METRO JAPÓN

Grupo 8

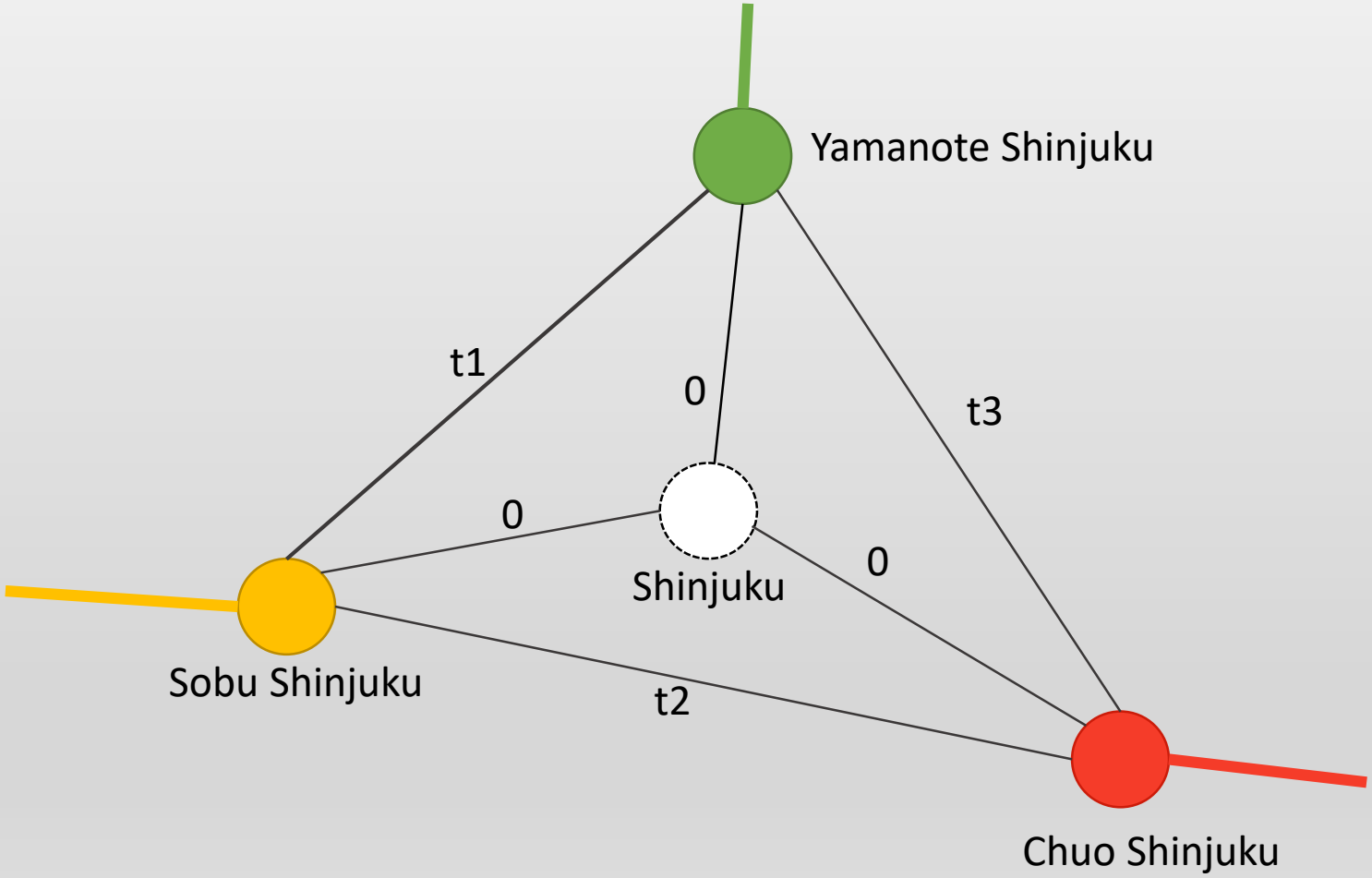
Diego José Abengózar Vilar, Alejandro García Castellanos, Jaime Vallejo Benítez-Cano, Alejandro Gil Ferrer, Ignacio Encinas Ramos

Modelización

Modelo

- Cada estación es un nodo en el grafo.
  - Si una estación tiene varios transbordos, entonces se añadirán tantos nodos como líneas partan de dicha estación
- Las aristas modelizan la línea de metro y sus pesos son el tiempo medio de ir de un punto (nodo) a otro.

