Politechnika Wrocławska

Projektowanie efektywnych algorytmów

Rozwiązywanie problemu komiwojażera (TSP) za pomocą różnych metod

Data: 03.12.2024r

Wykonawca: Jędrzej Radłowski 272927

Spis treści

Opis problemu

Problem komiwojażera jest zagadnieniem optymalizacyjnym, należy do klasy problemów NP-trudnych, co oznacza, że znalezienie rozwiązania musi mieć złożoność przynajmniej wykładniczą. W praktyce oznacza to, że czasy sprawdzania, a tym samym znalezienie optymalnego rozwiązania zauważalnie zwiększają się, wraz ze wzrostem rozpatrywanego problemu.

Dotyczy on zagadnienia związanego z grafami ważonymi, a polega on na znalezieniu cyklu Hamiltona o jak najmniejszej sumie wag. Sam cykl Hamiltona natomiast charakteryzuje się tym, że każdy z wierzchołków zawiera się w tym cyklu tylko raz. W odniesieniu do samej nazwy, czyli komiwojażera (osoby która odwiedza różne miasta w celu sprzedaży dóbr), polega na wyznaczeniu najkrótszej trasy dla zbioru miejscowości, tak aby każda z miejscowości została odwiedzona tylko jeden raz.

Opis zadania

W ramach sprawozdania przenalizowano efektywność wybranych metod rozwiązania problemu TSP, w szczególności analizę czasową danej metody, a co za tym idzie sytuację w której dana z metod warta jest wykonania.

Instancje danych były generowane na bieżąco do pliku .txt gdzie w pierwszym wierszu podana jest liczba wierzchołków (miast) a w kolejnych same dane jako macierz odległości. Wielkość testowanych instancji to <3;13> dla pierwszych trzech algorytmów i <3;25> dla pozostałych. Zostały one dobrane tak, aby reprezentowały różnorodne przypadki problemu – od prostych po skomplikowane. W momencie gdy programy wykonywały się szybko powtórzono dany algorytm 100 razy, natomiast w momencie gdy czas wykonywania się wydłużał powtórzenia odpowiednio malały. Dokładne dane dotyczące powtórzeń znajdują się w dalszej części sprawozdania. Poddane testom zostały zarówno symetryczne jak i asymetryczne grafy. Dzięki temu algorytmy te zostały przetestowane w różnych warunkach co pozwoli na lepsze dobranie danego algorytmu do danej liczby wierzchołków badanych dla problemu TSP. Zmniejszenie liczby instancji jest spowodowane tym, że gdyby każdy algorytmu i każda wielkości instancji została testowana 100 razy, czas potrzebny na testy wydłużyłby się znacznie, dlatego też liczba ta została zmniejszona.

Wybrane metody optymalizacyjne

W tym sprawozdaniu analizowanych będzie sześć różnych metod służących znalezieniu rozwiązania dla problemu komiwojażera.

Metoda przeglądu zupełnego

Pierwszą badaną metodą jest metoda przeglądu zupełnego. Przegląd zupełny (ang. Brute Force), w przypadku problemu komiwojażera polega na sprawdzeniu wszystkich możliwych permutacji miast i wybraniu tej, która daje minimalny koszt trasy. Metoda ta gwarantuje znalezienie rozwiązania optymalnego, jednak jej złożoność obliczeniowa wynosi O(n!), gdzie n to liczba miast. Z tego powodu przegląd zupełny jest praktyczny jedynie dla małych instancji problemu, gdyż czas obliczeń rośnie bardzo szybko wraz z liczbą miast. Warto dodać, że w samym programie użyta jest funkcja *std::next_permutation()*, która idealnie sprawdza się do tej metody, generuje ona bowiem wszystkie możliwe permutacje w porządku leksykograficznym. Porządek leksykograficzny najłatwiej zrozumieć na podstawie omówienia jego szczegółowego przypadku, czyli porządku alfabetycznego. Gdy porównujemy dwa słowa, na początku patrzymy na jego pierwszą literę i sprawdzamy, dla którego ze słów litera na pierwszej pozycji znajduje się szybciej w alfabecie. Jeżeli litery są takie same przechodzimy na następną pozycję i znowu sprawdzamy, w którym ze słów litera występująca na drugiej pozycji znajduje się szybciej w alfabecie. Proces ten powtarzamy aż do znalezienia różnicy w literach na danej pozycji w obu słowach. Rozumiejąc ten porządek, łatwo jest zrozumieć jak działa porządek leksykograficzny dla permutacji. Przykładowo, dla grafu V = {0, 1, 2} porządek leksykograficzny jego permutacji będzie występował następująco: {0, 1,2},{0,2,1},{1,0,2},{1,2,0},{2,0,1} oraz {2,1,0}.

