

Algorytmy przeszukania w problemach transportu komunikacją miejską

Mikołaj Jastrzębski

Marzec 2024

Spis treści

1	Opis problemów	2
2	Wykorzystane biblioteki	2
3	Analiza i przygotowanie danych	2
3.1	Analiza danych	2
3.2	Przygotowanie danych	3
4	Utworzona struktura danych	3
5	Dane testowe	3
5.1	Problem połączeń komunikacji	4
5.2	Problem komiwojażera	4
6	Usprawnienia dokonane w rozwiązaniach	4
7	Algorytm Dijkstra	4
7.1	Wprowadzenie	4
7.2	Wyniki	5
8	Algorytm A*	5
8.1	Algorytm A* o kryterium czasu	6
8.1.1	Heurystyka	6
8.1.2	Wyniki	6
8.2	Algorytm A* o kryterium przesiadek	7
8.2.1	Heurystyka	7
8.2.2	Wyniki	7
9	Porównanie wyników dla pierwszego problemu	8
9.1	Typowy przejazd	8
9.2	Krótki przejazd	8
9.3	Długi przejazd	8
9.4	Skomplikowany przejazd	8
10	Tabu search	8
10.1	Wprowadzenie	8
10.2	Wyniki tabu seach	9
11	Podsumowanie projektu oraz wnioski	10

1 Opis problemów

Celem projektu jest zapoznanie się z problemami oraz algorytmami optymalizacyjnymi oraz praktyczne wykorzystanie algorytmów Dijkstra, A* oraz tabu search.

Pierwszy problem, który należy rozwiązać polega na znajdowaniu połączeń komunikacji miejskiej między przystankami we Wrocławiu w najlepszy możliwy sposób. Program powinien zwracać harmonogram przejazdu, wypisując w kolejnych liniach informacje o kolejno wykorzystanych przystankach, liniach komunikacyjnych oraz czasach przejazdu. Dodatkowo wypisywane są wartości funkcji kosztu znalezionej odpowiedzi oraz czas obliczeń liczony od wczytania danych do uzyskania rozwiązania.

Drugi problem dotyczy problemu komiwojażera, mamy za zadanie podać listę przystanków komunikacji miejskiej, a następnie odnaleźć najlepszą trasę dzięki której odwiedzimy każdy z tych przystanków finalnie wracając do pierwszego. Na wyjściu powinno znaleźć się czas podróży, czas wykonywania obliczeń programu oraz finalna trasa, którą należy wybrać, aby w jak najkrótszym czasie odwiedzić wszystkie przystanki i wrócić do początkowego.

2 Wykorzystane biblioteki

1. **pandas** - wykorzystana przy obróbce jak i analizie danych.
2. **time** - wykorzystana do pomiaru czasu wykonywania się funkcji
3. **heapq** - wykorzystana do obsługi kolejki priorytetowej w algorytmach djikstra oraz A*
4. **tabulate** - wykorzystana do wyświetlania danych przystanków w zadaniu pierwszym
5. **datetime** - wykorzystana do obliczania różnicy czasów
6. **geopy.distance** - wykorzystana do obliczania odległości między dwoma parami koordynatów

3 Analiza i przygotowanie danych

3.1 Analiza danych

Do projektu wykorzystano plik csv zawierający przetworzone dane z rozkładu jazdy komunikacji miejskiej we Wrocławiu, które można znaleźć na stronie opendata.cui.wroclaw.pl/. Dane wykorzystane do rozwiązania problemu zawierają 996520 rekordów dotyczących przystanków autobusowych jak i tramwajowych. Zawiera on następujące kolumny:

1. **id** - identyfikator krawędzi.
2. **company** - nazwa przewoźnika.
3. **line** - nazwa linii.
4. **departure_time** - czas odjazdu w formacie HH:MM:SS.
5. **arrival_time** - czas przyjazdu w formacie HH:MM:SS.
6. **start_stop** - przystanek początkowy.
7. **end_stop** - przystanek końcowy.
8. **start_stop_lat** - szerokość geograficzna przystanku początkowego w systemie WGS84.
9. **start_stop_lon** - długość geograficzna przystanku początkowego w systemie WGS84.
10. **end_stop_lat** - szerokość geograficzna przystanku końcowego w systemie WGS84.
11. **end_stop_lon** - długość geograficzna przystanku końcowego w systemie WGS84.

Tak bogaty zbiór danych umożliwia kompleksową analizę dostępności i efektywności komunikacji miejskiej we Wrocławiu, z naciskiem na optymalizację tras i czasu podróży między poszczególnymi przystankami.

3.2 Przygotowanie danych

Ogromne zbiory danych zawsze należy dostosować do rozwiązywanego problemu. W tym przypadku dane są bardzo dobrze skonstruowane oraz kompletne, lecz nasze obliczenia różnicy odległości do przystanków mogą zostać uproszczone przez zunifikowanie koordynatów przystanków. W tym celu utworzono krótki skrypt. Skrypt zamienia koordynaty danego przystanku na średnią arytmetyczną koordynatów wszystkich przystanków identyfikujących się tą samą nazwą.

4 Utworzona struktura danych

W pierwszej kolejności utworzono graf składający się z Wierzchołków oraz krawędzi.

