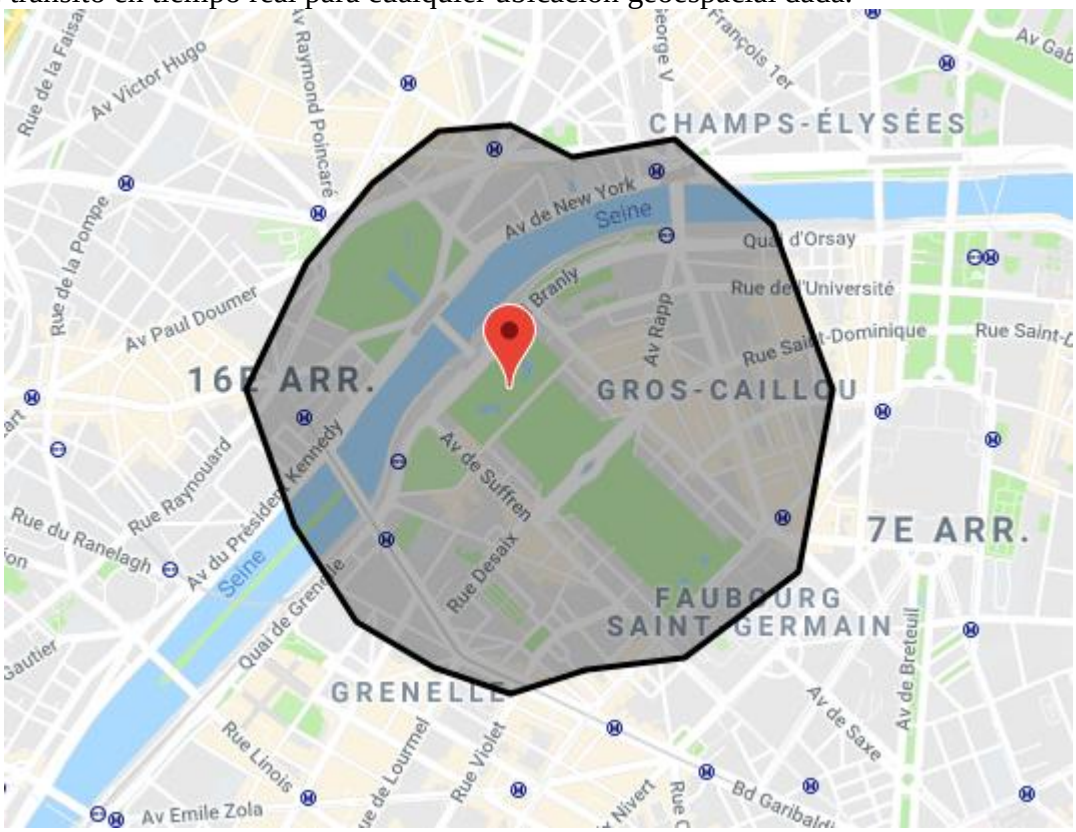
Elaboración de Isócronas con datos de Google Maps

Introducción:

Una Isócrona es un polígono espacial que representa en un mapa, una distancia máxima a recorrer desde un punto, en cualquier dirección y en un período de tiempo determinado. Es decir cualquier punto en el contorno de la isócrona de debe tomar un total de x minutos de viaje para llegar desde el origen y cualquier punto dentro del contorno debe ser alcanzable en menos de x minutos.

Basado en el software codificado por Drew Fustin, científico líder en datos de SpotHero. Drew escribió un módulo de Python que cálcula isócronas para el transporte privado usando datos de la API de Google Maps. El mismo fue adapatdo y expandido para el transporte público utilizando datos de tránsito en tiempo real para cualquier ubicación geoespacial dada.

1 -



Generar un acceso a las API's a usar

Si no se tiene acceso a una API de trabajo, lo primero que se debe hacer es obtener un proyecto de desarrollo de Google. Cree una cuenta de Google cloud y haga clic en "Crear proyecto" y asígnele un nombre. Una vez creado, puede hacer clic en el nombre de su proyecto, y esto lo llevará a una pantalla con una lista que contiene las "API's y autenticaciones" con submenus que contienen a "API's "Y" Credenciales ". Diríjase a "API's"y haga clic en "más" y luego en "API de Google Maps". Para este proyecto, utilizaremos la"API de geocodificación" y la"API de matriz de distancias", así que haga clic en cada uno de ellos y en "Habilitar API". Finalmente, vaya al subtítulo "Credenciales", "Crear nueva clave", "Acceso público a la API", "Clave del navegador" y luego en "Crear".