Metoda najbliższego sąsiada

Drugą metodą, którą należało zaimplementować w ramach projektu, jest metoda najbliższego sąsiada. Metoda najbliższego sąsiada (ang. Nearest Neighbor) to algorytm heurystyczny, który rozpoczyna się w wybranym wierzchołku (mieście) i iteracyjnie wybiera najbliższe, jeszcze nieodwiedzone miasto, aż wszystkie miasta zostaną odwiedzone. Jest ona przykładem algorytmu zachłannego, który cechuje się właśnie dokonywaniem wyboru lokalnie optymalnego. W przeciwieństwie do algorytmu przeglądu zupełnego, metody heurystyczne nie muszą zwracać optymalnego rozwiązania.Implementacja tej metody w programie wygląda następująco. Jako wierzchołek początkowy wybierany jest ten o numerze 0, a następnie według heurystyki najbliższego sąsiada wybierana jest krawędź o najniższej wadze, która prowadzi do wierzchołka jeszcze nieodwiedzonego. W taki sposób algorytm działa do momentu, gdy wszystkie wierzchołki zostaną odwiedzone. Algorytm zapisuje wtedy ścieżkę i koszt przejścia, i porównuje do dotychczasowej najlepszej (na tym momencie koszt przejścia będzie na pewno lepszy od dotychczasowego, ponieważ jest to pierwsze przejście algorytmu). Proces ten jest powtarzany dla każdego z wierzchołków. Przez ten zabieg teoretyczna złożoność obliczeniowa wynosi $O(n^3)$.

Metoda losowa

Metoda ta wykorzystuje generator liczb pseudolosowych w celu wygenerowania kolejnych wierzchołków, które mają być odwiedzone. Oparty jest on na algorytmie Mersenne Twister, który jest zainicjalizowany aktualnym czasem systemowym. Do generowania liczb całkowitych z przedziału [0, n-1] użyto rozkładu *std::uniform_int_distribution<int> dist(0, n - 1)*, co gwarantuje równomierne prawdopodobieństwo wylosowania każdej liczby z podanego przedziału. Samo działanie algorytmu wygląda następująco. Na początku losowany jest wierzchołek startowy, dodawany jest on do wierzchołków odwiedzonych, a jako aktualny koszt ustawiane jest 0. Następnie, z listy wierzchołków nieodwiedzonych losowany jest kolejny wierzchołek, ten też ustawiany jest jako odwiedzony a koszt przejścia zwiększony jest o wagę krawędzi między tym a poprzednim wierzchołkiem. Tak program kontynuuje działanie do momentu odwiedzenia wszystkich wierzchołków. Koszt przejścia jak i ścieżka są zapisywane, tak aby można je było porównywać z kolejnymi wynikami metody losowej. Program całkowicie zatrzymuje się w momencie gdy zostało znalezione rozwiązanie w granicach 10% od rozwiązania optymalnego lub gdy minęło 30 minut, lub gdy zostało wykonane c powtórzeń. Jedno przejście takiego algorytmu ma złożoność O(n), natomiast przez fakt dodania innych ograniczeń, całkowita złożoność obliczeniowa zależna jest od parametru c i wynosi O(c * n).