Caso particular: Nodos con transbordos



- Ponemos a cero el tiempo entre el nodo simbólico y los nodos de los transbordos para no influir al algoritmo por qué línea empezar.
- Antes de la ejecución se eliminaran todos los nodos simbólicos que no sean el origen y el destino.

Obtención de datos

Paso 1: obtener las distancias entre estaciones

Información obtenida de Wikipedia:

- https://en.wikipedia.org/wiki/Yamanote_Line
- https://en.wikipedia.org/wiki/Ch%C5%AB%C5%8D_Line_(Rapid)
- https://en.wikipedia.org/wiki/Ch%C5%AB%C5%8D-S%C5%8Dbu_Line

Line name	No.	Station	Japanese	Distance (km)	
				Between stations	Total
Yamanote Line	SGW JY 25	Shinagawa	品川	from Tamachi 2.2	0.0
	OSK JY 24	Ōsaki	大崎	2.0	2.0
	JY 23	Gotanda	五反田	0.9	2.9
	JY 22	Meguro	目黒	1.2	4.1
	EBS JY 21	Ebisu	恵比寿	1.5	5.6
	SBY JY 20	Shibuya	渋谷	1.6	7.2
	JY 19	Harajuku	原宿	1.2	8.4
	JY 18	Yoyogi	代々木	1.5	9.9

Obtención de datos

Paso 2: obtener los tiempos y distancias entre transbordos

Información obtenida de las apps Train Info de JR-East y de Google Maps

- Usamos la velocidad media de 5km/h de una persona andando para obtener la distancia

Ref: https://es.wikipedia.org/wiki/Kil%C3%B3metro_por_hora

Obtención de datos

Paso 3: obtener los tiempos medios entre estaciones a partir de los datos previos y haciendo uso de información adicional de la Wikipedia de cada estación

- Tiempo entre estaciones:

$$\left(\frac{\text{distancia calculada en paso 1}}{\text{velocidad media del tren en dicha línea}}\right) + \text{tiempo medio de espera de un tren en una estación hasta que parte}$$

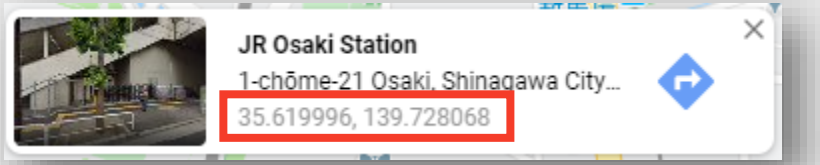
- Tiempo de transbordo:

$$\text{tiempo calculado en el paso 2} + \text{tiempo medio entre trenes en dicha línea}$$

Heurística

Tomamos como heurística el tiempo medio que tardaría el tren más rápido (el que tiene una velocidad de 100km/h) en ir en línea recta desde una estación a otra.

- Obtención de la distancia en línea recta:
  - Primero obtenemos con Google Maps las coordenadas decimales de todas las estaciones.



  - Segundo usamos al fórmula del Haversine para calcular la distancia entre puntos en una esfera

$$a = sin^2(\frac{\varphi_B - \varphi_A}{2}) + cos(\varphi_A) * cos(\varphi_B) * sin^2(\frac{\lambda_B - \lambda_A}{2})$$
$$c = arctan2(\sqrt{a}, \sqrt{1-a})$$
$$d = R * c$$

$$Donde:$$
$$\varphi_i \rightarrow latitud$$
$$\lambda_i \rightarrow longitud$$
$$R \rightarrow radio\ de\ la\ Tierra$$

Ref: https://community.esri.com/groups/coordinate-reference-systems/blog/2017/10/05/