Wierzchołki zawierają takie dane jak:

1. **name** - Nazwa przystanku
2. **lat** - Szerokość geograficzna przystanku
3. **long** - Wysokość geograficzna przystanku

W trakcie rozwiązywania problemu w celu ułatwienia implementacyjnego oraz zmniejszenia złożoności grafu ograniczono wierzchołki do posiadania jedynie o nazwie przystanku.

Krawędzie zawierają takie dane jak:

1. **id** - identyfikator krawędzi.
2. **company** - nazwa przewoźnika.
3. **line** - nazwa linii.
4. **departure_time** - czas odjazdu w formacie HH:MM:SS.
5. **arrival_time** - czas przyjazdu w formacie HH:MM:SS.
6. **start_stop** - przystanek początkowy.
7. **end_stop** - przystanek końcowy.
8. **start_stop_lat** - szerokość geograficzna przystanku początkowego w systemie WGS84.
9. **start_stop_lon** - długość geograficzna przystanku początkowego w systemie WGS84.
10. **end_stop_lat** - szerokość geograficzna przystanku końcowego w systemie WGS84.
11. **end_stop_lon** - długość geograficzna przystanku końcowego w systemie WGS84.

Graf składa się ze słownika, którego kluczami są wierzchołki, czyli nazwa przystanku, a wartościami są krawędzie, czyli dane o wszystkich możliwych połączeniach z danego przystanku.

5 Dane testowe

Rozwiązania były testowane zarówno na typowych kursach jak i edge-case'ach, do porównania i ukazania wyników wybrano cztery trasy.

5.1 Problem połączeń komunikacji

Jako typowe połączenie mianowano kurs ze stacji **Kwiska** do stacji **Pl. Grunwaldzki** o godzinie 12:00. Jest to połączenie, które nie wymaga przesiadek oraz zaczyna się w momencie w którym nie trzeba długo czekać na tramwaj.

Najkrótszym połączeniem jest przejazd ze stacji **Pomorska** do stacji **Mosty Pomorskie** o godzinie 9:00. Jest on nietypowy, ponieważ jest wiele tramwajów które przejeżdżają między tymi przystankami, a znajdują się obok siebie.

Najdłuższym połączeniem jest przejazd ze stacji **Gęsia** do stacji **Ołtaszyn** o godzinie 14:36. Stacje te są na dwóch odległych od siebie końcach miasta, które nie są silnie zabudowane (sugeruje to mniejszą liczbę przystanków).

Najbardziej skomplikowanym połączeniem jest przejazd ze stacji **Dworzec główny** do stacji **Pl. Dąbrowskiego** o godzinie 14:30, ponieważ między tymi trasami przejeżdża kolosalna liczba tramwajów jak i autobusów.

5.2 Problem komiwojażera

Jak typowe połączenie wybrałem kurs **Dworzec główny, pl. Grunwaldzki, Pomorska, pl. Bema, Świdnicka**, gdyż są to stacje ułożone w idealnym kole oraz stosunkowo blisko siebie.

Jako najkrótsze połączenie wybrałem trasę **Pomorska, Rynek, pl. Bema, Dubois**, są to bardzo blisko siebie ustawione przystanki, które biegną w jednej linii.

Jako krótkie, ale skomplikowane połączenie wybrałem **Bardzka, Hallera, Górnicza**, przystanki te są daleko od siebie oraz nie są połączone bezpośrednią linią.

Jako najdłuższe oraz skomplikowane połączenie wybrałem **Kwiska, Kozanów, Wolska, Dworzec Główny, Świdnicka**. przystanki te znajdują się daleko od siebie, a co więcej znajdują się w zatłoczonych sferach Wrocławia.

6 Usprawnienia dokonane w rozwiązaniach

Aby usprawnić funkcjonowanie algorytmów zdecydowałem się pomijać przystanki uwcześnie odwiedzane, jeśli ich czas jest wyższy od czasu przy poprzednim odwiedzeniu, dzięki temu dokładność rozwiązań jak i czas pracy programu się zmniejszył.

Kolejnym usprawnieniem jest dodanie pola "weight" do krawędzi grafu, które wynosi czas podróży z przystanku początkowego do przystanku końcowego. Dzięki temu program działa zdecydowanie szybciej, ponieważ nie trzeba co odwiedzenie krawędzi kalkulować dla niej czasu przejazdu.

Następnym miniaturowym usprawnieniem jest wykorzystanie odpowiedniej biblioteki do kalkulowania różnicy odległości między dwoma parami koordynatów, dzięki temu obliczenia są dokładniejsze.

7 Algorytm Dijkstra

7.1 Wprowadzenie

Algorytm Dijkstry to algorytm znajdowania najkrótszych ścieżek w grafie ważonym (o nieujemnych wagach) z jednym źródłem. Algorytm działa poprzez utrzymywanie zbioru wierzchołków o najkrótszej odległości od źródła oraz aktualizację tych odległości wraz z dodawaniem kolejnych wierzchołków do zbioru.

