Paweł Twardawa 235072 Data: 16.11.2018 r.

PROJEKTOWANIE EFEKTYWNYCH ALGORYTMÓW

Projekt 1

Prowadzący:

Dr inż. Zbigniew Buchalski

Spis treści

1. Wstęp

Celem projektu była implementacja oraz analiza efektywność algorytmu podziału i ograniczeń, programowania dynamicznego oraz przeglądu zupełnego dla asymetrycznego problemu komiwojażera (ATSP).

2. Analiza Problemu

Problem komiwojażera należy do klasy problemów NP-trudnych. Polega on na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie. Każdy wierzchołek grafu reprezentuje miasto, które musi odwiedzić komiwojażer. Grupę n miast reprezentuje zbiór N = {1, …, n}. Miasta są ze sobą połączone krawędziami d. Długość tych krawędzi zawiera macierz D = { d_{ij}, i ϵ N, j ϵ N, i ≠ j}. Gdzie $d_{ij} \geq 0$ oznacza odległość między miastem i oraz j. W wersji asymetrycznej, odległość pomiędzy miastem i oraz j może być inna niż odległość miasta j od i: $d_{ij} \neq d_{ji}$ Główną trudnością w rozwiązaniu problemu jest znacząca liczba możliwych kombinacji.

3. Opis Algorytmów

3.1. Przegląd zupełny

Algorytm przeglądu zupełnego polega na szukaniu najlepszego rozwiązania. Zaletą tego algorytmu jest to że zawsze otrzymamy najlepszy wynik rozwiązania problemu. Jednak algorytm jest nieefektywny obliczeniowo. Jego złożoność obliczeniowa wynosi $O(2^n)$, co powoduje że algorytm jest bezużyteczny dla dużych n. W projekcie został zaimplementowany algorytm przeszukiwania w głąb wywołany rekurencyjnie.

3.2. Metoda podziału i ograniczeń

Metoda podziału i ograniczeń służy do rozwiązywania problemów optymalizacyjnych. Jej działanie opiera się na analizie drzewa przestrzeni stanów. Drzewo to reprezentuje wszystkie możliwe ścieżki jakimi może pójść algorytm rozwiązujący problem. Algorytm zaczyna w korzeniu drzewa i przechodząc do któregoś liścia konstruuje rozwiązanie. Metoda w każdym węźle oblicza granicę, która pozwala określić go jako obiecujący bądź nie. Minusem tego algorytmu jest duża złożoność pamięciowa algorytmu, gdyż dla każdego elementu tworzymy uaktualnioną kopię macierzy kosztów przejścia pomiędzy wierzchołkami. W najgorszym przypadku złożoność obliczeniowa będzie równa złożoności przeglądu zupełnego.

3.3. Programowanie dynamiczne

Programowanie dynamiczne jest metodą rozwiązywania zagadnień optymalizacyjnych. Opiera się na podziale rozwiązanego problemu na podproblemy względem kilku parametrów. Każdy z podproblemów rozważany jest tylko raz, a wynik jego analizy przechowywany jest do wykorzystania w późniejszych obliczeniach. Dla problemu komowojażera najlepszym rozwiązaniem jest implementacja algorytmu Helda-Karpa, posiadający złożoność $O(n^2 * 2n)$. Algorytm korzysta z $2^{(n-1)}$ elementowej tablicy. Tablica jest indeksowana od 0 a każdy z nich jest maską odwiedzonych miast. Przykładowo dla 4 miast ostatnim indeksem jest 7, czyli 111_2. Mimo 4 miast potrzebne są nam tylko 3 bity, gdyż ostatnie miasto jest już wybrane i znajdujemy jedynie wierzchołek o najniższym koszcie przejścia od ostatniego wierzchołka. W tym elemencie znajduje się lista kosztów przejścia poprzedzających miast, ze wskazaniem na ostatni punkt. Wybierając najmniejszy element na przykład 2 przechodzimy do maski 010_2, gdzie powtarzamy algorytm, aż do całkowitego odtworzenia drogi. Należy także pamiętać o dodaniu kosztów powrotu do ostatniego wierzchołka.