Después de esto su proyecto estará configurado con Google. Usará su clave de API que ahora se encuentra en la lista para hacer un seguimiento de todo su uso, para asegurarse de que no está superando los límites de solicitud de datos (la API de matriz de distancia permite solo 100 elementos cada 10 segundos y 2.500 elementos por día en su versión gratuita). Debido a esto a

veces se requiere disminuir la velocidad del código.

El módulo de Python accede a las credenciales desde un archivo externo. Cree un archivo llamado 'google_maps.cfg' que contendrá su clave API (o ID de cliente y Clave de cifrado, si están utilizando una API paga). Con el siguiente formato:

[api]
api_number = <INSERTAR AQUI SU CONTRASEÑA DE API>
client_id =

2 - Usando la API de Google Maps

API de geocodificación

La API de geocodificación se usa para transformar una dirección en su correspondiente par [lat, lng]. El uso de la API es tan simple como crear una URL y luego leer el JSON (o XML) que esta devuelve. En este caso, el modulo reemplazará cualquier espacio en la dirección con un "+". Si se está utilizando una API gratuita, agregar a esta cadena '& key =' al final del prefijo "https://maps.googleapis.com/maps/api/geocode/json?".

Si se usa una cuenta de bussines de Google Maps, esto resulta más complicado. La seguridad requiere que se genere una firma cada vez que envíe una solicitud, y esto se debe hacer con la codificación Base64 y el algoritmo HMAC-SHA1.

Una vez completa la URL se envía la solicitud, e interpreta el JSON que la misma devuelve obteniendo el geocódigo [lat, lng]

API de matriz de distancia

La API de matriz de distancia es más complicada porque tiene más parámetros y un retorno JSON más complejo. La API de matriz de distancia devuelve una matriz de tiempos y distancias de viaje dada una lista de orígenes y destinos, con cada elemento de la matriz correspondiente a un par de origen-destino. Para calcular las isócronas, usaremos solo un origen (por lo que este es más un Vector de Distancia), y los puntos de destino serán un conjunto de puntos dispuestos simétricamente en diferentes ángulos alrededor del punto de origen (cada uno de ellos un radio x que varía a medida que avanza el algoritmo).

La API de matriz de distancias toma los orígenes y los destinos como cadenas de dirección, con grupos concatenados con un símbolo (por ejemplo, 'orígenes = Dirección + 1 | Dirección + 2 y destinos = 41.88, -87.62 | Dirección + 3'). En nuestro caso, solo tenemos un origen y un conjunto de destinos que se seleccionan en el algoritmo más adelante. Pueden agregarse ademas varios parámetros opcionales especificados más adelante.

El archivo JSON devuelto tendrá forma de mariz, con "filas" correspondientes a los orígenes y "elementos" correspondientes a los destinos. Para analizarlo usaremos la función 'parse_json' y luego se enviará una consulta en forma de URL para cada elemento, las cuales devolverán una lista en formato de [direcciones, tiempo de viaje], con los tiempos de viaje en minutos.

Parámetros opcionales

Modo: (predeterminado para transporte público): especifica el modo de transporte que se debe utilizar al calcular la distancia.

Restricciones: Se pueden calcular distancias que se adhieran a ciertas restricciones. Las restricciones se indican mediante el uso del parámetro de evitar, y un argumento específico a ese parámetro que indica la restricción. Las mismas incluyen peajes, rutas, ferries y caminos internos. La adición de restricciones no excluye las rutas restringidas, simplemente sesga el resultado a rutas más favorables.

Hora: Se especifica el tiempo en cuyas condiciones de transito se evaluará la isocrona, esta se determina como un entero en segundos desde la medianoche del 1 de enero de 1970 (UTC). Alternativamente, puede especificar un valor de que establece la hora de salida (correcta al segundo más cercano). Si no se especifica, esta predeteminada en el ahora (es decir, la hora actual de ejecución del módulo).