Heurística

- La distancia entre los puntos es menor o igual a la distancia del trayecto del tren entre estaciones
- La velocidad usada es mayor o igual a la velocidad media en el trayecto entre estaciones o transbordos

$\Rightarrow h(n) \leq h^*(n) \Rightarrow$

Nuestra modelización es válida para un Algoritmo A^*

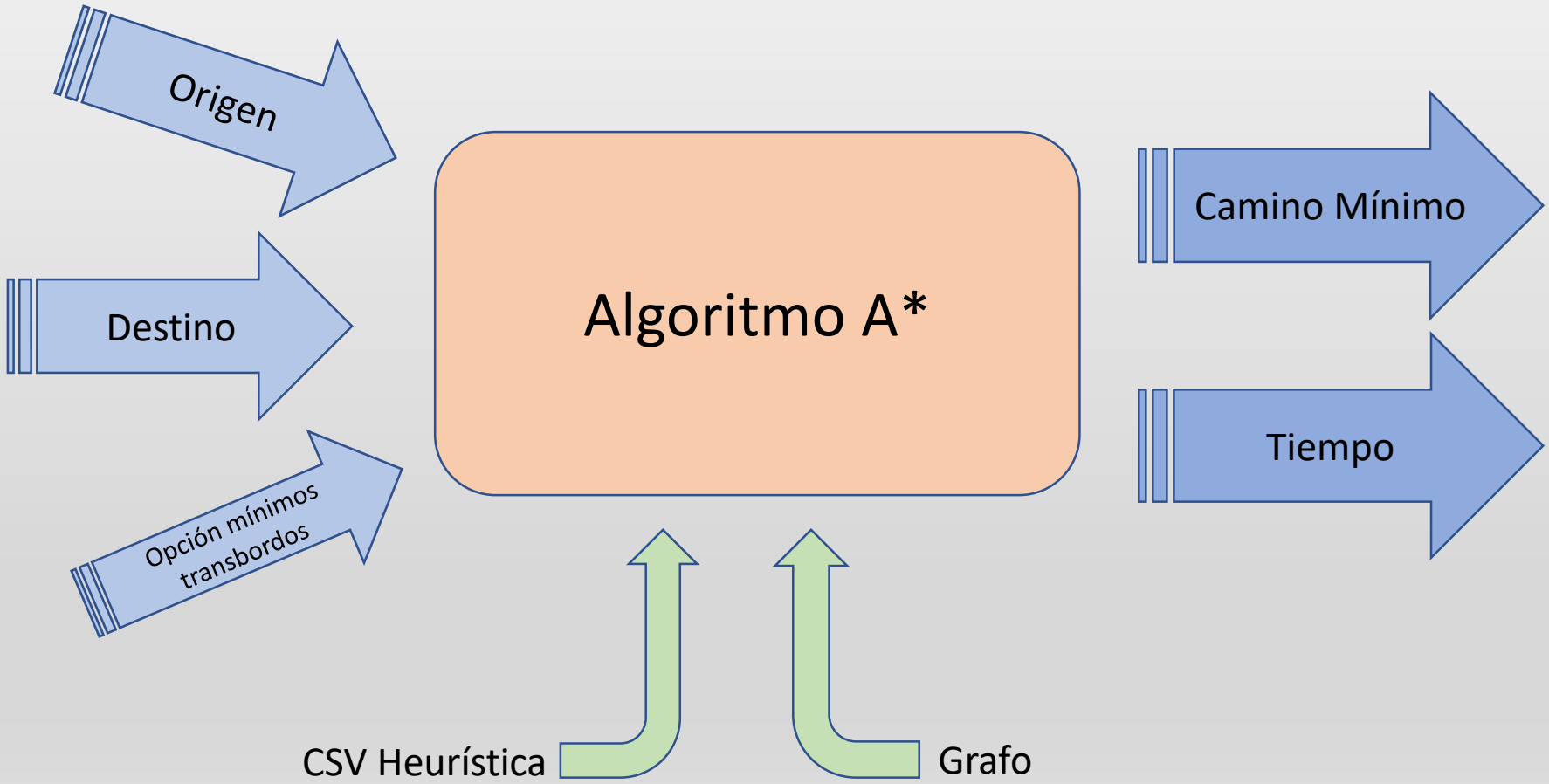
Implementación

¿Por qué hemos elegido Python?