Metoda podziału i ograniczeń

Metoda ta jest ulepszeniem metody przeglądu zupełnego. Metoda podziału i ograniczeń (ang. Branch and bound) polega na ignorowaniu ścieżek, których przewidywany koszt jest większy niż wartość aktualnie najlepszej ścieżki. Innym problemem, gdzie metoda ta znajduje swoje zastosowanie jest np. problem plecakowy. Przestrzeń rozwiązań można reprezentować w postaci drzewa, którego liście oznaczają dopuszczalne rozwiązanie problemu. Każdy węzeł oznacza dokonania wyboru, polegającego na obliczeniu czy dany węzeł może prowadzić do rozwiązania lepszego niż aktualnie najlepsze. Z tego powodu, implementacja algorytmów B&B polega na zaimplementowaniu metody obliczenia granicy dla pojedynczego węzła jak i przeglądania przestrzeni rozwiązań.

Algorytm obliczający dolną granicę dla problemu TSP

Algorytm ten oblicza dolne ograniczenie kosztu rozwiązania w problemie komiwojażera na podstawie aktualnego stanu węzła w algorytmie. Wykorzystuje ona informacje o dotychczasowej ścieżce oraz macierz odległości między wierzchołkami, aby oszacować minimalny możliwy koszt przed kontynuowaniem dalszego przeszukiwania. Proces rozpoczyna się od przypisania wartości kosztu dotychczasowej ścieżki do zmiennej „bound”. Kolejnym krokiem jest szacowanie minimalnych kosztów przejść z każdego nieodwiedzonego miasta, dla każdego takiego miasta przeszukiwane są krawędzie prowadzące do innych nieodwiedzonych miast w celu znalezienia tej o najniższym koszcie. Koszt ten dodawany jest do zmiennej „bound”. Dzięki temu dolne ograniczenie uwzględnia zarówno koszt dotychczasowej ścieżki jak i minimalne przewidywane koszty odwiedzenia pozostałych miast.

Przeszukiwanie wszerz (Breadth First Search)

Pierwszą z metod przeszukiwania przestrzeni rozwiązań jest przeszukiwanie wszerz. Oznacza to przeszukiwanie w pierwszej kolejności wszystkich wierzchołków na poziomie drzewa, na którym obecnie się znajdujemy. Program implementujący tą metodę wykorzystuje strukturę danych o nazwie kolejka. Działa ona na podstawie FIFO (First In First Out), czyli tak jak działają kolejki znane z życia codziennego – pierwsza osoba która dołączyła do kolejki jest w pierwszej kolejności obsługiwana. W przypadku tej metody warto zauważyć, że dla szerokich drzew dojście do pierwszego wierzchołka będącego liściem może zająć bardzo dużo czasu, dlatego też w samym programie jako pierwszy koszt przejścia ustawiany jest koszt wyliczony poprzez metodę najbliższego sąsiada.

Przeszukiwanie w głąb (Depth First Search)

Drugą z metod przeszukiwania przestrzeni rozwiązań jest przeszukiwanie w głąb. Zamiast rozpatrywania wszystkich wierzchołków na danym poziomie drzewa, metoda ta dąży to rozpatrzenia liści jako pierwszych. Program implementujący tą metodę wykorzystuje strukturę danych o nazwie stos. Działa ona na podstawie LIFO (Last In First Out). Tutaj porównaniem do codziennego życia może być stos talerzy w szafce kuchennej. Nowy talerz układa się na wierzchu stosu, a w momencie gdy potrzebny jest jakiś talerz to brany jest ten z samej góry stosu. Tutaj dzięki skupieniu się na liściach w pierwszej kolejności nie jest potrzebne żadne wcześniejsze ustawienie początkowego kosztu ścieżki, zatem koszt ten w programie ustawiony jest jako maksymalna wartość dla typu int.

