es:

San Pablo Etla Oaxaca

Capulalpam de Mendez

San Pablo Villa de Mitla

Mazunte

Santa Catarina Juquila

Huautla de Jimenez

San Pedro y San Pablo Teposcolula

San Pablo Etla Oaxaca

Con un costo de 1452.081 km

Una imagen de esta ruta se muestra en la siguiente figura:

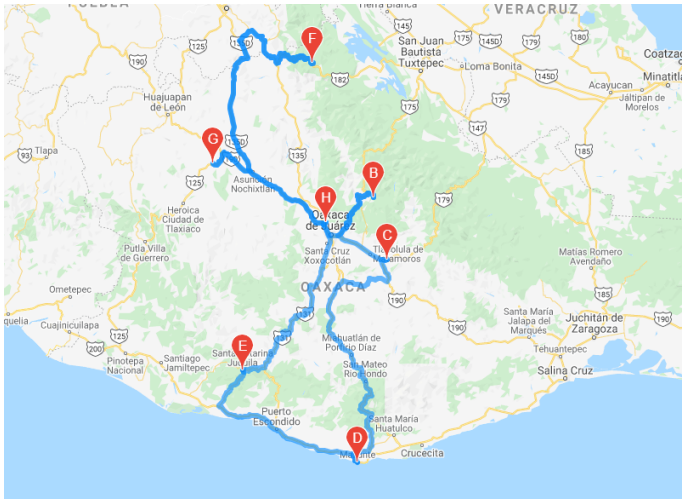

Fig. 3. Ruta recomendada para visitar todos los Pueblos Mágicos de Oaxaca, partiendo de San Pablo Etla, Oaxaca

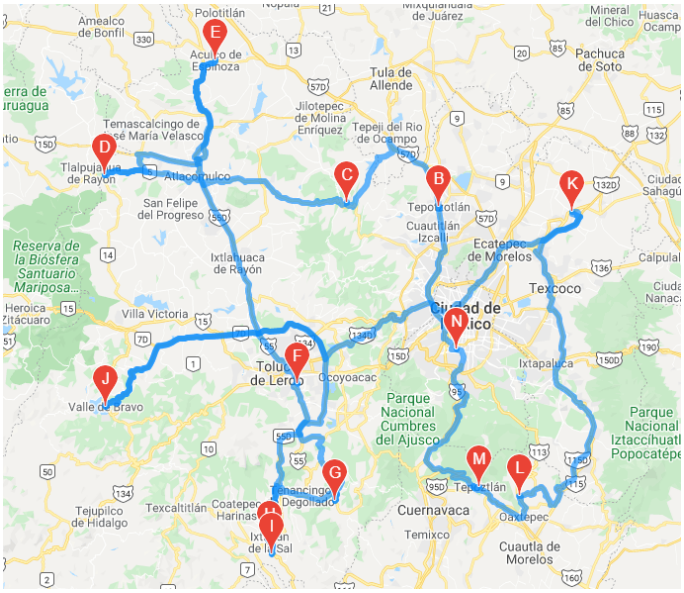

Fig. 4. Ruta recomendada para visitar todos los Pueblos Mágicos del Estado de México y Morelos, partiendo de Coyoacán.

Ejemplo 2:

Entradas:

- Estados a visitar: México y Morelos

- Pueblos Mágicos a visitar en México: Todos

- Pueblos Mágicos a visitar en Morelos: Todos

- Punto de Partida: Coyoacán

Salida:
La ruta a seguir es:
Coyoacán
Tepotzotlán
Villa del Carbón
El Oro
Aculco
Metepec
Malinalco
Ixtapan de la Sal
Tonatico
Valle de Bravo
Teotihuacán y San Martín de las Piramides
Tlayacapan
Tepoztlán
Coyoacán
Con un costo de 1066.872 km

Notemos que en este caso la ruta se está calculando por aproximación dada la cantidad de Pueblos Mágicos que el usuario ingresó. La ruta sugerida se muestra en la siguiente figura.

Ejemplo 3:

Entradas:

- Estados a visitar: Jalisco

- Pueblos Mágicos a visitar en Jalisco: Todos

- Punto de Partida: Puerto Vallarta

Salida:
La ruta a seguir es:
Puerto Vallarta
San Sebastián del Oeste
Mascota
Talpa de Allende
Tapalpa
Mazamitla
Ajijic
Lagos de Moreno
Tlaquepaque
Tequila
Puerto Vallarta
Con un costo de 1307.694 km

La ruta sugerida se muestra en la siguiente figura.

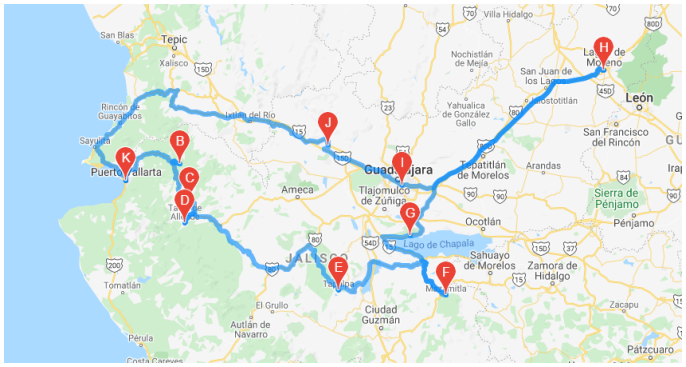

Fig. 5. Ruta recomendada para visitar todos los Pueblos Mágicos del Estado de Jalisco partiendo de Puerto Vallarta.