Tipo de modelo de tráfico: especifica las suposiciones que se utilizarán al calcular el tiempo en el tráfico. Esta configuración esta disponible solo para transporte privado pero se menciona en este apartado ya que afecta el valor devuelto en el transporte público, que contiene el tiempo predicho en el tráfico basado en promedios históricos. Los valores disponibles para este parámetro son:

- Mejor Aproximación: (predeterminado) indica que el tiempo de viaje devuelto debe ser la mejor estimación dado lo que se sabe sobre las condiciones de tráfico históricas y el tráfico en ese momento. El tiempo se vuelve más exacto cuanto más cerca de el tiempo de partida.

- Pesimista: indica que el tiempo de viaje debe ser más largo en la mayoría de los días, aunque las condiciones de tráfico particularmente malas pueden exceder este valor.

- Optimista indica que el tiempo de viaje debe ser más corto que en la mayoría de los días, aunque las condiciones particularmente buenas pueden ser menores que este valor

Preferencia de modo: Especifica uno o más modos preferidos de transporte público. El parámetro soporta las siguientes opciones:

- Autobús: indica que la ruta calculada debe preferir viajar en autobús.

- Metro: indica que la ruta calculada debe preferir viajar en metro.

- Tren: indica que la ruta calculada debe preferir viajar en tren.

- Tranvía: indica que la ruta calculada debe preferir viajar en tranvía y tren ligero.

Preferencias: especifica preferencias para solicitudes de tránsito. Usando este parámetro, puede sesgar las opciones devueltas, en lugar de aceptar la mejor ruta predeterminada elegida por la API. Este parámetro solo se puede especificar para solicitudes en las que el modo sea tránsito. El parámetro soporta los siguientes argumentos:
Menos caminata: indica que la ruta calculada debe preferir cantidades limitadas de caminata.
Menos Transferencia: indica que la ruta calculada debe preferir un número limitado de transferencias.

Nota: Las rutas para caminar y andar en bicicleta a veces no incluyen rutas peatonales o para andar en bicicleta, por lo que estas respuestas mostrarán advertencias en el resultado devuelto que debe mostrar al usuario.

3 - Selección de destinos mediante geometría esférica

Dada la dirección de origen (que se traduce en un par [lat, lng] mediante la función de dirección de geocodificación), debemos seleccionar una serie de direcciones de destino que se elegirán encontrando x puntos distribuidos simétricamente (en ángulo) alrededor del punto de origen. Para cada punto de destino alrededor del origen (que está definido por un radio x y distancia y), debemos calcular su [lat, lng] correspondiente. Esto se logra al usar el semiverseno en lugar de la distancia pitagórica simple en el espacio euclidiano (ecuaciones de Haversine). Usando el radio de la Tierra y convirtiendo el origen [lat, lng] y el ángulo x en radianes, se obtiene la latitud y la longitud del destino.

En python, esto utiliza las siguientes operaciones del math: import cos, sin, tan, sqrt, pi, radianes, grados, asin, atan2

4 - Cálculo de isócronas.

Una vez realizados los pasos anteriores tendremos una dirección de origen que hemos geocodificado en [lat, lng], un número x de ángulos para los que podemos calcular los pares de destino y un determinado tiempo de viaje. El objetivo del algoritmo constructor de isócronas es realizar una búsqueda binaria en el radio a lo largo de cada ángulo hasta que cumpla con las condiciones establecidas. La API de matriz de distancias nos permite realizar una búsqueda a lo largo de cada ángulo con solo una consulta (siempre que nos mantengamos dentro de los límites de tamaño de salida, por ejemplo, 100 elementos por consulta). Esto reduce en gran medida nuestras solicitudes y nos permite mantenernos dentro del límite de consultas permitido.

El algoritmo para el cálculo es el siguiente:

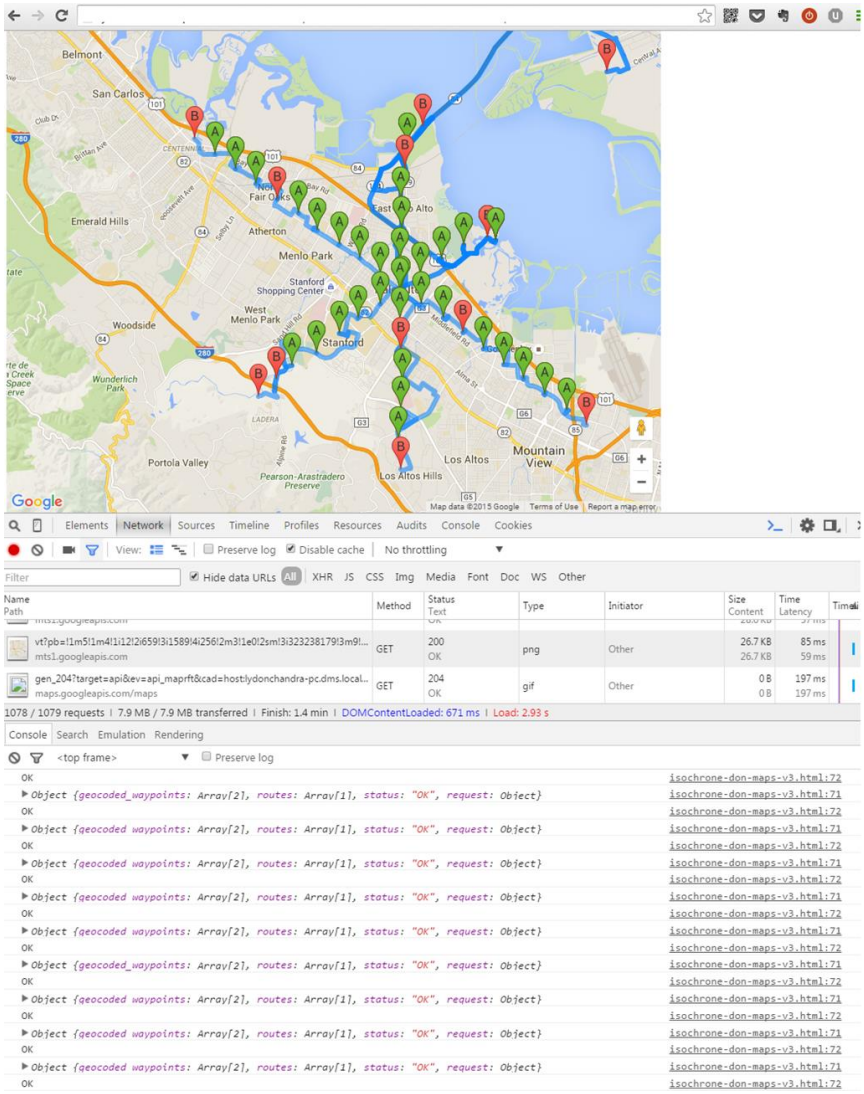
Comienza con un radio inicial ‘rad1’ que es una lista de longitud x, cada elemento es una ‘duration’ / 12 (donde ‘duration’ corresponde al tamaño de los minutos de la isócrona).

Para cada ángulo distribuido simétricamente alrededor del origen (almacenado en la lista "phi1"), se calcula el par de destino [lat, lng] utilizando el método descrito anteriormente y las guarda en una lista"iso"; esta será una vez termine el algorimo la isócrona.