Przeszukiwanie pod kątem najniższego kosztu (Lowest Cost)

Ostatnią z metod przeszukiwania przestrzeni rozwiązań analizowanej w tym sprawozdaniu jest metoda, gdzie wykorzystana jest wartość dolnego ograniczenia poszczególnych węzłów. W kolejnych krokach rozpatrywania przestrzeni rozwiązań wybierany jest węzeł spośród kolekcji węzłów obiecujących, którego dolne ograniczenie posiada najmniejszą wartość. Program implementujący tą metodą wykorzystuje kolejkę priorytetową, gdzie element o najwyższym priorytecie (tutaj najniższy koszt) jest wybierany jako pierwszy do rozpatrzenia.

Hipotezy badawcze

1. Metody heurystyczne mogą działać niestabilnie w pewnych przypadkach
2. Metody B&B znacznie lepiej działają dla instancji o n > 13, gdzie n jest wielkością instancji, niż metody z pierwszej kategorii
3. Metoda losowa powinna zwracać optymalne rozwiązania do grafów do 6 wierzchołków
4. Metoda DFS powinna osiągać lepsze czasy niż metoda BFS

Specyfikacja sprzętu i procedura badawcza

Maszyna testowa wyposażona jest w procesor M3 Apple (8 rdzeni) o taktowaniu 4 x 3.2 GHz & 4 x 2 GHz. Posiada ona 16 GB zunifikowanej pamięci RAM. Badania zostały przeprowadzone w momencie gdy żaden inny program nie działał w tle, a sam komputer był podłączony do zasilania sieciowego. Same wyniki prezentowane są na sprzęcie na którym prowadzono badania.

Badania zostały przeprowadzone zarówno dla instancji symetrycznych jak i asymetrycznych. Jak już zostało wcześniej opisane testy zostały uruchomione dla instancji grafów, których liczba wierzchołków należała do przedziału <3;14> dla pierwszych trzech algorytmów i <3;25> dla reszty. Dla pierwszych trzech algorytmów testowano program stukrotnie dla wierzchołków z przedziału <3;13>, dla grafu z 14 wierzchołkami programy testowano zaledwie dwukrotnie, zatem wyniki mogą być obarczone dużym błędem pomiarowym. Dla algorytmów B&B stukrotnie testowano grafy do 16 wierzchołków włącznie. Dla grafów posiadających 17 i więcej wierzchołków algorytmy testowano pięćdziesięciokrotnie. Dla tych trzech algorytmów przyjęto maksymalny czas pojedynczej iteracji na wartość 5 minut. W przypadku gdy pojedyncza iteracja trwała dłużej niż 5 minut to została ona przerywana i nie była brana pod uwagę w wykresach. Informacja ile razy algorytm został przerwany przedstawione w Tabeli 3 oraz 4. Przyjęto takie założenia, aby uzyskać jak najbardziej wiarygodne wyniki i jednocześnie nie przedłużyć czasu prowadzenia testów w nieskończoność. Czasy, optymalna ścieżka jak i koszt przejścia zostały zapisywane do plików .csv.