7.2 Wyniki

Journey from KŁESKA to PL. GRUNWALDZKI started at: 9:00:00:
Travel time: 26.8 minutes
Number of changes: 1
Elapsed in: 1.7752 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
KŁESKA	09:01:00	PAŁOWIENSKA	09:02:00	20
PAŁOWIENSKA	09:02:00	NIEOŚCIEŻA	09:04:00	20
NIEOŚCIEŻA	09:04:00	MOCŁAW PRZEDAJĄC (ZACHĘBIA)	09:06:00	20
MOCŁAW PRZEDAJĄC (ZACHĘBIA)	09:06:00	DOŁBĘD	09:07:00	20
DOŁBĘD	09:07:00	PL. STRZEGOMSKI (MUSEUM WOPŁCZESNE)	09:09:00	13
PL. STRZEGOMSKI (MUSEUM WOPŁCZESNE)	09:09:00	PLACOWY TECHNIKÓW	09:10:00	13
PLACOWY TECHNIKÓW	09:10:00	PL. JANA PAWŁA II	09:12:00	13
PL. JANA PAWŁA II	09:12:00	RYNEK	09:14:00	13
RYNEK	09:14:00	ZAPRAWA	09:15:00	13
ZAPRAWA	09:15:00	SŁODKOWA	09:17:00	13
SŁODKOWA	09:17:00	GALERIA DOPINGOWA	09:19:00	13
GALERIA DOPINGOWA	09:19:00	URZĄD MIEJSCOWY (DPMPT)	09:22:00	13
URZĄD MIEJSCOWY (DPMPT)	09:22:00	POST GRUNWALDZKI	09:24:00	13
POST GRUNWALDZKI	09:24:00	PL. GRUNWALDZKI	09:26:00	13

(a) Kwiska do Pl. Grunwaldzki

Journey from POMORSKA to MOSTY POMORSKIE started at: 9:00:00:
Travel time: 3.0 minutes
Number of changes: 0
Elapsed in: 0.0625 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
POMORSKA	09:02:00	MOSTY POMORSKIE	09:03:00	18

(b) Pomorska do Mosty pomorskie

Journey from GĘSIA to OLTASZYŃ started at: 14:36:00:
Travel time: 47.0 minutes
Number of changes: 5
Elapsed in: 2.4320 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
GĘSIA	14:38:00	KŁĘCZYŃSKA	14:40:00	23
KŁĘCZYŃSKA	14:40:00	KŁĘCZYŃSKA	14:41:00	23
KŁĘCZYŃSKA	14:41:00	KROPIERA	14:43:00	23
KROPIERA	14:43:00	POSTY WARSZAWSKIE	14:45:00	A
POSTY WARSZAWSKIE	14:45:00	WYSZYŃSKIEGO	14:47:00	A
WYSZYŃSKIEGO	14:47:00	OGROD BOTANICZNY	14:49:00	A
OGROD BOTANICZNY	14:49:00	KATEDRA	14:51:00	A
KATEDRA	14:51:00	URZĄD MIEJSCOWY (MUSEUM WARSZAWSKIE)	14:52:00	A
URZĄD MIEJSCOWY (MUSEUM WARSZAWSKIE)	14:52:00	POCZTA GŁÓWNA	14:53:00	A
POCZTA GŁÓWNA	14:53:00	SMER KRAKOWSKIEGO	14:55:00	A
SMER KRAKOWSKIEGO	14:55:00	DWORZEC GŁÓWNY	14:58:00	146
DWORZEC GŁÓWNY	15:00:00	DWORZEC AUTOBUSOWY	15:03:00	9
DWORZEC AUTOBUSOWY	15:05:00	BORKOWSKA (AQUAPARK)	15:08:00	K
BORKOWSKA (AQUAPARK)	15:08:00	SŁĘCZNA	15:10:00	K
SŁĘCZNA	15:10:00	DEZAKOMA	15:13:00	K
DEZAKOMA	15:14:00	BORKOWSKA (SZPITAL)	15:16:00	612
BORKOWSKA (SZPITAL)	15:16:00	PRZYSTANOWA	15:17:00	612
PRZYSTANOWA	15:17:00	WOSZYŃSKIE	15:18:00	612
WOSZYŃSKIE	15:18:00	PAWIA JAŁNA	15:20:00	612
PAWIA JAŁNA	15:20:00	SŁET	15:21:00	612
SŁET	15:21:00	OLTASZYŃ	15:23:00	612

(c) Gęsia do Ołtaszyn

Journey from Dworzec Główny (Dworzec) to PL. Daniłowski started at: 14:30:00:
Travel time: 31.0 minutes
Number of changes: 5
Elapsed in: 1.9153 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
DWORZEC GŁÓWNY (Dworzec)	14:32:00	SMER KRAKOWSKIEGO	14:34:00	146
SMER KRAKOWSKIEGO	14:34:00	KRAKOWSKIEGO	14:35:00	146
KRAKOWSKIEGO	14:41:00	URZĄD MIEJSCOWY (MUSEUM WARSZAWSKIE)	14:44:00	A
URZĄD MIEJSCOWY (MUSEUM WARSZAWSKIE)	14:44:00	KATEDRA	14:45:00	2
KATEDRA	14:46:00	OGROD BOTANICZNY	14:48:00	A
OGROD BOTANICZNY	14:48:00	WYSZYŃSKIEGO	14:50:00	A
WYSZYŃSKIEGO	14:50:00	DAPNOTA	14:51:00	A
DAPNOTA	14:52:00	KROPIERA	14:55:00	H
KROPIERA	14:56:00	BERENTA	14:59:00	A
BERENTA	14:59:00	PL. DANIŁOWSKIEGO	15:01:00	A

(d) Dworzec Główny do Pl. Daniłowski

Rysunek 1: Wyniki algorytmu Dijkstra dla różnych połączeń komunikacji miejskiej

Wyniki otrzymane przy algorytmie Dijkstra są zadowalające, gdyż odnaleziono połączenia między przystankami w prawidłowy sposób oraz godziny odjazdów i przyjazdów na siebie nie nachodzą. Problemem rozwiązania jest kolosalna liczba przesiadek, którą należy wykonać w niektórych przypadkach oraz czas wykonywania się programu, który mógłby być szybszy.

