Universidad Nacional Autónoma de México

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

Axel Daniel Malváez Flores

Logística de la Fórmula 1

Proyecto Final

Matemáticas Discretas

3 de diciembre de 2022

Introducción

La Fórmula 1 es uno de los deportes más vistos y más caros que han existido jamás y, si tenemos en cuenta el tamaño de los eventos, los coches de carreras multimillonarios y el espectáculo que todos podemos disfrutar, es una verdadera hazaña de habilidad e ingeniería que las carreras puedan celebrarse cada año. Es por esto que este evento se realiza en distintas ciudades al rededor del mundo en fines de semana llegando a hacer una cantidad enorme de viajes. Ahora si tenemos en cuenta la cantidad de equipos, tecnología y, por supuesto, los propios coches, que viajan por todo el mundo, la Fórmula 1 tiene que ser una de las mayores hazañas de planificación logística que se llevan a cabo anualmente.

Se estimó que en 2019, se transportaron más de 1000 toneladas de equipo a cada carrera cuando se combinaron todos los equipos y considerando las limitaciones de tiempo debido al calendario de carreras, existe aún más presión para hacer las cosas bien. El equipo de regatas más grande suele tener 100 personas y 50 toneladas de carga en cada carrera, por lo que la planificación y la organización son absolutamente fundamentales para que haya alguna posibilidad de éxito en la carrera y una transición sin problemas de una carrera a otra para todos los implicados.

Problema a resolver

Es por esto que se desea crear una ruta óptima donde podamos visitar cada circuito que esté dentro del contrato de la Fórmula 1 de manera que la distancia que tengamos que recorrer entre circuitos sea la mínima, esto por los altos cosos que conlleva transportar todo el equipo requerido para un Gran Premio durante un fin de semana, esto se podría reducir a resolver el famoso Travelling Salesman Problem (TSP) el cual es un problema:

Lista de los Grandes Premios que se tienen para esta temporada.

- GP de Bahrein
- GP de Arabia Saudí
- GP de Australia
- GP de China
- GP de Azerbaiyán

- GP de Miami (USA)
- GP de Emilia Romagna (Italia)
- GP de Mónaco
- GP de España
- GP de Canadá
- GP de Austria
- GP de Gran Bretaña
- GP de Hungría
- GP de Bélgica
- GP de los Países Bajos
- GP de Italia
- GP de Singapur
- GP de Japón
- GP de Qatar
- GP de Estados Unidos
- GP de México
- GP de Brasil
- GP de Las Vegas (USA)
- GP de Abu Dhabi

Modelado del Problema

Dentro del Data Set destacan las columnas "name" la cual corresponde al nombre del circuito, "location" la cual corresponde a la ubicación (ciudad) en donde se encuentra el circuito, "country" la cual corresponde al país en donde se ubica el circuito y sus coordenadas correspondientes en "latitudz "longitud".

Entonces utilizando esta información podemos crear una gráfica de las vistas en clase donde cada nodo corresponde a la ubicación de un circuito en el mundo y las aristas son aristas ponderadas cuyo valor corresponde a la distancia entre dos circuitos. Así podemos crear una gráfica completa ya sea de 76 nodos o bien de 24 correspondientes a todos los circuitos en los que se han realizado Grand Prixe's o bien a los circuitos en donde la Fórmula 1 tiene presencia actual respectivamente.

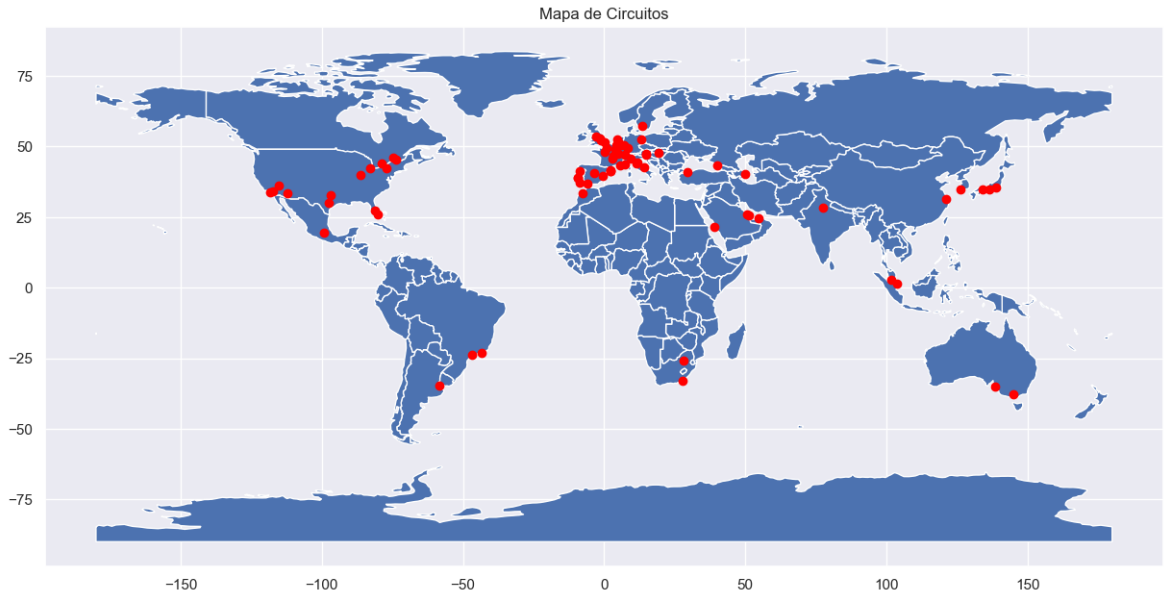


Figura 1: Mapa Mundial de Circuitos de F1

Problema algorítmico general

El problema que buscamos resolver se divide en dos partes:

1. Árbol de pesos mínimo

   El árbol generador de peso mínimo es aquel que comienza desde un vértice y encuentra todos sus nodos accesibles y las relaciones en conjunto que permiten que se conecten dichos nodos con el menor peso posible. Para esto utilizaremos tres algoritmos, el algoritmo de **Boruvka**, el algoritmo de **Prim**, el algoritmo de **Kruskal**.

2. TSP

   En el Problema del Agente Viajero – TSP (Travelling Salesman Problem), el objetivo es encontrar un recorrido completo que conecte todos los nodos de una red, visitándolos tan solo una vez y volviendo al punto de partida, y que además minimice la distancia total de la ruta, o el tiempo total del recorrido.

   Y dado que este tipo de problemas tiene gran aplicación en el ámbito de la logística y distribución, así como en la programación de curvas de producción, es por eso que trataremos de resolverlo para encontrar la ruta más eficiente que podemos seguir en nuestro viaje por el mundo.

Propuesta de solución

Primero tendremos que obtener las distancias entre cada par de vértices de nuestra gráfica para saber cuanto nos cuesta viajar desde un circuito a otro por lo que tendremos que calcular las combinaciones ya sea de 76 o 24 circuitos en 2, es decir:

$$\binom{n}{2}$$

siendo n el número de circuitos. Posteriormente creamos nuestra gráfica calculando las distancias de las parejas que residen en el conjunto retornado por el cálculo de las combinaciones. Finalmente aplicamos los algoritmos propuestos (Boruvka, Prim, Kruskal y TSP) para finalmente obtener un árbol en el que podamos recorrer nuestra gráfica o bien una ruta específica que podamos seguir.

Análisis y estrategias

Análisis de los algoritmos

- Algoritmo de Prim

  Es un algoritmo que encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si la gráfica no es conexa, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicha gráfica no conexa.

  Algoritmo:

  1. Inicializamos un árbol con un único vértice, elegido arbitrariamente de la gráfica.

2. Incrementamos el árbol con una arista de la siguiente forma: de las aristas que conectan un vértice en el árbol y otro vértice aún no del árbol, encontrar la arista de peso mínimo y trasferirla al árbol.

3. repetimos paso 2 (hasta que todos los vértices se encuentren en el árbol).

La complejidad del algoritmo depende bastante de la estructura en la que se esté implementando la gráfica y la forma en la que se ordenan las aristas por pesos, lo cual puede ser hecho usando una cola de prioridades.

Utilizando una matriz de adyacencias, tenemos que el algoritmo corre en tiempo $O(|V|^2)$, no obstante si tenemos que G se almacena en un Fibonacci Heap, entonces su complejidad en tiempo se reduce a $O(|E| + |V| \log |V|)$. Ahora dado que solo estamos almacenando en estructura las aristas y los vértices la complejidad en espacio es de $O(|V| + |E|)$

Prueba de correctud:

Sea G una gráfica conexa ponderada. En cada iteración del algoritmo de Prim, una arista debe ser encontrada que conecte un vértice en una subgráfica y otro vértice fuera de la subgráfica (el árbol que estamos construyendo). Dado que P es conexa, habrá siempore un camino a cada vértice. La salida del algoritmo es un árbol T el cuál es un árbol generador de peso mínimo. Para demostrar esto sea T_1 un árbol generador de peso mínimo de G. Si $T_1 = T$, entonces T sería un árbol generador de peso mínimo. De otra forma, sea e la primer arista agregada durante la construcción del árbol T que no está en el árbol T_1 y sea V el cojunto de vértices conectados por las aristas agregadas antes que la arista e. Por tanto un punto final de la arista e está en el conjunto de vértices V y el otro punto final no lo está (por como se añaden aristas en este algoritmo). Dado que el árbol T_1 es un árbol generador de peso mínimo de la gráfica G, existe un camino en el árbol T_1 que une a los dos puntos finales. Conforme vamos recorriendo el camino, debemos ir econtrando una arista f que une un vértice en V y otro que no está en V. Ahora, en el momento de la itercación en que la arista e es agregada al árbol Y, la arista f puede también ser agregada y sería agregada en lugar de la arista e si su peso es menor a el de e, dado que f no fue agregada entonces notamos que $w(f) \geq w(e)$. Sea T_2 la gráfica obtenida al remover la arista f de T_1 y añadida la arista e a T_1. Ahora es fácil mostrar que T_2 es conexo, pues tiene el mismo nçúmero de aristas que T_1, y el peso total de sus aristas no es más grande que el de T_1, por lo tanto es también un árbol generador mínimo de la gráfica G y el cual contiene a e y todas las aristas añadidas durante la construcción de V. Repetimos los pasos de arriba y eventualmente obtendrémos un árbol generador mínimo de la gráfica P que es idéntico a T. Esto prueba que T es un árbol generador mínimo.[**?**]

- Algoritmo de Boruvka

El algoritmo de Borůvka es un algoritmo que sirve para encontrar el árbol generador mínimo en una gráfica ponderado en el que todas sus aristas tienen distinto peso. Este algoritmo puede llegar a obtener una complejidad de $O(|E| \log |V|)$. En una gráfica plana puede implementarse para ejecutarse en tiempo lineal, eliminando aristas de menor peso entre cada par de nodos después de cada etapa del algoritmo.