Wyniki

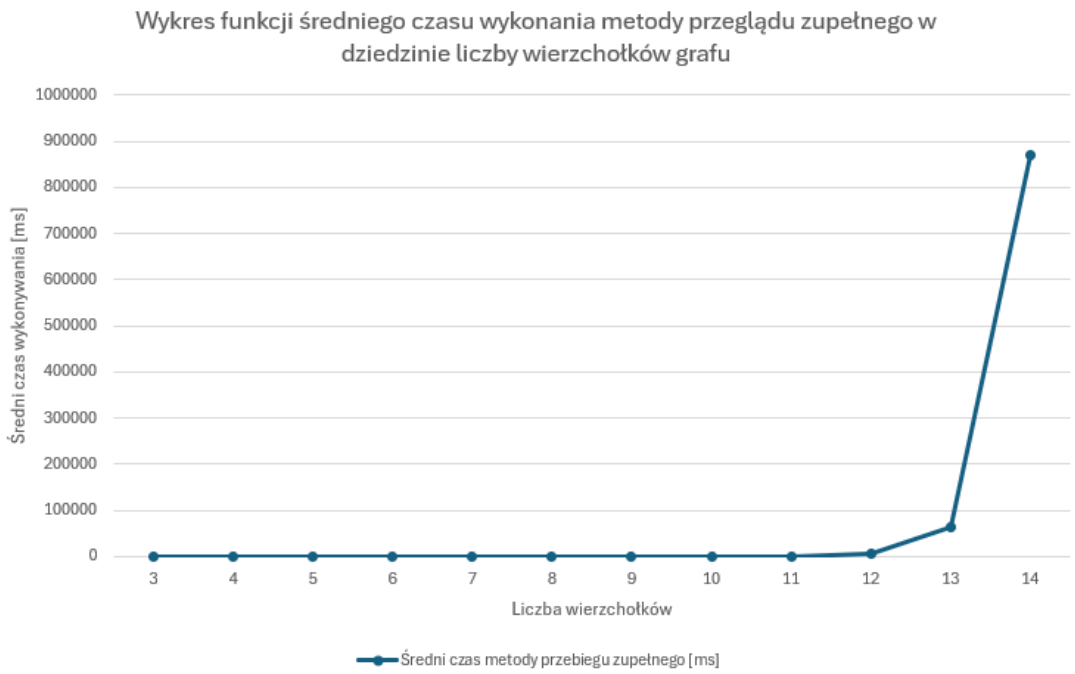
Zadania pierwszej kategorii (brute-force, nearest-neighbour, random)

Grafy symetryczne

Tabela 1. Średnia czasów wykonania oraz różnica procentowa wybranych algorytmów dla grafów symetrycznych

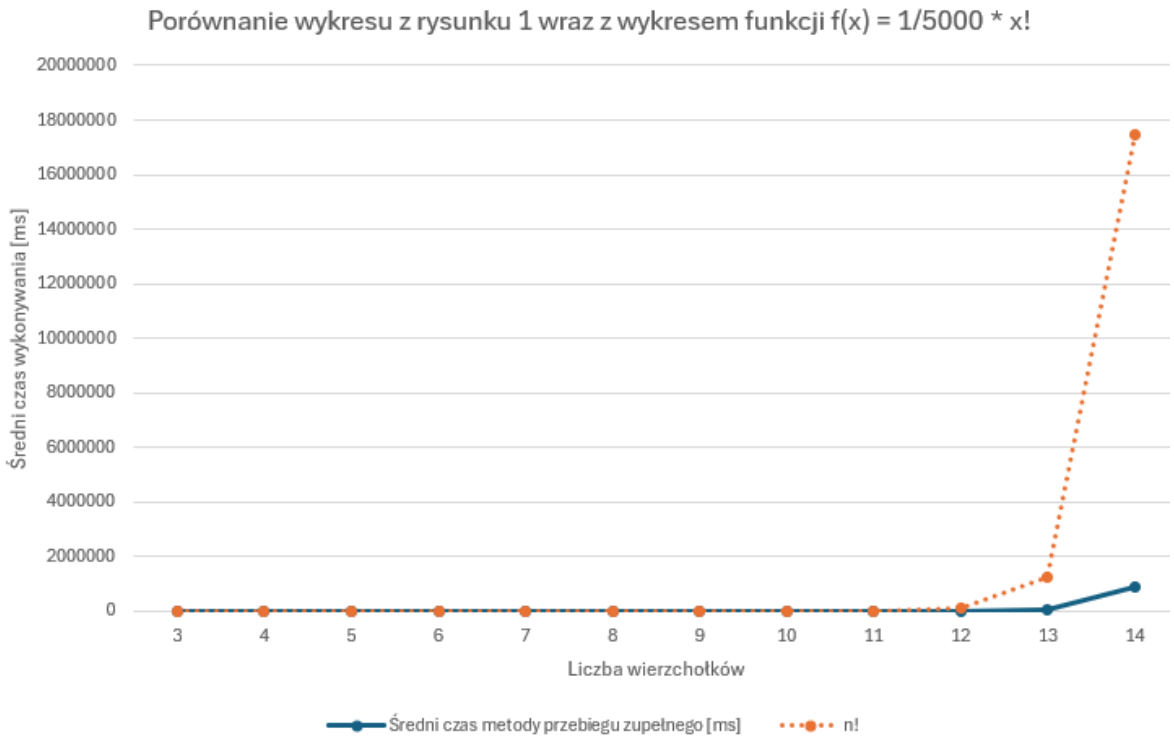
Liczba wierzchołków	Średni czas metody przebiegu zupełnego [ms]	Średni czas metody najbliższego sąsiada [ms]	Średni czas metody losowej [ms]	Różnica procentowa między optymalnym rozwiązaniem a metodą najbliższego sąsiada	Różnica procentowa między optymalnym rozwiązaniem a metodą losową
3	0,001136	0,01971959	0,01478005	0	0,0%
4	0,001559	0,0348479	0,05827038	0	0,0%
5	0,00391	0,06556882	0,04053746	0	0,0%
6	0,014614	0,07655088	0,06360998	0	0,0%
7	0,080867	0,1079921	1,22865443	0,70%	0,0%
8	0,570499	0,14716301	0,02931917	2,30%	0,0%
9	4,612742	0,20048455	0,60142	4,21%	0,0%
10	43,128091	0,25488326	4,12692	5,79%	0,0%
11	446,10547	0,32018953	1,89423	7,43%	8,9%
12	5087,135789	0,41814789	2,1948	8,92%	13,7%
13	65058,39116	0,4821458	2,3725	11,74%	14,6%
14	869354,2	0,72291295	2,59461	15,25%	18,3%