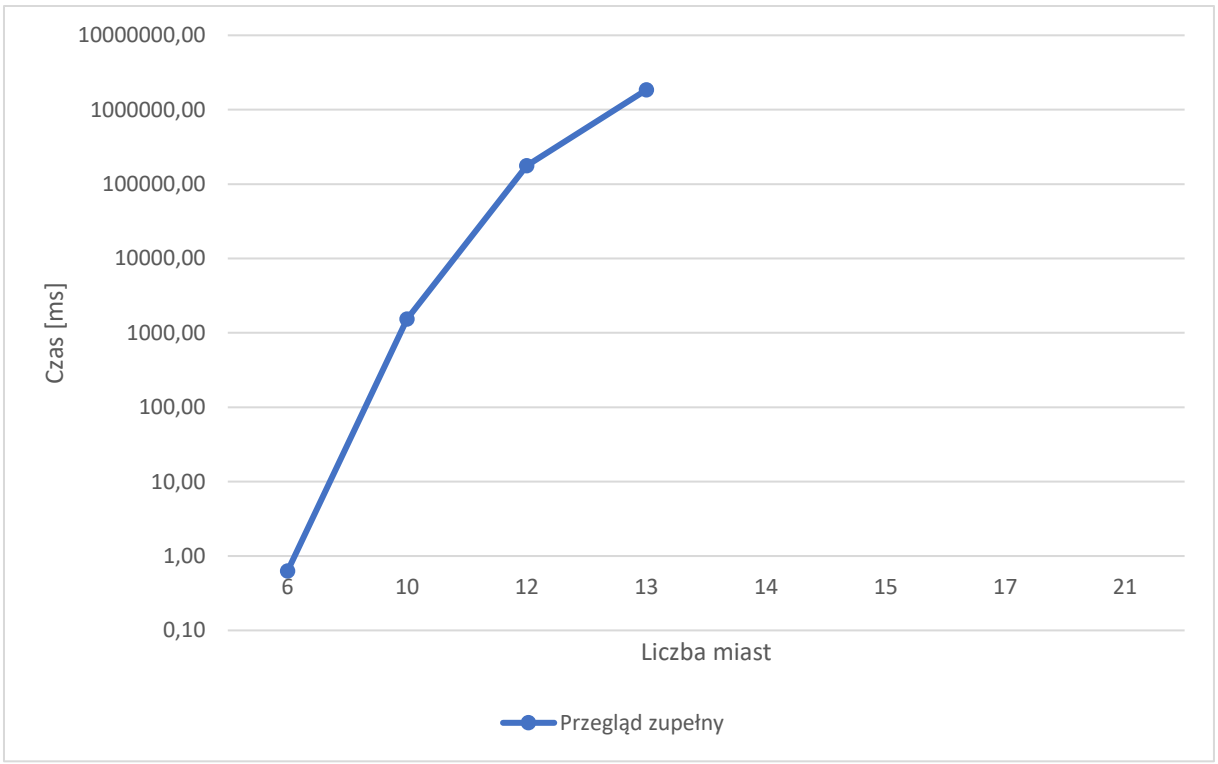
4. Plan Eksperymentu

Badanie czasu wykonywania poszczególnych algorytmów zostało wykonane za pomocą metody StartCounter() oraz GetCounter(). Metody to wykorzystują systemowe funkcje które po przeliczeniu zwracają dokładny czas wykonywania operacji. Każdy z zaimplementowanych algorytmów został zbadany kilkanaście razy na różnej wielkości danych wejściowych , a wartości zostały uśrednione. Program został wykonany w języku C++ zgodnie z obiektowym paradygmatem programowania w wersji konsolowej. Program jako dane wejściowe przyjmuje macierz określającą koszt przejścia pomiędzy miastami. Do przechowywania danych została wykorzystana biblioteka Vector.

