|  |  |
| --- | --- |
| Mateusz Woźniak 260292 | 03.04.2023 |

**Sztuczna inteligencja i inżynieria wiedzy – Raport lista 1**

Cel:

Celem ćwiczenia jest praktyczne zapoznanie się z problemami optymalizacyjnymi oraz praktyczne przećwiczenie omawianych na wykładzie metod rozwiązywania pewnej podklasy tych problemów. Po wykonaniu listy, student powinien wiedzieć czym jest problem optymalizacyjny, jakie trudności mogą się wiązać z uzyskaniem dokładnego rozwiązania przedstawionego problemu oraz jak poradzić sobie z rozwiązaniem problemu przy ograniczonych zasobach (np. moce obliczeniowe, czas). W szczególności znane powinny być różnice między aproksymacją a heurystyką oraz przykłady podejścia heurystycznego do problemu przeszukania, oraz znajdowania ścieżek.

Dane do wykonania ćwiczenia:

Plik *connection\_graph.csv*:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Którego kolumny oznaczają kolejno:

1. id przejazdu
2. company – przewoźnik
3. line – nazwa linii
4. departure\_time – czas odjazdu (hh:mm:ss)
5. arrival\_time – czas przyjazdu (hh:mm:ss)
6. start\_stop – nazwa przystanku początkowego
7. end\_stop – nazwa przystanku końcowego
8. start\_stop\_lat – szerokość geograficzna przystanku początkowego
9. end\_stop\_lat – szerokość geograficzna przystanku końcowego
10. start\_stop\_lon – długość geograficzna przystanku początkowego
11. end\_stop\_lon – długość geograficzna przystanku końcowego

Informacje na temat projektu:

1. język programowania – Python
2. implementowane algorytmy – algorytm Dijkstry, algorytm A\*
3. struktura plikowa:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Dane są serializowane do pliku *graph.pkl,* aby uniknąć ich wielokrotnego przetwarzania. Graf jest reprezentowany za pomocą listy węzłów (przystanków), w których zawarte są listy sąsiedztw (listy odjazdów).

Obraz zawierający tekst

Opis wygenerowany automatycznie

Algorytm Dijkstry

Algorytm Dijkstry to popularny algorytm służący do znajdowania najkrótszej ścieżki między wierzchołkami w grafie ważonym. Na podstawie iteracji, na bieżąco poprawiane są najkrótsze ścieżki od początku drogi do pozostałych wierzchołków.

W implementacji tego algorytmu skorzystałem z kolejki priorytetowej, a po ukończeniu iteracji, droga została odtworzone idąc wstecz wzdłuż przystanków na podstawie słownika zawierającego min. poprzedni przystanek.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Algorytm A\*

Algorytm ten rozszerza algorytm Dijkstry o heurystykę, która zwiększa efektywność przeszukiwania grafu. W wyżej wspomnianej kolejce priorytetowej, kolejka jest ustawiana na podstawie sumy kosztu oraz wartości heurystyki.

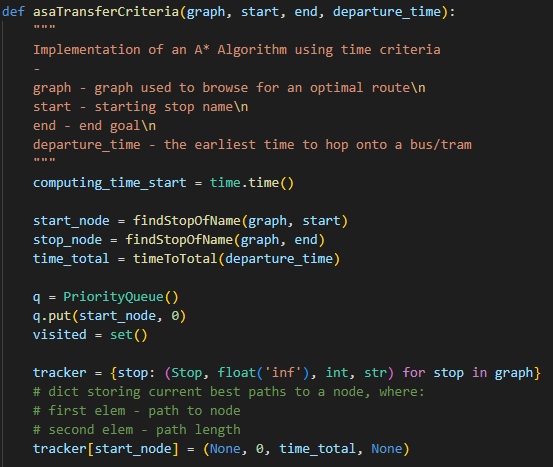
W zadaniu rozważane były 2 przypadki – kryterium przesiadkowe, a także kryterium czasowe, które kładły większy nacisk na odpowiednie wartości. W przypadku kryterium przesiadkowego aplikowano dodatkowe kary w wysokości 10 za przesiadkę.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie



Obraz zawierający tekst

Opis wygenerowany automatycznie

Heurystyka oraz kara związana z przesiadką:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Heurystyka jest zawarta w klasie Stop

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obie funkcje są zawarte w klasie Departure

Wypisanie wyników:

Obraz zawierający tekst, w pomieszczeniu, zrzut ekranu

Opis wygenerowany automatycznie

Modyfikacja:

Usunąłem z przeglądania powtarzające się połączenia (odjazdy tych samych linii w tym samym kierunku, lecz o innych godzinach) pozostawiając jedynie najwcześniejsze z nich.

Przed:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Czas wykonywania obliczeń - 9.529773235321045s

Po:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Czas wykonywania obliczeń - 0.560560941696167s

Kod

Pozostała część kodu dostępna jest w repozytorium:

<https://github.com/matto0O/Artificial-Intelligence/tree/master/Lista%201>

Problemy i wnioski

Największym problemem okazała się implementacja poprawnie działającej wartości heurystyki, gdyż bez odpowiedniego mnożnika, może dojść do niepotrzebnego przeszukiwania dodatkowych węzłów, zaś przy zbyt dużej wartości, otrzymujemy sytuację, gdy algorytm szuka najkrótszej, a nie najszybszej drogi. Poza tym algorytm A\* znacząco przyspieszył działanie programu, bo o około 8%.