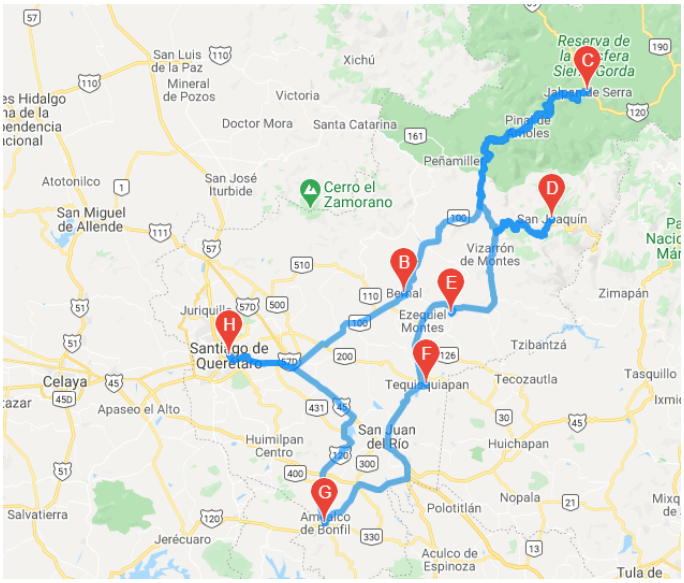

Fig. 6. Ruta recomendada para visitar todos los Pueblos Mágicos del Estado de Querétaro partiendo de Querétaro Centro.

Ejemplo 4:

Entradas:

- Estados a visitar: Querétaro

- Pueblos Mágicos a visitar en Querétaro: Todos

- Punto de Partida: Querétaro Centro

Salida:

La ruta a seguir es:

Querétaro Centro

Bernal

Jalpan de Serra

San Joaquín

Cadereyta de Montes

Tequisquiapan

Amealco de Bonfil

Querétaro Centro

Con un costo de 553.775 km

La ruta sugerida se muestra en la siguiente figura.

En los ejemplos anteriores se ilustran posibles entradas que puede dar el usuario para encontrar las rutas óptimas para conocer Pueblos Mágicos partiendo y regresando del lugar donde este decida con los algoritmos mencionados anteriormente y utilizando la base de datos de las matrices de distancia y la base de datos de coordenadas para hacer la graficación de las rutas.

9. CONCLUSIONES Y POSIBLE TRABAJO A FUTURO

En este trabajo se implementaron algoritmos, tanto exactos por búsqueda exhaustiva como de aproximación al problema del agente viajero. El problema del agente viajero en este caso es el problema de un turista que quiere conocer un conjunto arbitrario de los Pueblos Mágicos de México desde un punto de partida y de retorno arbitrario.

Debido a que el problema de encontrar el ciclo de longitud mínima es un problema que no se puede resolver en tiempo polinomial, dependiendo de la instancia a resolver introducida por el usuario el algoritmo decide si resolverla por búsqueda exhaustiva o por aproximación. De esta manera se puede ofrecer al usuario una propuesta de ruta a seguir en caso de que decida realizar este viaje turístico.

Que el usuario pueda obtener una ruta óptima y pueda elegir visitar cualesquiera de los Pueblos Mágicos del territorio nacional puede incentivar que este se decida a realizar el viaje turístico y aporte a la derrama económica de los municipios que visite.

Debido a que ningún estado tiene más de 10 Pueblos Mágicos es factible encontrar rutas óptimas a los Pueblos Mágicos de cada estado partiendo de su ciudad capital, lo cual podría servir para ofertar viajes turísticos y fomentar el turismo dentro de los estados.

Como trabajo a futuro pueden implementarse mejores aproximaciones a la utilizada para instancias de más de 10 Pueblos Mágicos, entra las cuales podría utilizarse algún algoritmo genético implementado en paralelo para encontrar rutas óptimas. También se podría mejorar la interfaz de usuario para no tener que ejecutar este programa desde el código fuente y en cambio poderlo ejecutar desde algún sitio web.

BIBLIOGRAFÍA

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.
[2] T. S. Team, *How old is the universe?* NASA, 2000. Consultado el 03 de enero en `https://starchild.gsfc.nasa.gov/docs/StarChild/questions/question28.html`.
[3] S. de Turismo, *Se instituye al 5 de octubre como el "Día Nacional de los Pueblos Mágicos"*. Gobierno de México, 2020. Consultado el 1 de enero del 2021 en `https://www.gob.mx/sectur/articulos/pueblos-magicos-206528`.
[4] S. de Turismo, *Localidades que cuentan con el nombramiento de Pueblo Mágico (DGGD)*. Gobierno de México, 2017.
[5] S. de Turismo, *Pueblos Mágicos de México*. Gobierno de México, 2020. Consultado el 1 de enero del 2021 en `https://www.gob.mx/sectur/articulos/pueblos-magicos-206528`.
[6] S. S. Skiena, *The algorithm design manual*. Springer International Publishing, 2020.