Algoritmo:

El algoritmo comienza examinando cada vértice y añadiendo la arista de menor peso desde ese vértice a otro en la gráfica, sin tener en cuenta las aristas ya agregadas, y continua uniendo estos grupos de la misma manera hasta que se completa un árbol que cubra todos los vértices. Si denominamos a cada vértice o conjunto de vértices conectados como una componente, el algoritmo de Boruvka es:

```
1. Comenzamos con una grafica conexa G en el que todas sus
   aristas tengan distinto peso, y un conjunto vaco de aristas
   T.
2. Mientras los vertices de G conectados por T sean disjuntos:
3.   Crear un conjunto vacio de aristas S
4.   Para cada componente:
5.       Crear un conjunto vacio de aristas S
6.       Para cada vertice v en el componente:
7.           Agregar la arista de menor peso desde el vertice v
   a otro vertice en un componente disjunto a S
8.       Agregar la arista de menor peso en S a E
9.   Agregar el conjunto resultante E a T.
10. El conjunto de aristas resultante T es el arbol generador
   minimo de G.
```

- Algoritmo de Kruskal

El algoritmo de Kruskal es un algoritmo que sirve para encontrar un árbol generador mínimo en una gráfica conexa y ponderada. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un árbol expandido mínimo para cada componente conexa).

Es un ejemplo de algoritmo voraz cuya complejidad en estructuras de datos sim-

ples y realizando DFS tendríamos un tiempo de $O(m \log m + m(m+n))$ pasos, lo cual es demasiado. No obstante este tiempo se mejora usando una estructura de datos para conjuntos disjuntos (**disjoint-set data structure**) y así poder controlar qué vértices están en qué componentes. Es necesario hacer operaciones de orden O(m) ya que por cada arista hay dos operaciones de búsqueda y posiblemente una unión de conjuntos. Incluso una estructura de datos sobre conjuntos disjuntos simple con uniones por rangos puede ejecutar las operaciones mencionadas en O(m log n). Por tanto, la complejidad total en dicha estructurá en tiempo es de $O(m \log n)$.

Al igual que los anteriores la complejidad en espacio de este algorito es de $O(|V| + |E|)$ pues son aquellas estructuras donde guardamos nuestros vértices y aristas.

- Travelling Salesman Problem (TSP)

Rápidamente veamos que, de manera muy intuitiva (y un poco informal) un problema que es **NP** es fácil de verificar, es decir si un oráculo nos arroja un resultado, será fácil con un algoritmo verificar si en efecto es un resultado válido o no de algún problema que estemos tratando. Por otra parta, un problema del tipo **NP-hard** es un problema que resultará difícil de encontrar una posible solución. Finalmente un problema perteneciente al conjunto de problemas **NP-Complete** es un problema que es tanto **NP** como **NP-hard** y por tanto será fácil de verificar, pero difícil encontrar soluciones.

Ahora si, el problema del agente viajero (TSP por sus siglas en inglés, Travelling Salesman Problem) 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 visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?

Este es un problema del tipo **NP-Hard** dentro en la optimización combinatoria, muy importante en investigación operativa y en ciencias de la computación. En Ciencias de la Computación, la complejidad, que recae en la versión de decisión del algoritmo TSP (donde dada una longitud L, la tarea es decidir si la gráfica tiene un tour de a lo más longitud L) pertenece a la clase de problemas del tipo $NP - Complete$. Por tanto, es posible que en el peor caso en tiempo para cualquier algoritmo implementado para TSP crezca superpolinomialmente (pero no más que exponencialmente) con respecto al número de ciudades.

El tiempo de ejecución es un factor polinómico de orden $O(n!)$, el cuál es el factorial del número de ciudades, por lo que esta solución es inviable para dado solamente 20 ciudades. Una de las mejores aplicaciones de la programación

dinámica es el algoritmo Held–Karp que resuelve el problema en un orden de $O(n^2 2^n)$.

Estrategias

Para este proyecto, se utilizó la librería de **Network X** de Python para implementar las gráficas en una computadora. La representación de una gráfica en esta librería está basada en una estructura de datos conocida como **lista de adyacencias** y está implementada usando la estructura de datos **diccionarios**. Lo cuál es una de las formas más eficientes de implementar una gráfica debido a la complejidad de acceso a los datos de la misma. $O(1)$ para obtención de datos y recorrido de los mismos en $O(n)$.