5. Wyniki

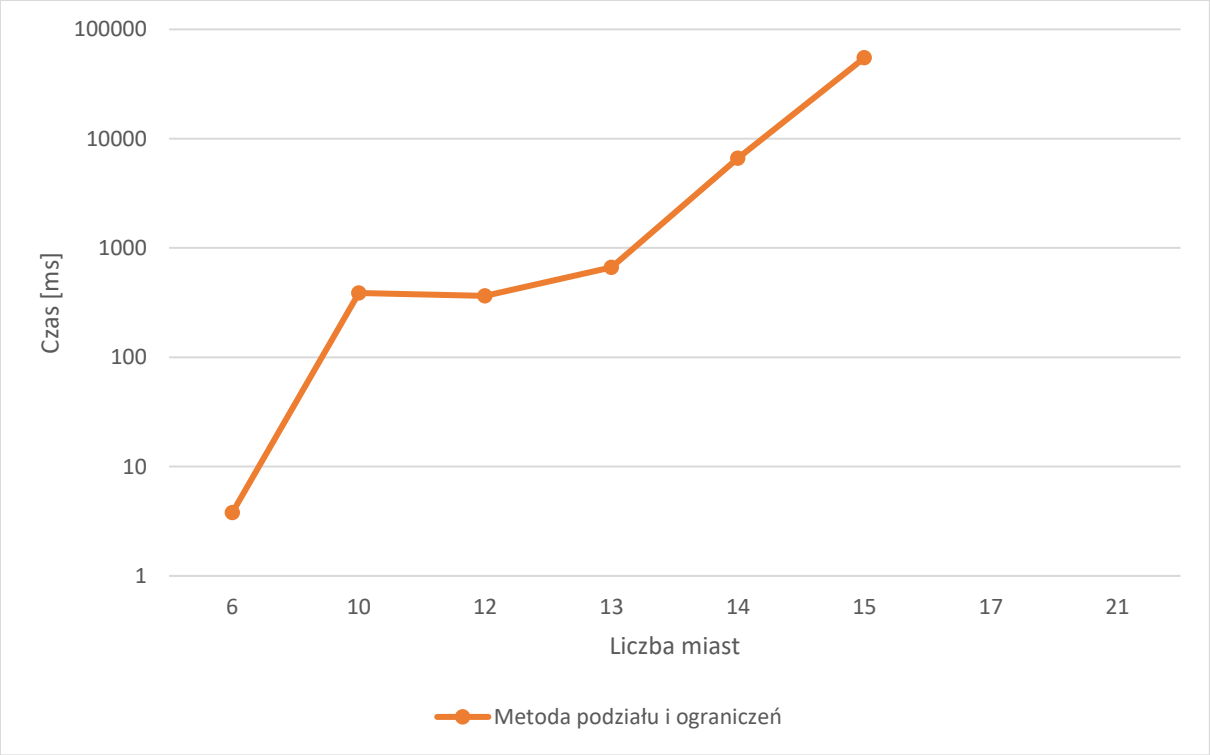
5.1. Przegląd zupełny

	Przegląd zupełny
6	0,630
10	1534,495
12	177155,000
13	1848010,000