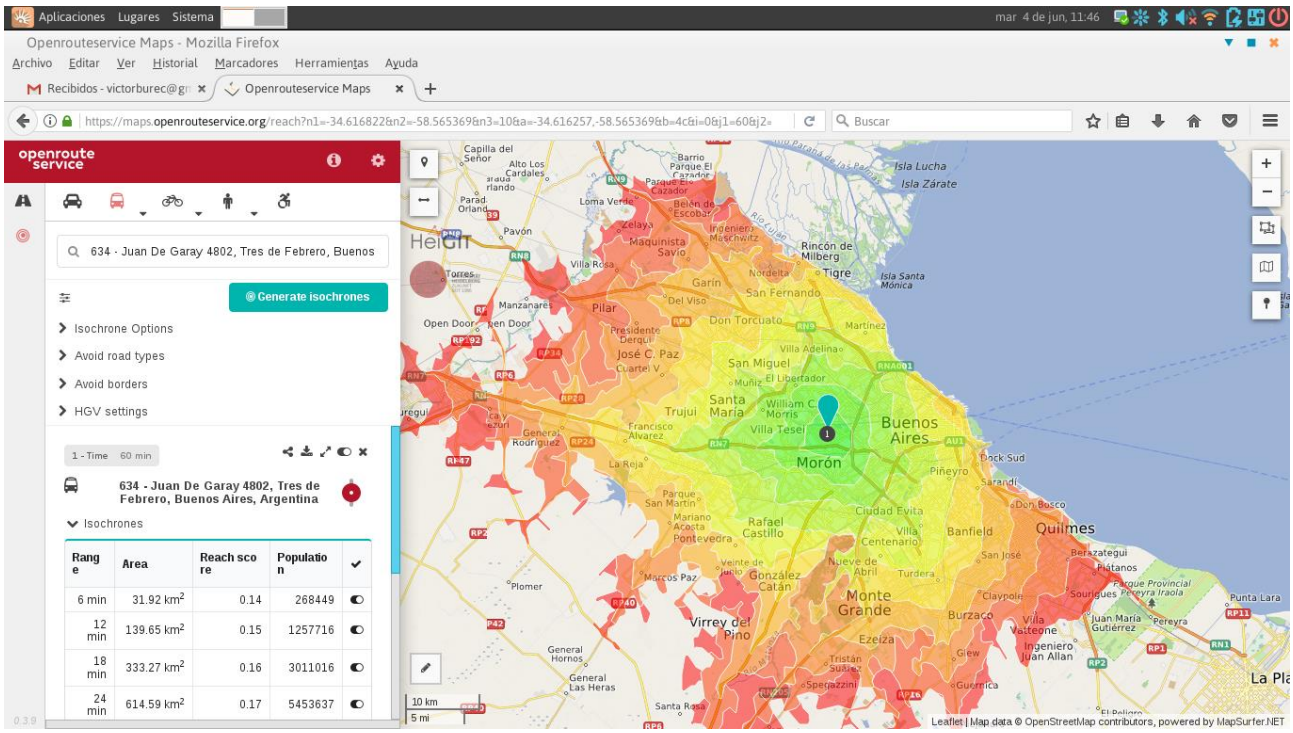
Se crea una URL usando la dirección de origen o geocodificación y los pares de destino almacenados en "iso" y los pasa a la API de la Matriz de Distancia. Analiza el JSON y devuelve las duraciones a cada punto.

Para cada punto x, si el tiempo de viaje devuelto por la API es mayor que el buscado (es decir el "tiempo de viaje" más la "tolerancia", ya que nunca será exactamente igual a la buscado), establece un componente x al ángulo 'rad1'. Si el tiempo de viaje devuelto es mayor o menor que el deseado por la isócrona el componente es cero. Si la duración del tiempo de viaje devuelta por la API para el componente está dentro de la "tolerancia" de "duración" de la isócrona, esta es aceptada y se prosigue a la siguiente.

Esto continuará hasta que los componentes de ‘rad1’ sean adecuados o en su defecto hasta que transcurra demasiado tiempo (para que lo bucles “while” no continúen hasta el infinito).



5 - Iteratividad del programa



Debido a la naturaleza de la expresión de la isocrona, la misma permite la representación iterativa apilable, debido a que cada una de las sucecivas capas abarca a la anterior, permitiendo representar multiples distancias máximas, basadas en variados tiempos de viaje. Esto resulta de gran utilidad a la hora de hacer gráficos, ya que permite integrar múltiples isocronas en un único medio. Para realizar esto con nuestro módulo deberemos efectuar sucecivas mediciones cambiando el tiempo máximo de viaje, descargarlas como polígonos “.kml” e integrarlas mediante un sofware de georepresentación (por ejemplo “QGIS”) de manera externa.

Cambiar parámetros de medición:

(en la línea 376 del códigodel módulo)

```
def generate_isochrone_map(origin='ORIGEN',
 duration='TIEMPO DE VIAJE A REPPRESENTAR',
 number_of_angles=12,
 tolerance=0.1,
 access_type='personal',
 config_path='config/'):
```

Nota: Se recomienda aumentar el número de ángulos bajo la siguiente tabla, modificar hasta tener un resultado óptimo.

Tiempo de viaje	N° de ángulos
5 min	6
15 min	12
30 min	24
1 hora	36

Una vez ejecutado el modulo de la isocrona se deberá copiar la URL que aparece en la opción de “enviar enlace para mail”, pegarla en la barra de direcciones, añadiendo al final esta serie de

caracteres “&output=kml” y presionar ENTER. Aparecerá un cuadro de diálogo de descarga para guardar el KML donde se desee. Repetir el proceso hasta lograr el número de capas deseadas. Finalmente se pueden superponer desde cualquier herramienta de visualización de mapas llamese Qgis, ArcMap, etc.