8 Algorytm A*

Algorytm A* jest heurystycznym algorytmem służącym do znajdowania najkrótszej ścieżki w grafie i jest rozbudowaniem algorytmu Dijkstry o dodatkowy moduł estymacji kosztu ścieżki od celu. Algorytm ten przy nieprawidłowo zaprojektowanej heurystyce może osiągnąć wyniki dalekie od optymalnych. Rozwiązanie rozdzielono na dwa podejścia - kryterium czasu oraz kryterium przystanków. Pierwsze ma na celu przeprowadzić użytkownika z przystanku A do przystanku B w najkrótszym możliwym czasie, a drugie ma na celu dokonania tego samego, lecz z najkrótszą możliwą liczbą zmian linii.

8.1 Algorytm A* o kryterium czasu

8.1.1 Heurystyka

W rozwiązaniach wykorzystano heurystykę dotyczącą odległości aktualnego przystanku do przystanku końcowego. W momencie wybierania kolejnych kursów, oprócz sprawdzania odległości mierzonej w minutach dodatkowo brano pod uwagę różnicę odległościową. Dodawana waga była obliczana za pomocą przekonwertowania koordynatów przystanku aktualnego i końcowego na kilometry dzięki bibliotece geopy.distance. Następnie przyjęto, iż prędkość komunikacji miejskiej we Wrocławiu to 30 km/h, jako iż należy brać pod uwagę potencjalne postoje na światłach.

8.1.2 Wyniki

Journey from KWIŚKA to PL. GRUNWALDZKI started at: 9:00:00:
Travel time: 26.0 minutes
Number of changes: 5
Elapsed in: 1.2998 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
KWIŚKA	09:01:00	PAŁAPANIEWSKA	09:02:00	20
PAŁAPANIEWSKA	09:02:00	WIELOŚCIECZKA	09:04:00	20
WIELOŚCIECZKA	09:04:00	WROCŁAW MIĘDULAZÓW (ZACHODNIA)	09:06:00	20
WROCŁAW MIĘDULAZÓW (ZACHODNIA)	09:06:00	DOLNE	09:07:00	20
DOLNE	09:07:00	PL. STRZEGOMSKI (MUSEUM WŚPÓŁCZESNE)	09:09:00	33
PL. STRZEGOMSKI (MUSEUM WŚPÓŁCZESNE)	09:09:00	PIŁCZYCH TECHNIKÓW	09:10:00	33
PIŁCZYCH TECHNIKÓW	09:10:00	PL. JANA PAWŁA II	09:12:00	33
PL. JANA PAWŁA II	09:12:00	RYNEK	09:14:00	33
RYNEK	09:14:00	ZAPRAWA	09:15:00	33
ZAPRAWA	09:15:00	ŚWIDRZKA	09:17:00	33
ŚWIDRZKA	09:17:00	GALERIA DOMINICAŃSKA	09:19:00	33
GALERIA DOMINICAŃSKA	09:19:00	URZĄD MIEJOWOŚCI (DZIWIART)	09:22:00	33
URZĄD MIEJOWOŚCI (DZIWIART)	09:22:00	POST GRUNWALDZKI	09:24:00	33
POST GRUNWALDZKI	09:24:00	PL. GRUNWALDZKI	09:26:00	33

(a) Kwiska do Pl. Grunwaldzki

Journey from GĘSIA to PL. GRUNWALDZKI started at: 9:00:00:
Travel time: 47.0 minutes
Number of changes: 5
Elapsed in: 2.1299 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
GĘSIA	14:38:00	KWIŚKA	14:40:00	23
KWIŚKA	14:40:00	KĘTRZYŃSKA	14:41:00	23
KĘTRZYŃSKA	14:41:00	KROPIERA	14:43:00	23
KROPIERA	14:43:00	POSTY WARSZAWSKIE	14:45:00	A
POSTY WARSZAWSKIE	14:45:00	WYSZYŃSKIEGO	14:47:00	A
WYSZYŃSKIEGO	14:47:00	OGROD BOTANICZNY	14:49:00	A
OGROD BOTANICZNY	14:49:00	KATEIDRA	14:51:00	A
KATEIDRA	14:51:00	URZĄD MIEJOWOŚCI (MUSEUM WARSZAWSKIE)	14:52:00	A
URZĄD MIEJOWOŚCI (MUSEUM WARSZAWSKIE)	14:52:00	POCZTA GŁÓWNA	14:53:00	A
POCZTA GŁÓWNA	14:53:00	SIEDER KRASIŃSKIEGO	14:55:00	A
SIEDER KRASIŃSKIEGO	14:55:00	DZIKOWA	14:58:00	146
DZIKOWA	15:00:00	DZIKOWA	15:01:00	9
DZIKOWA	15:01:00	KROPIERA	15:08:00	K
KROPIERA	15:08:00	ŚWIDRZKA	15:10:00	K
ŚWIDRZKA	15:10:00	DZIKOWA	15:13:00	K
DZIKOWA	15:14:00	KROPIERA	15:16:00	612
KROPIERA	15:16:00	PRZYSTANOK	15:17:00	612
PRZYSTANOK	15:17:00	WISZYŃSKIEGO	15:18:00	612
WISZYŃSKIEGO	15:18:00	PARAFIALNA	15:20:00	612
PARAFIALNA	15:20:00	ŚWIDRZKA	15:21:00	612
ŚWIDRZKA	15:21:00	OLASZYN	15:23:00	612