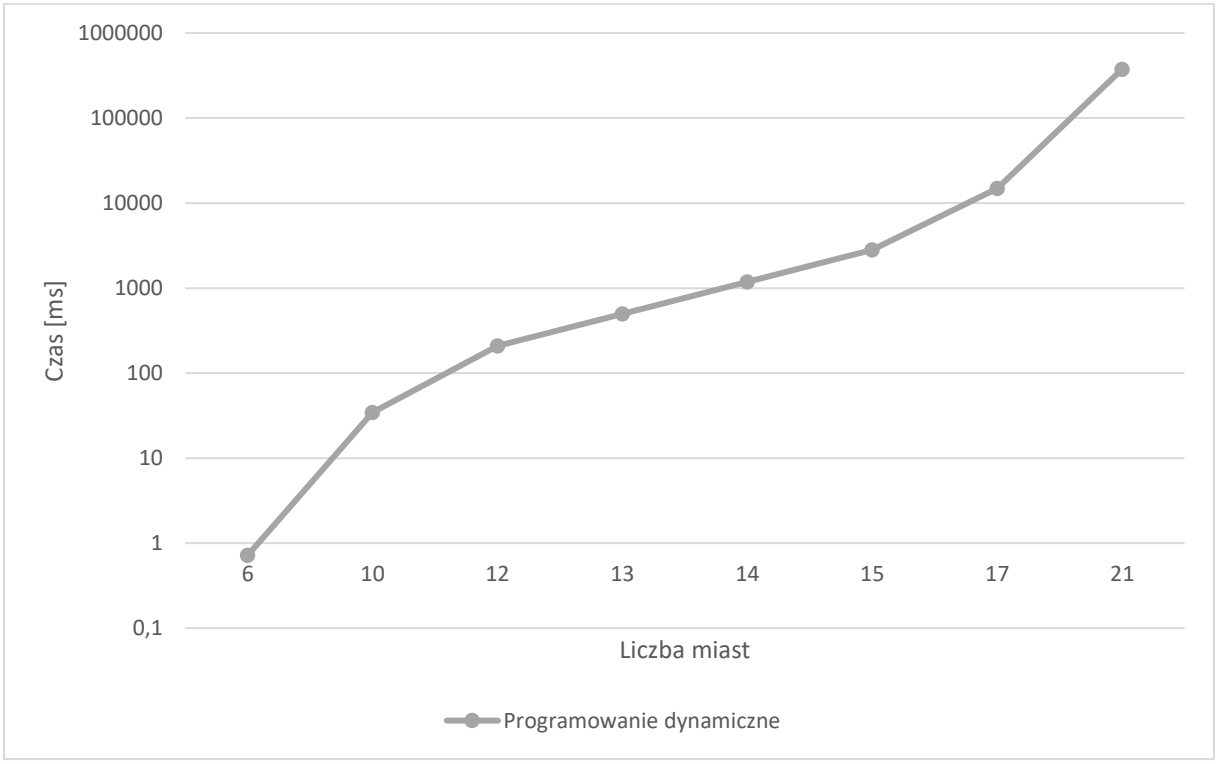
5.2. Metoda podziału i ograniczeń

	Metoda podziału i ograniczeń
6	3,790
10	387,206
12	364,677
13	662,763
14	6618,028
15	54919,940