Tabela 1 przedstawia średnie czasy wykonania metod przeglądu zupełnego, najbliższego sąsiada oraz losowej w zależności od rozmiaru problemu, który jest tutaj wyznaczony jako rozmiar instancji. Grafy na których przeprowadzone zostały testy były grafami symetrycznymi, o liczbie wierzchołków z przedziału <3,14>. Dla grafów do 10 wierzchołków włącznie metoda losowa działała do momentu znalezienia optymalnego rozwiązania, a dla dalszych procentowa różnica była ustawiona na odpowiednio : 10%, 15%, 15%, 20%. Zdecydowano się na takie rozwiązanie, by znacząco skrócić czas prowadzony na badania i jednocześnie pokazać, że metoda ta sprawdza się nie najgorzej w momencie gdy nie interesuje nas rozwiązanie optymalne, lecz takie które będzie mu bliskie procentowo. Warto zauważyć, że metoda najbliższego sąsiada, dla wszystkich rozmiarów instancji poddanych testom trwała krócej niż 1 milisekundę, jednocześnie generując rozwiązania bliskie do optymalnego.



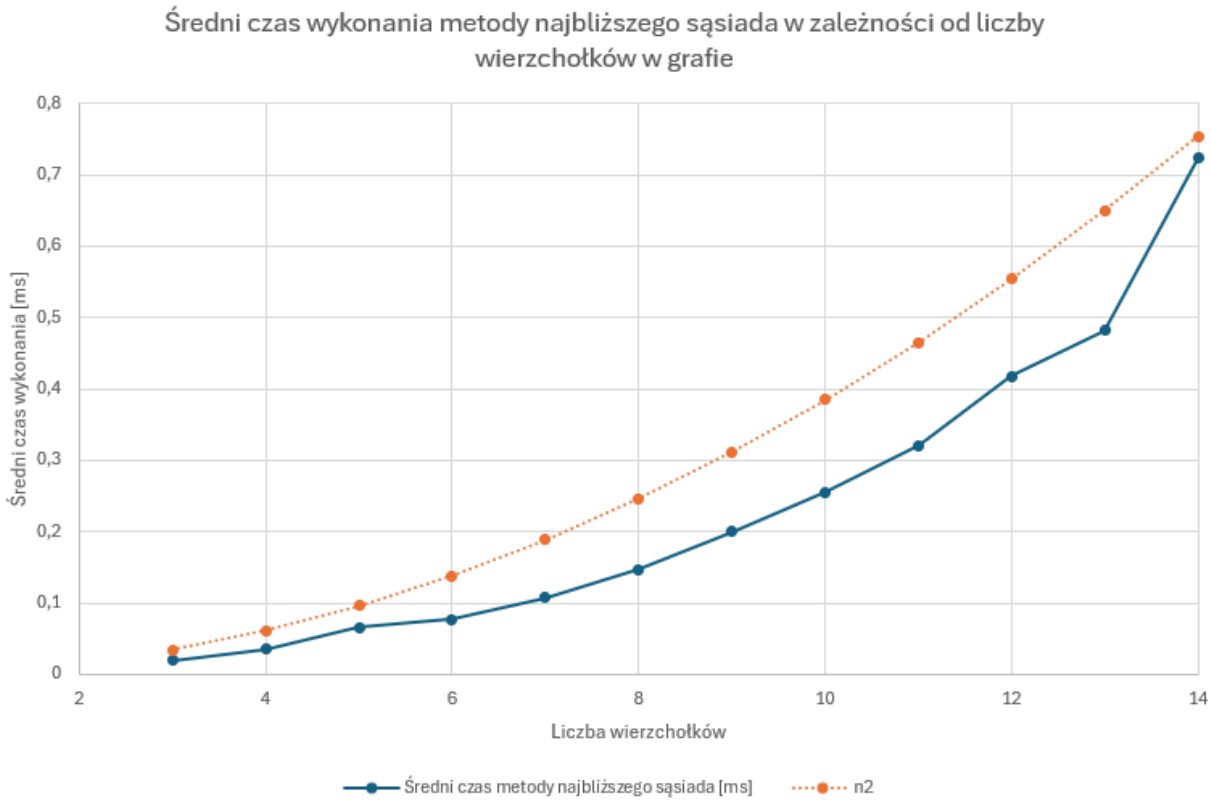
Rysunek 1. Wykres funkcji średniego czasu wykonania metody przeglądu zupełnego w dziedzinie liczby wierzchołków grafu

Rysunek 1 przedstawia wykres funkcji średniego czasu wykonania dla metody bruteforce w zależności od liczby wierzchołków grafu. Dla grafów posiadających 10 wierzchołków czasy wykonania różnią się nieznacznie. Różnicę widać dla grafów z przedziału <11,14>. Tutaj czasy wykonania rosną gwałtownie. Dla grafu o 14 wierzchołkach bowiem średni czas wyniósł prawie 15 minut.