De igual forma dado que la librería **Network X** ya tiene implementados los algoritmos, fue sencillo obtener los recorridos **DFS** y **BFS**, los árboles de peso mínimo con **Prim**, **Boruvka** y **Kruskal** y la ruta que resuelve **TSP**. Para el caso de **TSP**, basandonos en la documentación escrita por el sitio web de la librería, la función que nos regresa la ruta `travelling-salesman-problem` nos permite aproximar soluciones a este problema en redes (gráficas) que no son completas y/o en donde el agente no necesita visitar todos los nodos.

Esta última función procede en dos pasos. Primero crea una gráfica completa usando todas las parejas de caminos más cortos entre los nodos en V. Los pesos de las aristas de la nueva gráfica son las longitudes de los caminos entre cada par de nodos en la gráfica original. El segundo paso, un algoritmo (por default: christofides) es usado para aproximar el ciclo hamiltoniano mínimo en esta nueva gráfica.

Aplicación a los datos

Creamos nuestras gráficas ponderadas, una con todos los circuitos en los que se han corrido Grandes Premios y otra con los circuitos firmados para la temporada 2023:

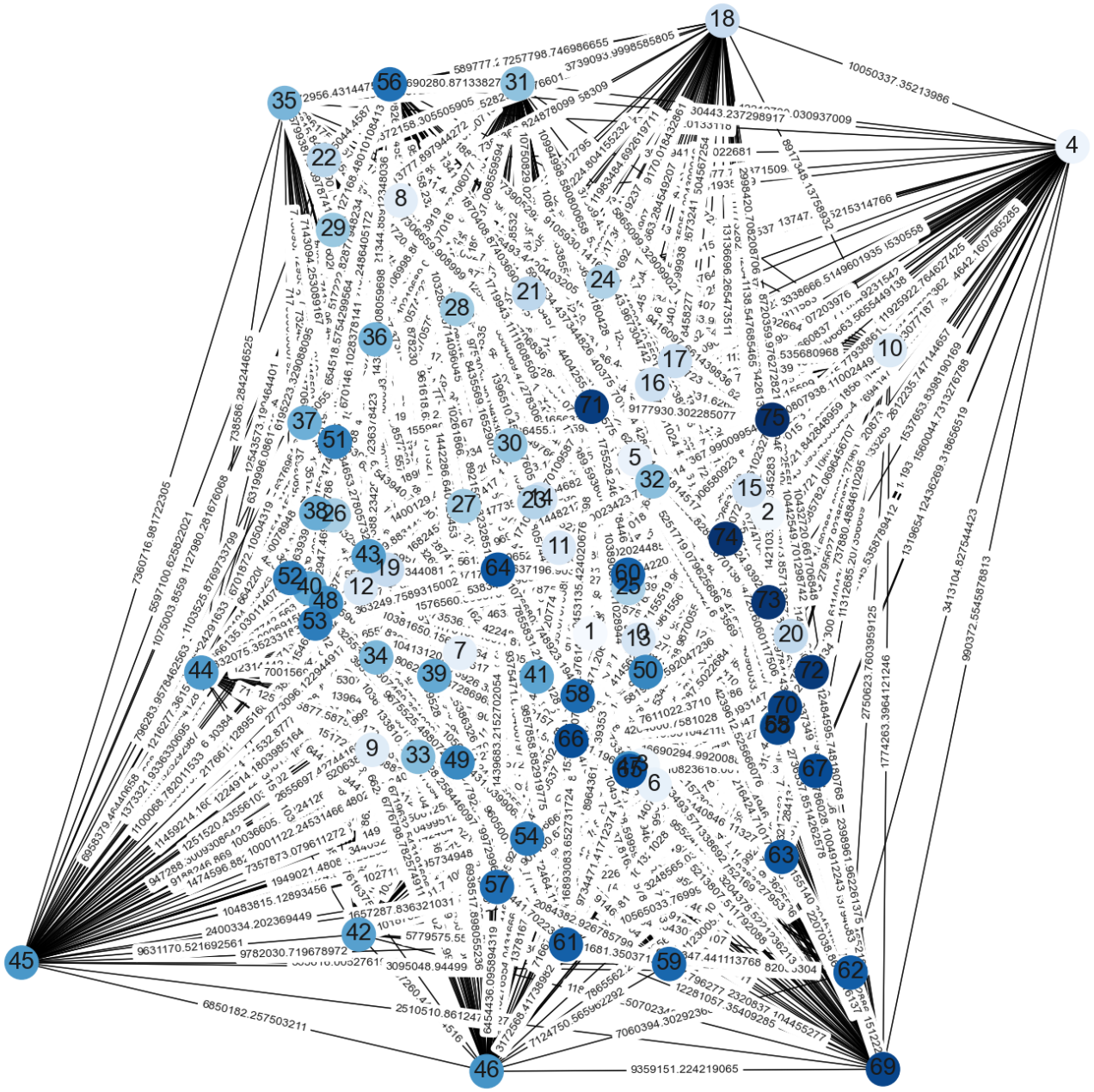


Figura 2: Gráfica 76 Circuitos

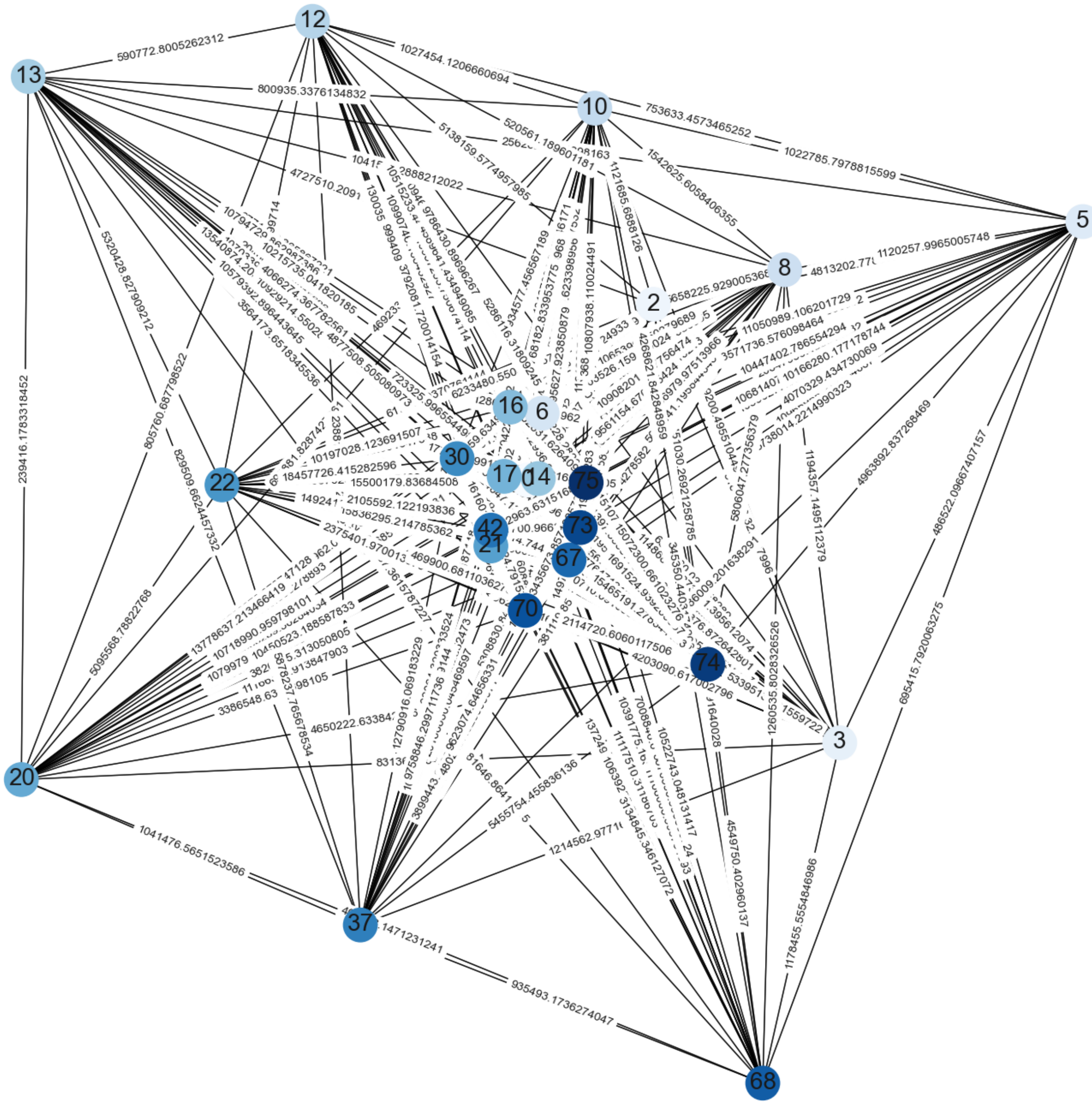
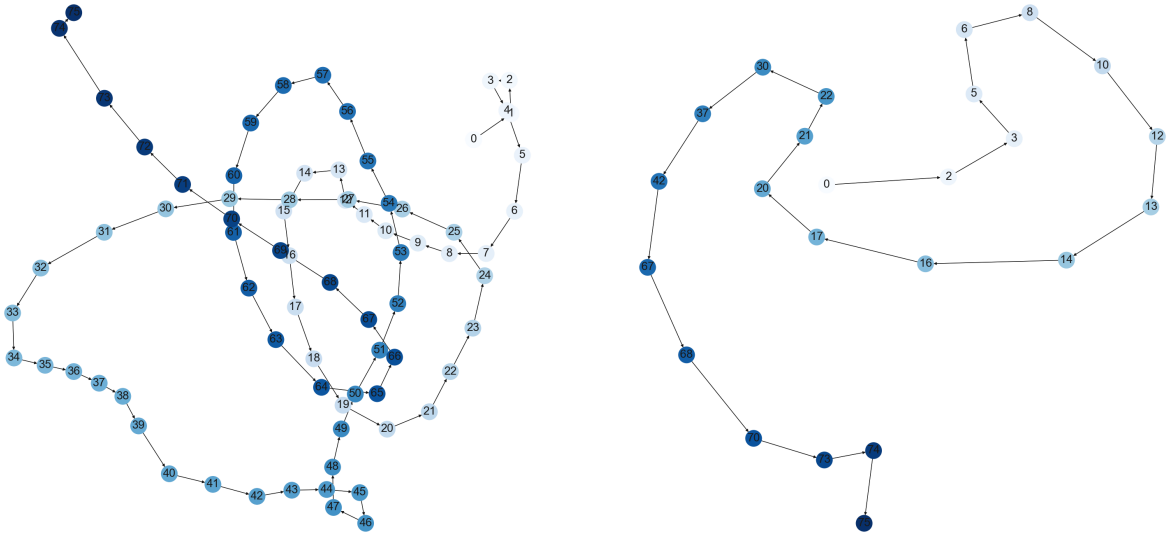


