Artificial Intelligent Commuting Agent

Lee Jui Chi
26001803777
ISSE

1. Introduction

This project is about an Artificial Intelligent agent for presenting my daily commuting to Ritsumeikan University. It is expected to make decisions automatically, such as which train to catch or which bus should ride. The agent is written in Python, it can find the fastest route and the route that save the most money.

2. Proposed Solutions

a. Environment: route of trains

Since the distance between my home and the school is quite far, and I don't have a car, so I commuted by train and buses all the time. Figure 1 shows the environment flowchart(the potential routes to go to the school). For the type of environment, in this case, it is **fully observable, discrete, stochastic, single-agent, and static.**

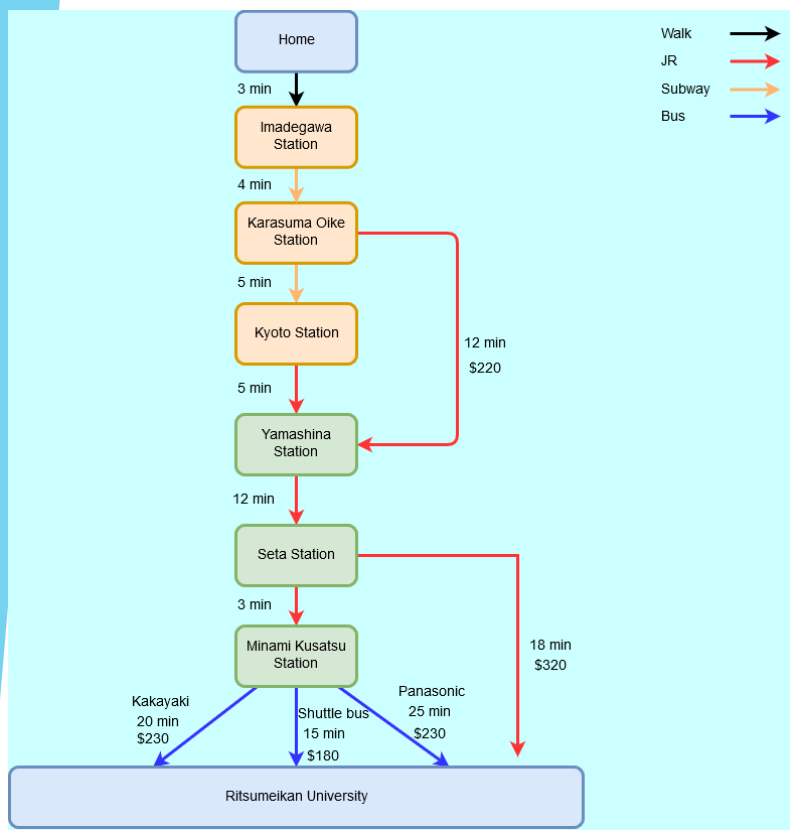

Figure 1

b. Agent PEAS specification

- Performance measurement – time cost, money cost
- Environment – JR, subway, bus, roads, transport fee
- Actuators – Screen display, OpenStreetMap(display the route on the map)
- Sensors – Keyboard, Python console app (take input)

c. Agent knowledge base

Figure 2 represents knowledge base of the agent by Semantic Networks

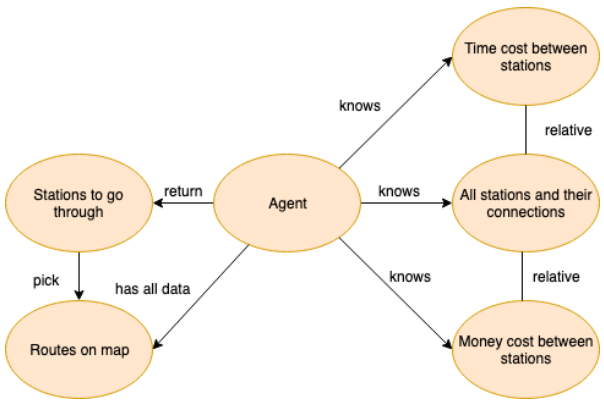

Figure 2

Knowledge base (KB) inference logic:

- KB $|\text{-}_{\text{Dijkstra}}$ routeFound
- Dijkstra(start, end, graph) ⇒ stationsToGoThrough
- stationsToGoThrough ⇒ routeFround
- $KB \models$ routeFound ⇒ KB $|\text{-}_{\text{Dijkstra}}$ routeFound

d. Dijkstra Path-Finding Algorithm

To find the fastest route and the route which saves the most money by the agent, I need a searching algorithm to traverse the graph and find the most effective route. To solve this, I build a graph class and applied the Dijkstra algorithm to implement the graph search. Time Complexity of the implementation is O(n^2)

3. Prototypes

Besides defining the knowledge and giving the data to the agent, the agent also needs a class for the graph, a function for the Dijkstra algorithm, a function for drawing the map, and a function for taking the user's input. After knowing the user's requirement, the agent creates a graph and finds the path by the algorithm. Next, returning the path it founds, then draw the route on the map and saving it into an HTML file

4. Tests

To test the agent and see if it is making the right decisions, I manually calculate the time costing by myself first and then run the program. By comparing my result and the agent's returning route, I found that the returning result of the route is correct, and the map was also drawn successfully. Similarly, the test for finding the route that save most money was also succeeded. Fattest route and save money route are different which shows that the agent is really making effect for making decisions. Figure 3 shows the stations that fattest route go through and the minutes it cost (47); figure 4 shows the map of it.

```
Juis-MacBook-Pro:project juichilee$ python3 project.py
Enter 1 for fatest route, enter 2 for save money route:1
The route are drawn on the map, please open route.html in results folder to see the result
Fatest route:
(['home', 'imadegawa', 'karasuma', 'kyoto', 'yamashina', 'seta', 'seta_local_bus', 'ritsumeikan'], 47)
```

Figure 3

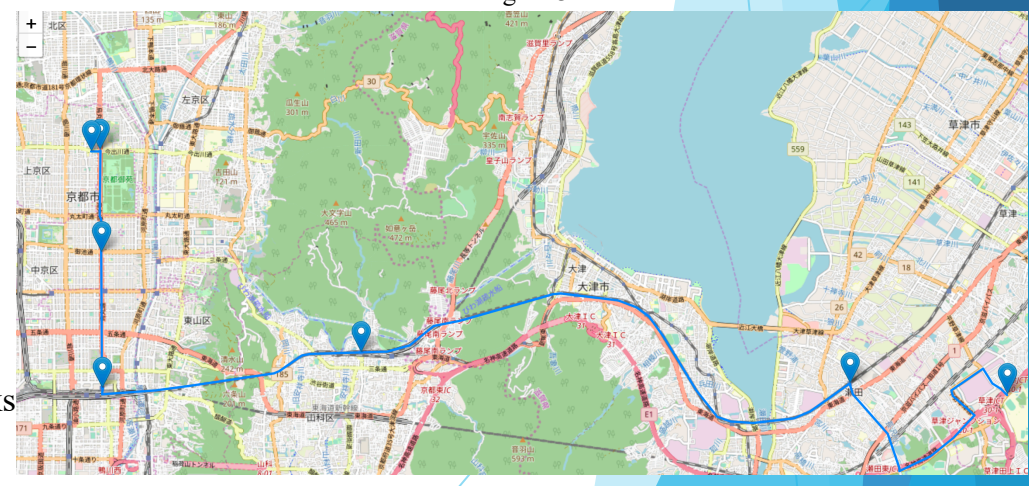

Figure 4

5. Conclusion

This project only achieves to have the prototype of this agent, it still can have a lot of potential. For example, we can consider the waiting time for trains and buses if we have the time table of them. To sum up, by using the Dijkstra algorithm, the agent can find the shortest path (considering time or money) successfully. If the agent can have more knowledge and data, it can be really useful for navigation.