(c) Gęsia do Ołtaszyn

Journey from POMORSKA to MOSTY POMORSKIE started at: 9:00:00:
Travel time: 3.0 minutes
Number of changes: 0
Elapsed in: 0.0656 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
POMORSKA	09:02:00	MOSTY POMORSKIE	09:03:00	18

(b) Pomorska do Mosty pomorskie

Journey from DZIKOWA GŁÓWNA (DZIKOWA) to PL. DANILÓWSKIEGO started at: 14:30:00:
Travel time: 31.0 minutes
Number of changes: 5
Elapsed in: 1.5272 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
DZIKOWA GŁÓWNA (DZIKOWA)	14:32:00	SIEDER KRASIŃSKIEGO	14:34:00	146
SIEDER KRASIŃSKIEGO	14:34:00	KROPIERA	14:35:00	146
KROPIERA	14:41:00	URZĄD MIEJOWOŚCI (MUSEUM WARSZAWSKIE)	14:44:00	A
URZĄD MIEJOWOŚCI (MUSEUM WARSZAWSKIE)	14:44:00	KATEIDRA	14:45:00	Z
KATEIDRA	14:46:00	OGROD BOTANICZNY	14:48:00	A
OGROD BOTANICZNY	14:48:00	WYSZYŃSKIEGO	14:50:00	A
WYSZYŃSKIEGO	14:50:00	DZIKOWA	14:51:00	A
DZIKOWA	14:52:00	KROPIERA	14:55:00	H
KROPIERA	14:56:00	BERENTA	14:59:00	A
BERENTA	14:59:00	PL. DANILÓWSKIEGO	15:01:00	A

(d) Dworzec Główny do Pl. Daniłowskiego

Rysunek 2: Wyniki algorytmu A* kryterium czasu dla różnych połączeń komunikacji miejskiej

Wyniki algorytmu A* o kryterium czasu są bardziej zadowalające od algorytmu Dijkstry jedynie pod względem czasu obliczeń. Finalne trasy jak i czas podróży nie różniły się w jakikolwiek sposób między tymi rozwiązaniami, lecz dzięki dodaniu heurystyki związanej z obliczaniem odległości między przystankiem aktualnym, a finalnym skrócono liczbę analizowanych połączeń.

8.2 Algorytm A* o kryterium przesiadek

8.2.1 Heurystyka

W rozwiązaniu ponownie wykorzystano heurystykę dotyczącą odległości aktualnego przystanku do przystanku końcowego. Dodatkowo w celu uwzględnienia kryterium przesiadek w pierwszej kolejności w prosty sposób sprawdzano czy zmieniono linie przy kolejnych stacjach. W tym przypadku dodawano karę do danego przejazdu, lecz rozwiązanie okazało się być marne. Problemem tego rozwiązania było niebranie pod uwagę, czy wsiadamy do linii, która dojeżdża do końcowego przystanku, zamiast tego brano pod uwagę czy po prostu zmieniamy linię. Heurystyka została zmieniona na taką, która analizuje aktualne linie wychodzące z przystanku na którym się znajdujemy. Analiza polega na sprawdzeniu w pierwszej kolejności czy się przesiadliśmy, a następnie na sprawdzeniu czy na aktualnym przystanku istnieją linie, która dojeżdża na przystanek końcowy. Na aktualnym przystanku sprawdzamy wszystkie wychodzące z niego linie, a następnie sprawdzamy wszystkie wychodzące linie na przystanku końcowym do którego dążymy, jeśli na naszym przystanku jest linia, która jest także na przystanku końcowym to znaczy, że znaleźliśmy bezpośrednie połączenie. Dodatkowo należy wziąć pod uwagę opłacalność czekania na taką linię. Możliwym jest wybranie połączenia bezpośredniego, lecz jeśli należy poczekać na taki tramwaj kilka godzin jest to irracjonalne i w takich przypadkach należy kierować się w stronę celu i szukać bezpośredniej linii w dalszych krokach.