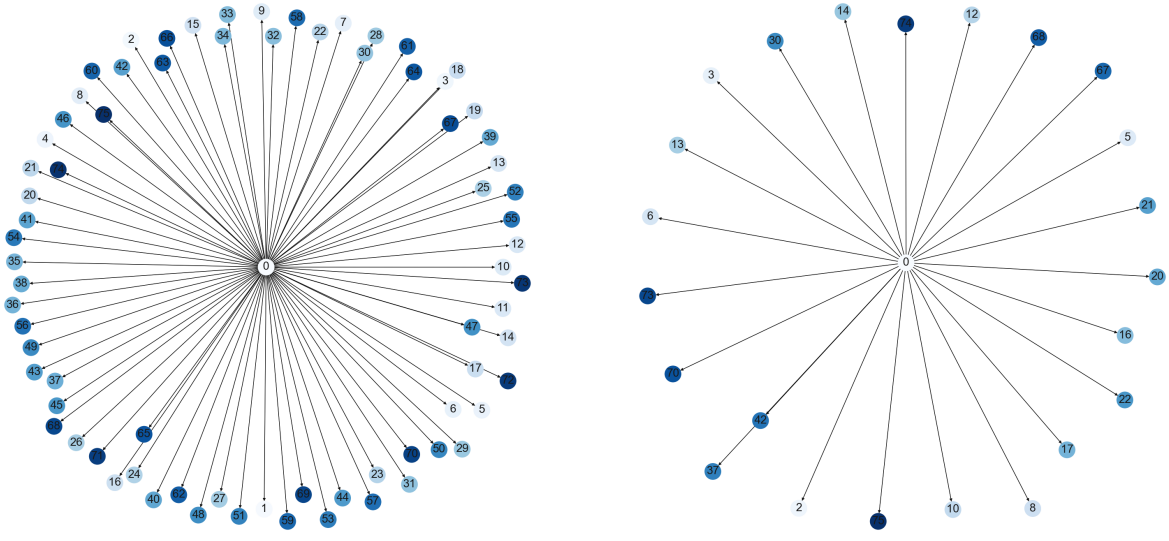
Figura 3: Gráfica 24 Circuitos

Aplicamos los recorridos tanto DFS como BFS para ambas lo cual resulta en



(a) 76 Circuitos (b) 24 Circuitos

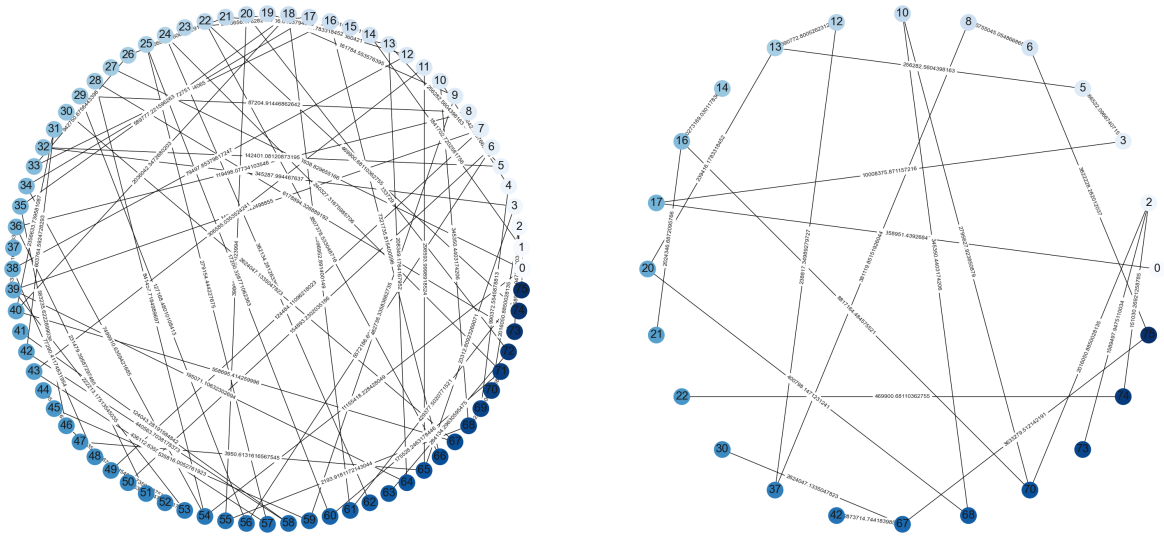
Figura 4: Algoritmo DFS



(a) 76 Circuitos (b) 24 Circuitos

Figura 5: Algoritmo BFS

Ahora dado que el árbol de pesos mínimo fue el mismo para todos los algoritmos implementados (Prim, Boruvka y Kruskal). Tenemos que los árboles de pesos mínimos son los siguientes:



(a) 76 Circuitos

(b) 24 Circuitos

Figura 6: Algoritmo Árbol Generador de Pesos Mínimo

Finalmente los resultados obtenidos por el algoritmo que resuelve **TSP** en **Network X** fueron (únicamente para la gráfica con 24 circuitos):

- TSP
  - Iniciaremos nuestro viaje en el Albert Park Grand Prix Circuit y posteriormente iremos al Autódromo José Carlos Pace recorriendo una cantidad de : 14358.951 km
  - posteriormente seguimos desde Autódromo José Carlos Pace hacia el Circuit de Barcelona-Catalunya recorriendo una cantidad de : 10008.376 km
  - posteriormente seguimos desde Circuit de Barcelona-Catalunya hacia el Circuit de Monaco recorriendo una cantidad de : 486.522 km
  - posteriormente seguimos desde Circuit de Monaco hacia el Autodromo Nazionale di Monza recorriendo una cantidad de : 256.283 km

- posteriormente seguimos desde Autodromo Nazionale di Monza hacia el Autodromo Enzo e Dino Ferrari recorriendo una cantidad de : 239.416 km
- posteriormente seguimos desde Autodromo Enzo e Dino Ferrari hacia el Red Bull Ring recorriendo una cantidad de : 400.798 km
- posteriormente seguimos desde Red Bull Ring hacia el Hungaroring recorriendo una cantidad de : 345.350 km
- posteriormente seguimos desde Hungaroring hacia el Circuit de Spa-Francorchamps recorriendo una cantidad de : 1027.454 km
- posteriormente seguimos desde Circuit de Spa-Francorchamps hacia el Circuit Park Zandvoort recorriendo una cantidad de : 238.817 km
- posteriormente seguimos desde Circuit Park Zandvoort hacia el Silverstone Circuit recorriendo una cantidad de : 381.120 km
- posteriormente seguimos desde Silverstone Circuit hacia el Baku City Circuit recorriendo una cantidad de : 4278.583 km
- posteriormente seguimos desde Baku City Circuit hacia el Bahrain International Circuit recorriendo una cantidad de : 2016.051 km
- posteriormente seguimos desde Bahrain International Circuit hacia el Losail International Circuit recorriendo una cantidad de : 151.030 km
- posteriormente seguimos desde Losail International Circuit hacia el Yas Marina Circuit recorriendo una cantidad de : 469.901 km
- posteriormente seguimos desde Yas Marina Circuit hacia el Jeddah Corniche Circuit recorriendo una cantidad de : 2105.592 km
- posteriormente seguimos desde Jeddah Corniche Circuit hacia el Circuit Gilles Villeneuve recorriendo una cantidad de : 10836.653 km
- posteriormente seguimos desde Circuit Gilles Villeneuve hacia el Miami International Autodrome recorriendo una cantidad de : 3822.228 km
- posteriormente seguimos desde Miami International Autodrome hacia el Circuit of the Americas recorriendo una cantidad de : 3633.280 km
- posteriormente seguimos desde Circuit of the Americas hacia el Autódromo Hermanos Rodríguez recorriendo una cantidad de : 2624.047 km
- posteriormente seguimos desde Autódromo Hermanos Rodríguez hacia el Las Vegas Street Circuit recorriendo una cantidad de : 4526.349 km
- posteriormente seguimos desde Las Vegas Street Circuit hacia el Suzuka Circuit recorriendo una cantidad de : 10149.068 km
- posteriormente seguimos desde Suzuka Circuit hacia el Shanghai International Circuit recorriendo una cantidad de : 2024.347 km
- posteriormente seguimos desde Shanghai International Circuit hacia el Marina Bay Street Circuit recorriendo una cantidad de : 10273.169 km

- posteriormente seguimos desde Marina Bay Street Circuit hacia el Albert Park Grand Prix Circuit recorriendo una cantidad de : 36205.245 km

- TSP (sin retorno a la ciudad central)

  - Iniciaremos nuestro viaje en el Jeddah Corniche Circuit y posteriormente iremos al Bahrain International Circuit recorriendo una cantidad de : 1589.498 km
  - posteriormente seguimos desde Bahrain International Circuit hacia el Yas Marina Circuit recorriendo una cantidad de : 617.616 km
  - posteriormente seguimos desde Yas Marina Circuit hacia el Losail International Circuit recorriendo una cantidad de : 469.901 km
  - posteriormente seguimos desde Losail International Circuit hacia el Baku City Circuit recorriendo una cantidad de : 2114.721 km
  - posteriormente seguimos desde Baku City Circuit hacia el Shanghai International Circuit recorriendo una cantidad de : 8817.164 km
  - posteriormente seguimos desde Shanghai International Circuit hacia el Marina Bay Street Circuit recorriendo una cantidad de : 10273.169 km
  - posteriormente seguimos desde Marina Bay Street Circuit hacia el Suzuka Circuit recorriendo una cantidad de : 11648.137 km
  - posteriormente seguimos desde Suzuka Circuit hacia el Las Vegas Street Circuit recorriendo una cantidad de : 10149.068 km
  - posteriormente seguimos desde Las Vegas Street Circuit hacia el Circuit of the Americas recorriendo una cantidad de : 2873.715 km
  - posteriormente seguimos desde Circuit of the Americas hacia el Autódromo Hermanos Rodríguez recorriendo una cantidad de : 2624.047 km
  - posteriormente seguimos desde Autódromo Hermanos Rodríguez hacia el Miami International Autodrome recorriendo una cantidad de : 4991.348 km
  - posteriormente seguimos desde Miami International Autodrome hacia el Circuit Gilles Villeneuve recorriendo una cantidad de : 3822.228 km
  - posteriormente seguimos desde Circuit Gilles Villeneuve hacia el Silverstone Circuit recorriendo una cantidad de : 5755.045 km
  - posteriormente seguimos desde Silverstone Circuit hacia el Circuit Park Zandvoort recorriendo una cantidad de : 381.120 km
  - posteriormente seguimos desde Circuit Park Zandvoort hacia el Circuit de Spa-Francorchamps recorriendo una cantidad de : 238.817 km
  - posteriormente seguimos desde Circuit de Spa-Francorchamps hacia el Autodromo Nazionale di Monza recorriendo una cantidad de : 590.773 km