- Versátil
- Muchas librerías útiles
  - TKinter
  - MatPlotLib
  - NetworkX
  - CSV
- Facilidad para testear/depurar
- Facilidad para la interfaz gráfica
- Aprender a hacer un proyecto en Python

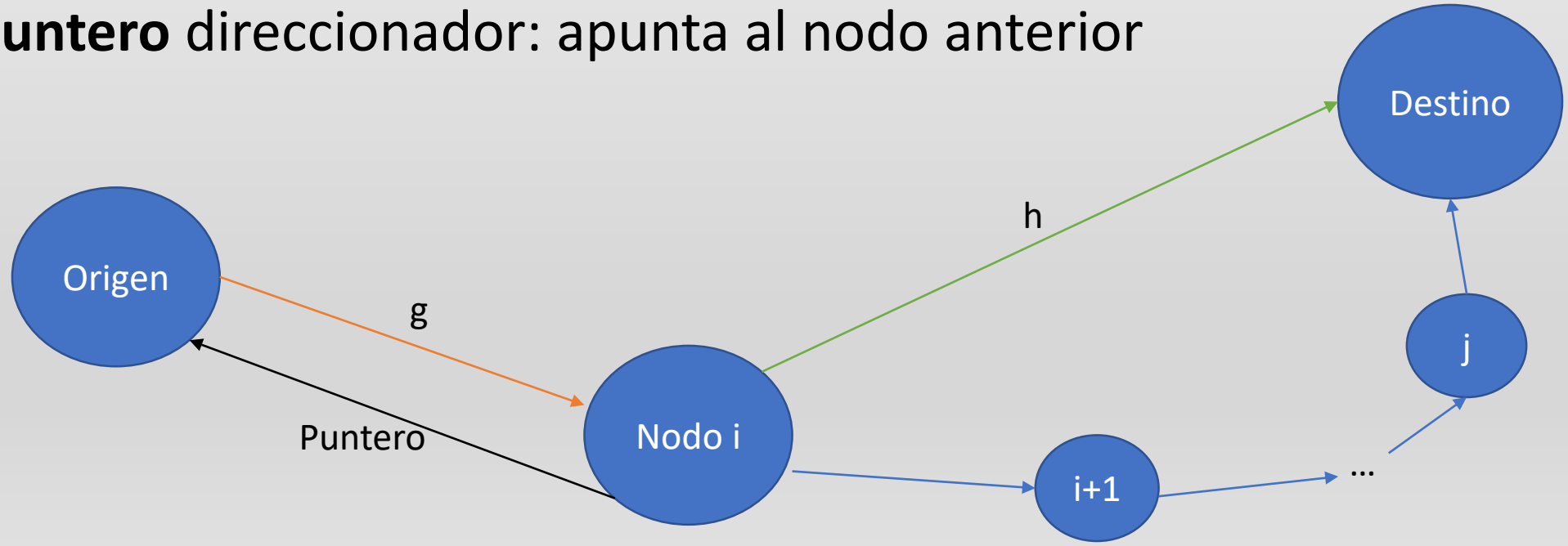
Algoritmo

Algoritmo A*


Origen
Destino
Opción mínimos transbordos
Algoritmo A*
Camino Mínimo
Tiempo
CSV Heurística
Grafo

Algoritmo A*

- Función **g**: tiempo desde el origen hasta el nodo actual
- Función **h**: heurística (estimación de tiempo hasta el nodo destino)
- Función **f**: g + h
- **Puntero** direccionador: apunta al nodo anterior



Algoritmo A*

- **Grafo G:** implementado con la librería **networkX**

- **Lista Abierta**: priority queue implementada con la librería **heapq**

  Guarda duplas **< f , estación >**

  Menor f → Más prioridad

- **Lista Cerrada**: lista que almacena los nodos ya visitados

Algoritmo A*

Inicialización

Mientras (no *HemosLlegado*)