8.2.2 Wyniki

Journey from KATSKA to PL. GRUNWALDZKI started at: 9:00:00:
Travel time: 31.0 minutes
Number of changes: 0
Elapsed in: 0.0722 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
KATSKA	09:06:00	MACOPIENSKA	09:07:00	12
MACOPIENSKA	09:07:00	WIEDZIĘCZA	09:09:00	12
WIEDZIĘCZA	09:09:00	MIECŁAN MOKŁASZÓW (ZACHODNIA)	09:11:00	12
MIECŁAN MOKŁASZÓW (ZACHODNIA)	09:11:00	PL. STRZEGOMSKI (MUZEUM WARSZĄCZESNE)	09:13:00	12
PL. STRZEGOMSKI (MUZEUM WARSZĄCZESNE)	09:13:00	MŁOCYCH TECHNIKÓW	09:14:00	12
MŁOCYCH TECHNIKÓW	09:14:00	PL. JANA PAWŁA II	09:16:00	12
PL. JANA PAWŁA II	09:16:00	RYNEK	09:19:00	12
RYNEK	09:19:00	ZAPRAWA	09:20:00	12
ZAPRAWA	09:20:00	ŚWIŁCZKA	09:22:00	12
ŚWIŁCZKA	09:22:00	GALERIA DOPINIEKARSKA	09:24:00	12
GALERIA DOPINIEKARSKA	09:24:00	URZĄD MIEJSCOWY (DIPART)	09:27:00	12
URZĄD MIEJSCOWY (DIPART)	09:27:00	POST GRUNWALDZKI	09:29:00	12
POST GRUNWALDZKI	09:29:00	PL. GRUNWALDZKI	09:31:00	12
PL. GRUNWALDZKI	09:31:00	POST GRUNWALDZKI	09:31:00	13
POST GRUNWALDZKI	09:31:00	PL. GRUNWALDZKI	09:36:00	13

(a) Kwiska do Pl. Grunwaldzki

PS D:\stacja\AStar\solution.py python main.py
Journey from GĘSIA to OŁTĄSZYN started at: 14:30:00:
Travel time: 57.0 minutes
Number of changes: 2
Elapsed in: 0.7548 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
GĘSIA	14:30:00	KWIDZYŃSKA	14:32:00	6
KWIDZYŃSKA	14:37:00	ZACISZE	14:38:00	D
ZACISZE	14:38:00	ŚWIĄTECKICH	14:39:00	D
ŚWIĄTECKICH	14:39:00	KOCZAŃSKIEGO	14:41:00	D
KOCZAŃSKIEGO	14:41:00	PL. GRUNWALDZKI	14:45:00	D
PL. GRUNWALDZKI	14:45:00	POST GRUNWALDZKI	14:47:00	D
POST GRUNWALDZKI	14:47:00	POCZTA GŁÓWNA	14:49:00	D
POCZTA GŁÓWNA	14:49:00	GALERIA DOPINIEKARSKA	14:51:00	D
GALERIA DOPINIEKARSKA	14:51:00	REKOPA	14:55:00	D
REKOPA	14:55:00	ARADY (CAPITOL)	14:57:00	D
ARADY (CAPITOL)	14:57:00	ROKÓ	15:01:00	D
ROKÓ	15:01:00	HALLERA	15:04:00	D
HALLERA	15:04:00	ORIA	15:07:00	D
ORIA	15:07:00	KRYKCI	15:09:00	D
KRYKCI	15:11:00	PRZYJAŃ	15:13:00	113
PRZYJAŃ	15:13:00	PARTYKICE (TOR WŚCISŁÓW KOMBICA)	15:17:00	113
PARTYKICE (TOR WŚCISŁÓW KOMBICA)	15:17:00	HURARIKA	15:19:00	113
HURARIKA	15:19:00	ROKÓ ŚW. OJCA PŁD	15:22:00	113
ROKÓ ŚW. OJCA PŁD	15:22:00	STRACHOWSKIEGO	15:25:00	113
STRACHOWSKIEGO	15:25:00	OŁTĄSZYN	15:27:00	113

(c) Gęsia do Ołtaszyn

Journey from POMORSKA to MOSTY POMORSKIE started at: 9:00:00:
Travel time: 3.0 minutes
Number of changes: 0
Elapsed in: 0.0225 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
POMORSKA	09:02:00	MOSTY POMORSKIE	09:03:00	18

(b) Pomorska do Mosty pomorskie

Journey from DWORZEC GŁÓWNY (MORCINA) to PL. DANIŁOWSKIEGO started at: 14:30:00:
Travel time: 31.0 minutes
Number of changes: 0
Elapsed in: 0.0322 seconds

Start Stop	Departure Time	End Stop	Arrival Time	Line
DWORZEC GŁÓWNY (MORCINA)	14:38:00	SIEMER KRASIŃSKIEGO	14:40:00	A
SIEMER KRASIŃSKIEGO	14:40:00	KRASIŃSKIEGO	14:41:00	A
KRASIŃSKIEGO	14:41:00	URZĄD MIEJSCOWY (MUZEUM NARODOWE)	14:44:00	A
URZĄD MIEJSCOWY (MUZEUM NARODOWE)	14:44:00	KATERA	14:46:00	A
KATERA	14:46:00	OGROD BOTANICZNY	14:48:00	A
OGROD BOTANICZNY	14:48:00	WYSZYKIEGO	14:50:00	A
WYSZYKIEGO	14:50:00	DAWIDA	14:51:00	A
DAWIDA	14:51:00	KOPRIBA	14:56:00	A
KOPRIBA	14:56:00	BERENTA	14:59:00	A
BERENTA	14:59:00	PL. DANIŁOWSKIEGO	15:01:00	A

(d) Dworzec Główny do Pl. Daniłowskiego

Rysunek 3: Wyniki algorytmu A* kryterium przesiadek dla różnych połączeń komunikacji miejskiej

Wyniki tego podejścia są najbardziej zadowalające, gdyż optymalizują podejście czasowe wraz z podejściem przesiadkowym. Dzięki temu uzyskujemy podróż w której nie trzeba zmieniać kilkakrotnie linii oraz

dojazd jest stosunkowo szybki. Dodatkowym ogromnym atutem tego rozwiązania jest diametralne zwiększenie szybkości działania programu. Ze względu na odrzucenie tysięcy linii, które nie dojeżdżają do końcowego przystanku wystarczy przeanalizować tylko kilka rekordów.

