Aleksander Woźniak

Sprawozdanie AI lista 1

Link do repo: https://github.com/0leslaw/l1

Pierwszym krokiem do rozwiązania każdego z zadań było załadowanie grafu. Osobiście, rozwiązałem ten problem z pomocą biblioteki networkX. Potraktowałem każdy przejazd jako krawędź MultiDiGraph-u i obsłużyłem połączenie przystanków bardzo bliskich w jeden wierzchołek grafu oraz sanityzacje pól czasowych.

Z pozyskanego grafu, który odtąd dla łatwiejszej pracy przechowywany był jako pickle file, wyciągnąłem informacje na drodze eksploracyjnej analizy. Okazało się, że między niektórymi przystankami jest oznaczony zerowy czas przejazdu, co oznacza, że nie da się zbudować sensownej heurystyki w A* (czyli albo pomocnej, albo kompletnej) co jednak nie przeszkadza w jej stosowaniu.

Zad1.

Rozwiązane przy pomocy algorytmów Dijkstry oraz A* dla minimalizacji kosztu poruszania się w grafie. Koszt został zdefiniowany w osobnej funkcji jako ilość przesiadek lub czas podróży, dobierane w zależności od preferencji użytkownika. Te algorytmy wykorzystują fakt przechowywania w wierzchołkach ostatniej najlepszej krawędzi wchodzącej, a także funkcję sprawdzającą dla niej obecny czas, który następnie jest traktowany jako starting point przy kolejnej iteracji. Cała magia tego rozwiązania jest taka, że jeśli iteracyjnie dobieramy obecnie najlepsze rozwiązania (o najdłuższej drodze lub w przypadku A* najkrótszej drodze i najmniejszej odległości od celu) to jesteśmy w stanie „kierunkować” przeszukiwanie naszego grafu. Przeszukiwanie można zoptymalizować przez priorytetyzowanie najlepszych rozwiązań oraz binarne przeszukiwanie krawędzi w celu znalezienia najbliższych obecnego czasu (heapq).

Zad2.

Jest rozwiązane przy użyciu Tabu Search, w którym dla każdej iteracji sprawdzamy koszt obecnej ścieżki jako sumę kosztów przejazdów sekwencyjnych między przystankami i aktualizujemy najlepsze rozwiązanie. Funkcja tabu search ujawnia się przy wybieraniu dwóch losowych wierzchołków do odwrócenia, tak odwrócone wierzchołki trzymamy w liście, z której je wyrzucimy jeśli okaże się że to rozwiązanie jest gorsze, a zostawimy, jeśli będzie lepsze. Warto wspomnieć o tym, że z kolejki odwróceń odrzucamy po czasie pary, aby zachować różnorodność