- posteriormente seguimos desde Autodromo Nazionale di Monza hacia el Autodromo Enzo e Dino Ferrari recorriendo una cantidad de : 239.416 km
- posteriormente seguimos desde Autodromo Enzo e Dino Ferrari hacia el Red Bull Ring recorriendo una cantidad de : 400.798 km
- posteriormente seguimos desde Red Bull Ring hacia el Hungaroring recorriendo una cantidad de : 345.350 km
- posteriormente seguimos desde Hungaroring hacia el Circuit de Monaco recorriendo una cantidad de : 1022.786 km
- posteriormente seguimos desde Circuit de Monaco hacia el Circuit de Barcelona-Catalunya recorriendo una cantidad de : 486.522 km
- posteriormente seguimos desde Circuit de Barcelona-Catalunya hacia el Autódromo José Carlos Pace recorriendo una cantidad de : 10008.376 km
- posteriormente seguimos desde Autódromo José Carlos Pace hacia el Albert Park Grand Prix Circuit recorriendo una cantidad de : 14358.951 km

Conclusiones y posible trabajo a futuro

El uso de **Network X** para implementar nuestras gráficas fue de bastante utilidad debido a que los algoritmos y las estructuras de datos se actualizan constantemente para poder tener resultados óptimos y rápidos. El tiempo que se tardó nuestra gráfica en crearse pudo ser bastante (app. 30s para 76 vértices) y (app. 15s para 24 vértices) en máquina local, no obstante los algoritmos de búsqueda de árboles mínimos fueron rápidos, así como TSP (lo cuál resulta bastante raro) debido a la complejidad en resolver este problema.

El resultado del algoritmo TSP causa un poco de confusión debido a que algo que podríamos esperarnos sería hacer los viajes por continente (sin menbargo la intuición no siempre es lo mejor).

Es así que logramos obtener una ruta donde viajemos por los 24 circuitos de manera que las distancias fueran las mínimas posibles y por tanto los costos de transportación se redujeran.

Bibliografía

Formula 1 Introducción

- Gilbert International. (2021, September 23). La logística de la Fórmula 1: Transportar por el mundo. https://blog.gilbertintl.com/...
- Formula 1, F. (2022, season-02). 2023 Formula 1 calendar revealed. Formula 1. Retrieved December 2, 2022, from https://www.formula1.com/...
- Galán, M. (2022, September 20). Confirmado el calendario de la F1 2023: carreras, sedes, fechas y más.
  https://es.motorsport.com/f1/...

Algoritmos

- minimum spanning tree — NetworkX 2.8.8 documentation. (n.d.).
  https://networkx.org/documentation/...
- Approximations and Heuristics — NetworkX 2.8.8 documentation. (n.d.).
  https://networkx.org/documentation/...
- Árboles de peso mínimo: algoritmos de Prim y Kruskal — Matemáticas Discretas para Ciencia de Datos. (n.d.).
  https://madi.nekomath.com/P5/ArbolPesoMin.html
- colaboradores de Wikipedia. (2020, November 16). Algoritmo de Kruskal. Wikipedia, La Enciclopedia Libre.
  https://es.wikipedia.org/wiki/Algoritmo$_d e_K ruskal$
- colaboradores de Wikipedia. (2021, January 24). Algoritmo de Boruvka. Wikipedia, La Enciclopedia Libre.
  https://es.wikipedia.org/wiki/Algoritmo$_d e_B oruvka$
- colaboradores de Wikipedia. (2022, November 12). Algoritmo de Prim. Wikipedia, La Enciclopedia Libre.
  https://es.wikipedia.org/wiki/Algoritmo$_d e_P rim$
- Contributor, T. (2020, June 26). traveling salesman problem (TSP). WhatIs.com.
  https://www.techtarget.com/whatis/...

- Wikipedia contributors. (2022c, November 18). Travelling salesman problem. Wikipedia.
  https://en.wikipedia.org/wiki/Travelling...
- Confusion about NP-hard and NP-Complete in Traveling Salesman problems. (2018, April 14). Stack Overflow.
  https://stackoverflow.com/questions/...