9 Porównanie wyników dla pierwszego problemu

9.1 Typowy przejazd

Dla przejazdu typowego wyniki między djikstrą, a algorytmem A^* z kryterium czasowym nie różnią się niczym poza czasem poszukiwań. Dla algorytmu djikstra czas poszukiwań trwał 1.77 sekundy, a dla algorytmu A^* o kryterium czasowym 1.29 sekundy. Wykazał się pod tym względem algorytm A^* o kryterium przesiadek, który odnalazł trasę w zaledwie 0,07 sekundy. Dodatkowym atutem kryterium przesiadek okazało się odnalezienie linii 12, która bezpośrednio łączy przystanek początkowy z końcowym. Pozostałe rozwiązania odnalazły trasę, którą należy przemierzyć tramwajami 20, a następnie 13. Finalnie zaletą djikstry oraz A^* z kryterium czasowym jest czas podróży, która wyniosła kilka minut mniej.

9.2 Krótki przejazd

Przykład ten miał jedynie sprawdzić czy algorytmy radzą sobie w nietypowych sytuacjach, które normalnie należałoby przemierzyć idąc pieszo. Algorytmy wybrały tę samą trasę, lecz różnił je czas działania. Ponownie A^* o kryterium przesiadek wyprzedził pozostałe rozwiązania, dokonując obliczeń 3 krotnie szybciej.

9.3 Długi przejazd

Długi przejazd, który odbył się między stacją Gęsia, a Ołtaszyn ponownie ukazał różnicę między rozwiązaniami. Względem czasu dokonywania obliczeń, ponownie A^* o kryterium przesiadek był najszybszy, następnie jego alternatywa, a na końcu Djikstra. Wyniki wyniosły kolejno 0.75 sekundy, 2.32 sekundy oraz 2.43 sekundy. Trasy djikstry oraz A^* o kryterium czasu wyglądały tak samo, dokonano aż 5 przesiadek oraz podróżowano 47 minut. Pozytywnym zaskoczeniem dla kryterium przesiadek, jest zwiększenie czasu podróży zaledwie do 57 minut, a dokonanie jedynie 2 przesiadek.

9.4 Skomplikowany przejazd

Finalnym porównywaniem przyjazdem jest kurs z Dworca Głównego na Pl. Daniłowskiego. Przystanki te nie są ogromnie oddalone, lecz przejeżdżają przez bardzo gęste strefy miasta. Czas wykonywania programu zajął 1.91 sekundy dla djikstry, 1.52 sekundy dla A^* oraz 0.03 sekundy dla A^* przesiadek. Djikstra oraz A^* czasowy znalazły trasę, która trwała jedynie 31 minut, a składała się z 5 przesiadek, wynika z tego, iż przejechano 2 przystanki jednym tramwajem i już należało zmienić linię. Podróż A^* o kryterium przesiadek trwała tyle samo czasu, czyli 31 minut. Jest to zaskakujące, ponieważ znaleziono trasę bez przesiadek autobusem A.

10 Tabu search

10.1 Wprowadzenie

Tabu search skupia się na rozwiązywaniu problemu komiwojażera, który polega na znalezieniu najkrótszej możliwej ścieżki odwiedzającej zadaną listę przystanków komunikacji miejskiej i powrotu do punktu startowego. W porównaniu do klasycznego algorytmu Tabu Search, wprowadzono kilka istotnych usprawnień, które znacząco wpłynęły na jakość i szybkość uzyskiwanych rozwiązań.

Kluczowe modyfikacje obejmują dynamiczne dostosowanie liczby iteracji w zależności od rozmiaru problemu, zastosowanie specyficznych kryteriów aspiracji oraz inteligentne zarządzanie listą tabu, co pozwoliło na efektywniejsze eksplorowanie przestrzeni rozwiązań i unikanie pułapek lokalnych minimów. Dodatkowo

wprowadzono modyfikacje dzięki której algorytm A* o kryterium czasu używany jest do estymacji dystansów jedynie w inicjalizacji tablicy. Dzięki tym zmianom, algorytm Tabu Search w prezentowanym projekcie wykazuje wyższą efektywność i adaptacyjność w kontekście optymalizacji tras komunikacji miejskiej.