Sacar el mas prioritario de *ListaAbierta* y meterlo en *ListaCerrada*

Si es el destino → *HemosLlegado*

Si no, para cada *nodo adyacente de G* (que no sea antecedente)

→ Calcular su f y g (y añadir penalización a transbordos)

Algoritmo A*

Si está en *ListaAbierta*, si el *nuevo f* es mejor
→ Actualizar *f, g, el puntero direccionador y ListaAbierta*

Si no está
→ Actualizar *f, g, el puntero direccionador* y añadirlo a *ListaAbierta*

Volver

Calcular Ruta usando los punteros direccionadores

Interfaz Gráfica

Interfaz

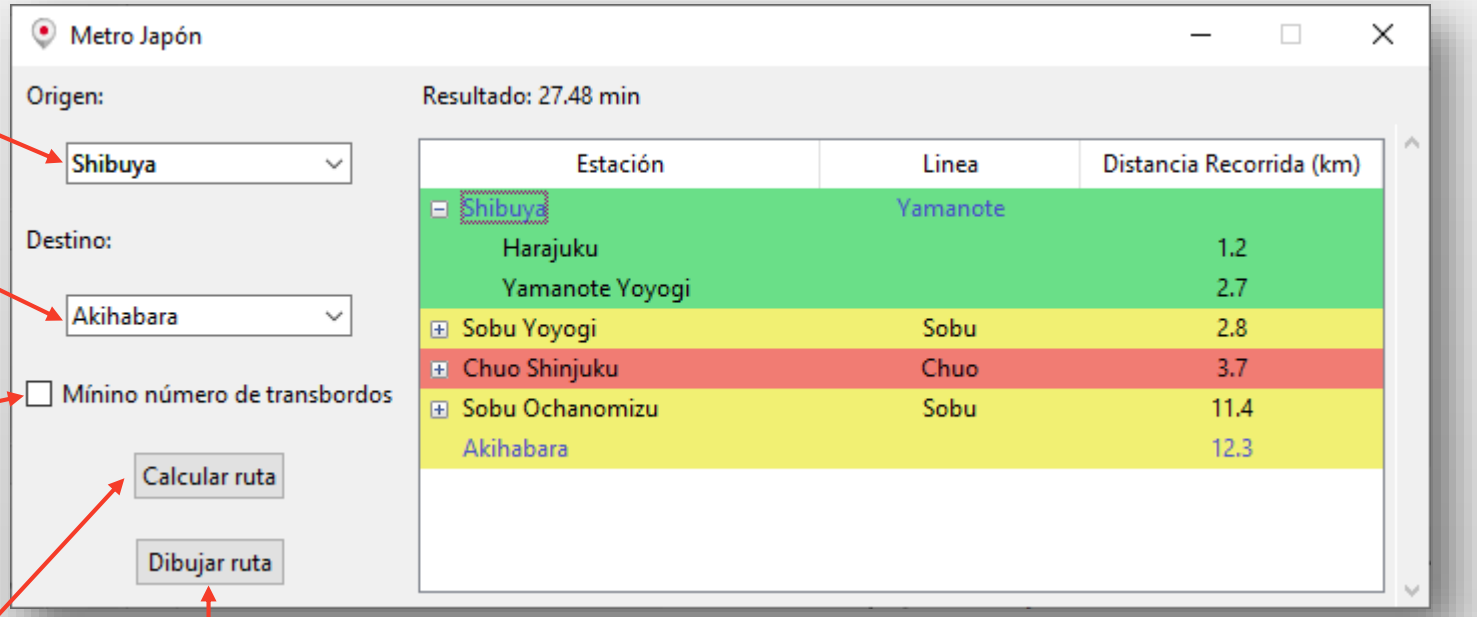
Seleccionar Origen

Seleccionar Destino

Opción de mínimo numero de transbordos: busca el menor camino con el menor numero de transbordos posibles

Calcular y mostrar lista de estaciones del camino mínimo: Con su correspondiente línea y distancia recorrida

