



Wrocław University
of Science and Technology

Projektowanie efektywnych algorytmów

Projekt 1

Wydział Elektroniki
Kierunek: Informatyka
Paweł SZYNAL 226026

Prowadzący:
mgr inż. Antoni Sterna

Wrocław 2020 r.

Spis treści

1	Wstęp	2
2	Specyfikacja techniczna	2
3	Analiza problemu	2
4	Opis Algorytmów	2
4.1	Przegląd zupełny	2
4.2	Metoda podziału i ograniczeń	3
5	Pomiary i wnioski	3

1 Wstęp

Celem projektu było wykonanie programu, wykorzystującego algorytmy programowania dynamicznego, podziału i ograniczeń oraz przeglądu zupełnego do rozwiązania problemu komiwojażera (ang. Travelling Salesman Problem).

2 Specyfikacja techniczna

- Program został wykonany obiektowo w języku c++
- Program akceptuje dane w postaci macierzy odległości
- Czas wykonania algorytmów mierzone był przy wykorzystaniu bibliotek systemowych
- do dynamicznego przechowywania danych została wykorzystana biblioteka *Vector*

3 Analiza problemu

Problem komiwojażera należy do klasy problemów NP-trudnych. Jest to optymalizacyjny problem, rozwiązaniem którego jest znalezienie minimalnego cyklu Hamiltona (ścieżki prowadzącej przez wszystkie wierzchołki grafu, powracając na końcu do wierzchołka początkowego). W wersji asynchronicznej, odległości pomiędzy wierzchołkami mogą dodatkowo zależeć także od kierunku przejścia pomiędzy nimi. Główną trudnością w rozwiązaniu problemu jest znacząca liczba możliwych kombinacji.

4 Opis Algorytmów

4.1 Przegląd zupełny

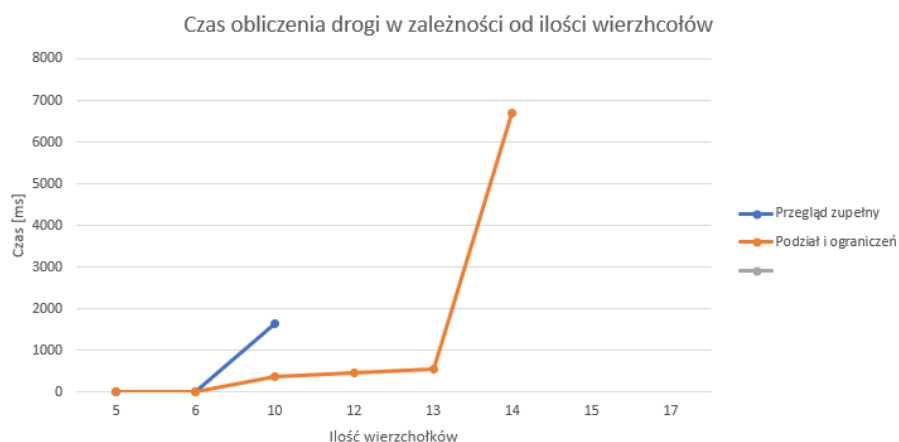
Algorytm przeglądu zupełnego (ang. brute force) polega na przeanalizowaniu wszystkich możliwych przypadków, oraz wybraniu tego o najlepszej wartości. Zaletą tego algorytmu jest pewność, że otrzymany wynik jest najlepszym rozwiązaniem problemu. Poważną jego wadą jest jednak złożoność czasową wynoszącą $O(n!)$, co w praktyce czyni ten algorytm bezużytecznym dla większych zbiorów danych. Zaimplementowany został algorytm przeszukiwania w głąb wywoływany rekurencyjnie ze zmiennymi śledzącymi najkrótszą drogę i koszt.

4.2 Metoda podziału i ograniczeń

Metoda polega przechodzeniu w głąb problemu przy jednoczesnym obliczaniu minimum - *upperBound* (poprzez metodę redukcji macierzy). Po znalezieniu pierwszej drogi zostaje zaktualizowana wartość *lowerBound* i wyeliminowane wszystkie elementy które posiadają wartość *upperBound* większą od *lowerBound*. Minusem tego algorytmu jest duża złożoność pamięciowa, gdyż dla każdego elementu tworzymy uaktualnioną dla danego elementu kopię macierzy kosztów przejścia pomiędzy wierzchołkami. W najgorszym przypadku odwiedzimy każdy wierzchołek, tak jak przy przeglądzie zupełnym.

5 Pomiary i wnioski

Każdy pomiar został wykonany dziesięciokrotnie, a potem uśredniony.



Rysunek 1: Pomiary w [ms] w zależności od ilości wierzchołów

Jak widać algorytmy przeglądu zupełnego i podziału i ograniczeń okazują się nieefektywne już przy odpowiednio 12 i 15 wierzchołkach. Ponadto algorytm podziałów i ograniczeń mimo większej wydajności zużywa dużo więcej pamięci, a jego szybkość nie jest stała - zależna od grafu jak i ilości wierzchołów grafu.