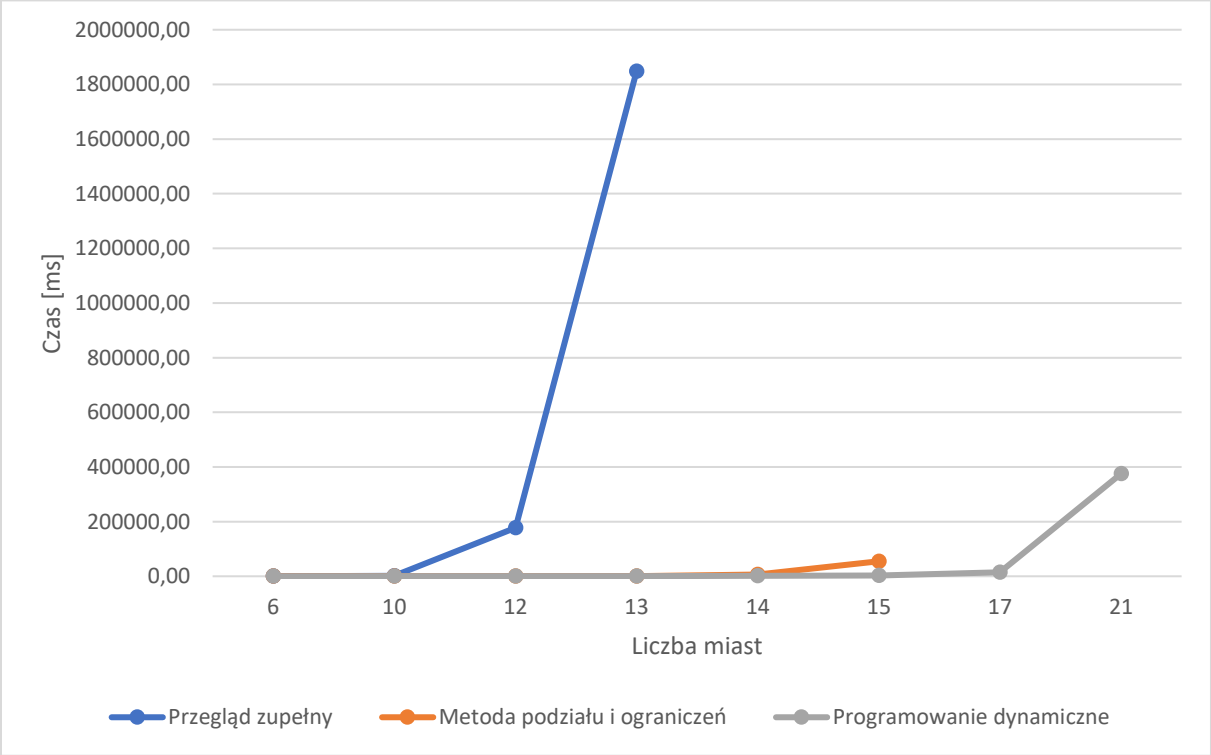
5.3. Programowanie dynamiczne

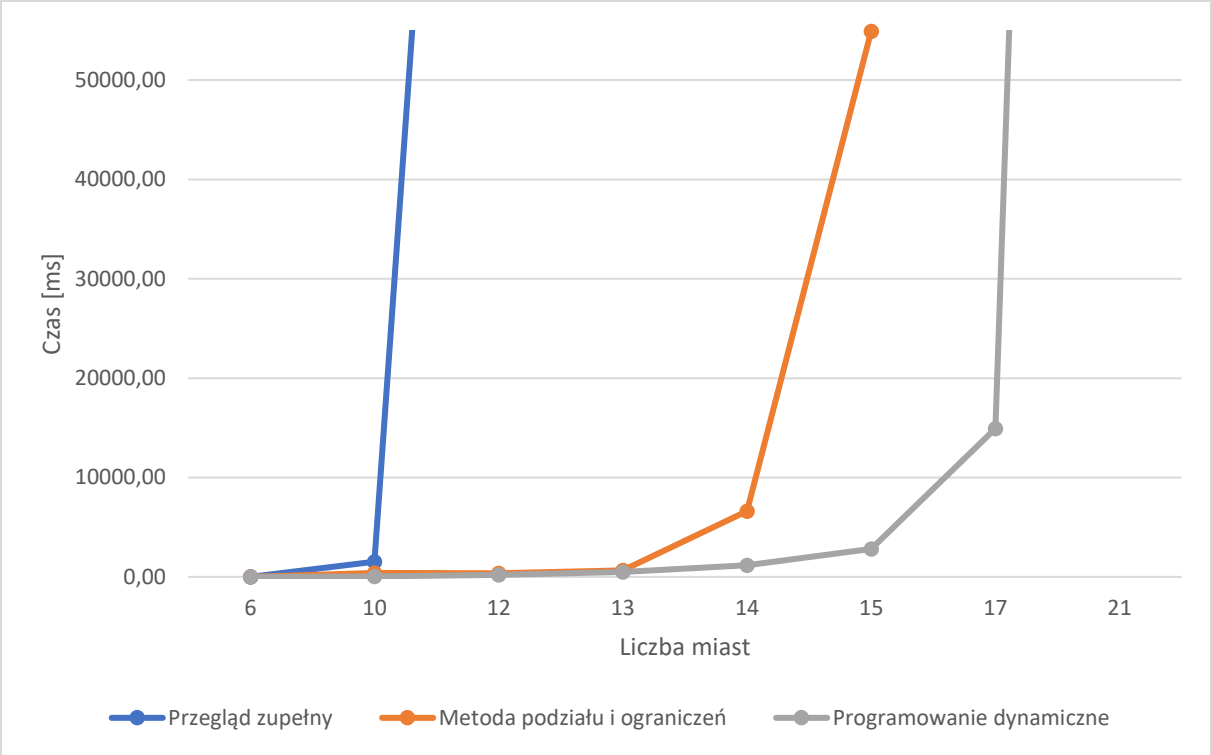
	Programowanie dynamiczne
6	0,719
10	34,495
12	210,034
13	497,689
14	1182,803
15	2827,066
17	14920,490
21	375350,500