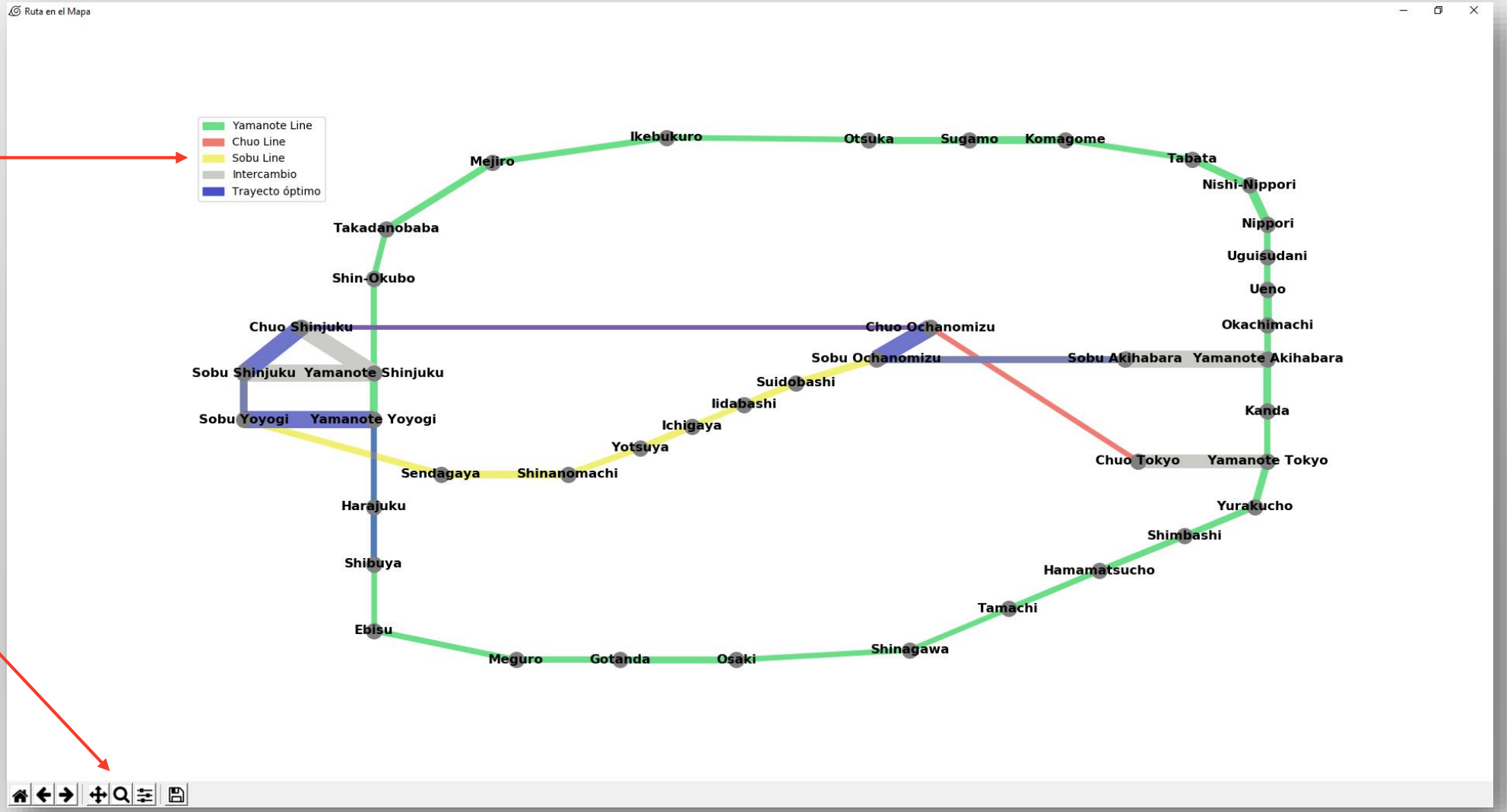
Dibujar la ruta sobre el mapa



Interfaz

Leyenda

Opciones



Demostración del funcionamiento

Coger ejemplos buenos: Ueno – Shibuya, Ueno – Meguro, etc