10.2 Wyniki tabu seach

```
Iteration 0: Best solution cost = 34.0
Iteration 1: Best solution cost = 34.0
Iteration 2: Best solution cost = 34.0
Iteration 3: Best solution cost = 34.0
time: 3.286006212234497
Best solution: ['Dworzec Główny', 'Pl. Grunwaldzki', 'Pl. Bema', 'Pomorska', 'Świdnicka']
```

(a) Dworzec główny, pl. Grunwaldzki, Pomorska, pl. Bema, Świdnicka

```
Iteration 0: Best solution cost = 130.0
Iteration 1: Best solution cost = 128.0
Iteration 2: Best solution cost = 128.0
Iteration 3: Best solution cost = 128.0
Iteration 4: Best solution cost = 128.0
time: 11.598221485829297
Best solution: ['Kwiska', 'Kozanów', 'Wolska', 'Dworzec Główny', 'Świdnicka']
```

(b) Pomorska, Rynek, pl. Bema, Dubois

```
Iteration 0: Best solution cost = 20.0
Iteration 1: Best solution cost = 20.0
Iteration 2: Best solution cost = 20.0
Iteration 3: Best solution cost = 20.0
time: 0.6543006896972656
Best solution: ['Pomorska', 'Rynek', 'Pl. Bema', 'Dubois']
```

(c) Bardzka, Hallera, Górnica

```
Iteration 0: Best solution cost = 78.0
Iteration 1: Best solution cost = 78.0
Iteration 2: Best solution cost = 78.0
Iteration 3: Best solution cost = 78.0
Iteration 4: Best solution cost = 78.0
Iteration 5: Best solution cost = 78.0
Iteration 6: Best solution cost = 78.0
Iteration 7: Best solution cost = 78.0
Iteration 8: Best solution cost = 78.0
Iteration 9: Best solution cost = 78.0
time: 3.9051008224487305
Best solution: ['Bardzka', 'Hallera', 'Górnica']
```

(d) Kwiska, Kozanów, Wolska, Dworzec Główny, Świdnicka

Rysunek 4: Wyniki algorytmu tabu seach dla tabu tenure równego liczbie przystanków do potęgi 8

Dla parametrów tabu tenure równego liczbie przystanków wyniki były takie same, lecz czasy diametralnie zwiększyły się i wynosiły kolejno

time: 6.03
time: 28.26
time: 1.51
time: 8.91

Gdzie czasy przy tabu tenure równym licznie przystanków do potęgi 8 wynosiły:

time: 3.51
time: 11.66
time: 0.63
time: 3.92

Czasy przy tabu tenure równym licznie przystanków do potęgi 4 wynosiły:

time: 4.14
time: 18.46
time: 0.93
time: 5.12

W teorii Kryterium aspiracji w Tabu Search służy do przełamywania reguł tabu w przypadku, gdy potencjalne rozwiązanie jest obiecujące i może prowadzić do znalezienia lepszego niż dotychczas najlepszego rozwiązania globalnego, lecz nie udało się znaleźć wartości kryterium która by usprawniała funkcjonowanie programu pod względem jakości wyników jak i czasu obliczeń.

11 Podsumowanie projektu oraz wnioski

Projekt dotyczący algorytmów przeszukania w problemach transportu miejskiego skupił się na analizie i optymalizacji tras między przystankami we Wrocławiu, wykorzystując do tego celu algorytmy Dijkstra, A* oraz Tabu Search. Celem projektu było zapoznanie się z problemami oraz algorytmami optymalizacyjnymi oraz praktyczne zastosowanie ich. Pierwszym krokiem podjętym w projekcie było zrozumienie wszystkich algorytmów i struktury danych na których operuje. Następnie należało przeanalizować dane podane w pliku csv oraz oczyścić je. Kolejnym krokiem było stworzenie struktury danych i napisanie 3 algorytmów optymalizacyjnych wraz z wymyśleniem heurystyk. Po przeprowadzeniu testów w zadaniu pierwszym ukazały się fundamentalne wnioski:

Każdy algorytm ma swoje zalety i wady. Dijkstra okazał się być najłatwiejszym i najszybszym w implementacji algorytmem. Jego negatywną stroną jest długi czas obliczeń oraz znajdowanie tras z ogromną ilością przesiadek. Algorytm A* z kryterium czasowym wyróżnił się szybszym czasem kalkulacji ze względu na branie pod uwagę jedynie tras, które zbliżają nas do celu, lecz tak samo przeciętnie radził sobie ze znajdowaniem wygodnych przejazdów. Finalnie algorytm A* z kryterium transportu okazał się być zwycięzcą względem wygody użytkownika oraz czasu obliczeń. Algorytm ten wymagał najmniej przesiadek. Podejście zawiera także wady. Pierwszą z nich jest ogromna złożoność implementacyjna. Zaimplementowanie algorytmu zajęło tyle czasu ile cała pozostała część projektu. Drugą częściową wadą jest czas podróży, który niestety przez oczekiwanie na potencjalne przesiadki się wydłuża, lecz jak ukazał przykład skomplikowanej trasy z dworca, odnalezienie pojedynczej linii poskutkowało skróceniu czasu podróży.

Tabu Search, zaimplementowany w celu rozwiązania problemu komiwojażera, wprowadził minimalne usprawnienia dzięki dynamicznemu dostosowywaniu parametrów i specyficznym kryteriom aspiracji. Mimo braku znaczących zmian w wynikach spowodowanych kryterium aspiracji, algorytm ten wykazał się zwiększoną adaptacyjnością i efektywnością. Projekt ten udowadnia, że odpowiednie zastosowanie algorytmów optymalizacyjnych może przyczynić się do optymalnego wyznaczania tras między przystankami komunikacji miejskiej we Wrocławiu. Dodatkowo modyfikacja algorytmów optymalizacyjnych może znacząco wpływać na jakość i szybkość rozwiązań w kontekście optymalizacji tras komunikacji miejskiej.