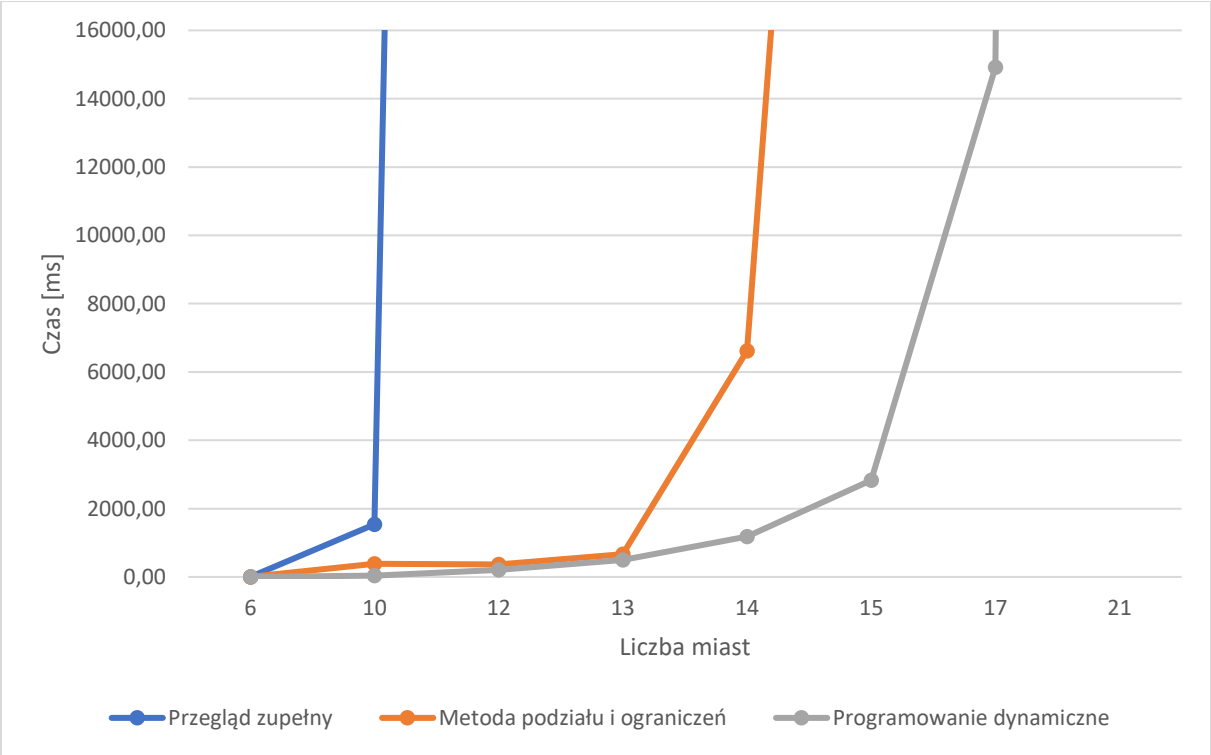


5.4. Porównanie wyników

	Przegląd zupełny	Metoda podziału i ograniczeń	Programowanie dynamiczne
6	0,630	3,790	0,719
10	1534,495	387,206	34,495
12	177155,000	364,677	210,034
13	1848010,000	662,763	497,689
14		6618,028	1182,803
15		54919,940	2827,066
17			14920,490
21			375350,500







6. Wnioski

Przeprowadzone testy wykazały że algorytm przeglądu zupełnego jest nieefektywny już przy 12 wierzchołkach. Algorytm podziałów i ograniczeń również staje się nieefektywny dla 15 wierzchołków oraz do wykonywania obliczeń wykorzystuje więcej pamięci. Najbardziej efektywną metodą okazała się metoda programowania dynamicznego, która w porównaniu do pozostałych algorytmów ma dużo mniejszą złożoność obliczeniową i pamięciową.