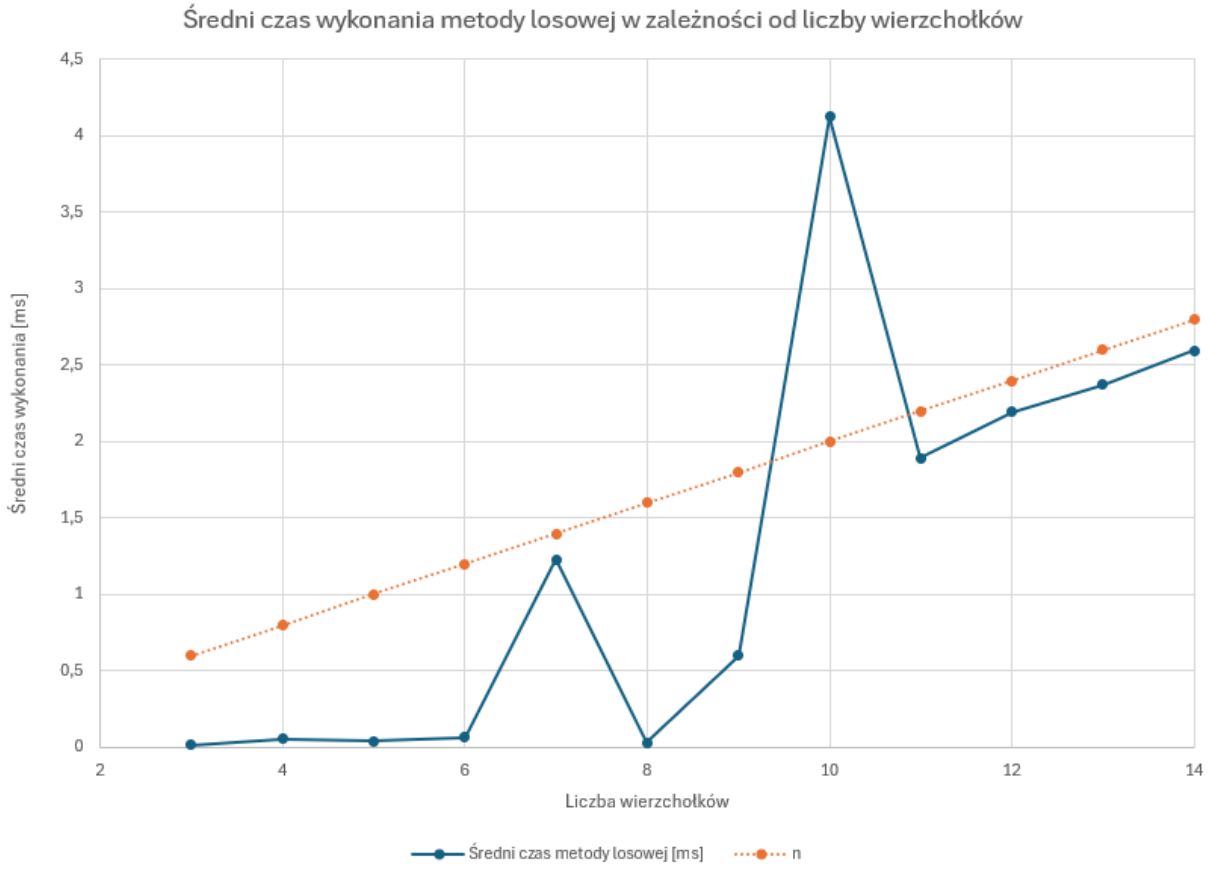
Rysunek 2. Porównanie wykresu z rysunku 1 do wykresu funkcji f(x) = 1/5000 * n

Złożoność teoretyczna dla metody przeglądu zupełnego wynosi O(n!), więc na podstawie Rysunku 1 można przypuszczać, że wyniki otrzymane z testów potwierdzają teorię. Żeby jednak mieć stuprocentową pewność, na Rysunku 2 przedstawiono porównanie średnich czasów wykonania z wykresem funkcji danej równaniem f(x) = $\frac{1}{5000}x!$. Jak można zauważyć na tym samym wykresie, funkcja f(x) = $\frac{1}{5000}x!$ Jest górnym ograniczeniem funkcji zaznaczonej niebieską linią, w związku z czym założenie o złożoności obliczeniowej metody przeglądu zupełnego równej O(n!) jest jak najbardziej prawdziwe.



Rysunek 3. Wykres przedstawiający funkcję średniego czasu wykonania metody najbliższego sąsiada w dziedzinie liczby wierzchołków grafu w porównaniu do wykresu funkcji f(x) = $\frac{x^2}{260}$

Rysunek 3 przedstawia wykres funkcji średniego czasu wykonania metody najbliższego sąsiada w zależności od liczby wierzchołków w grafie. Funkcja ta została zaznaczona linią w kolorze niebieskim. Pomarańczowa linia przerywana to wykres funkcji danej równaniem f(x) = $\frac{x^2}{260}$. Złożoność teoretyczna metody najbliższego sąsiada wynosi O(n^2), co potwierdzają dane przedstawione na Rysunku 3.



Rysunek 4. Wykres przedstawiający funkcję średniego czasu wykonania metody losowej w dziedzinie liczby wierzchołków w porównaniu z wykresem funkcji f(x) = 5x

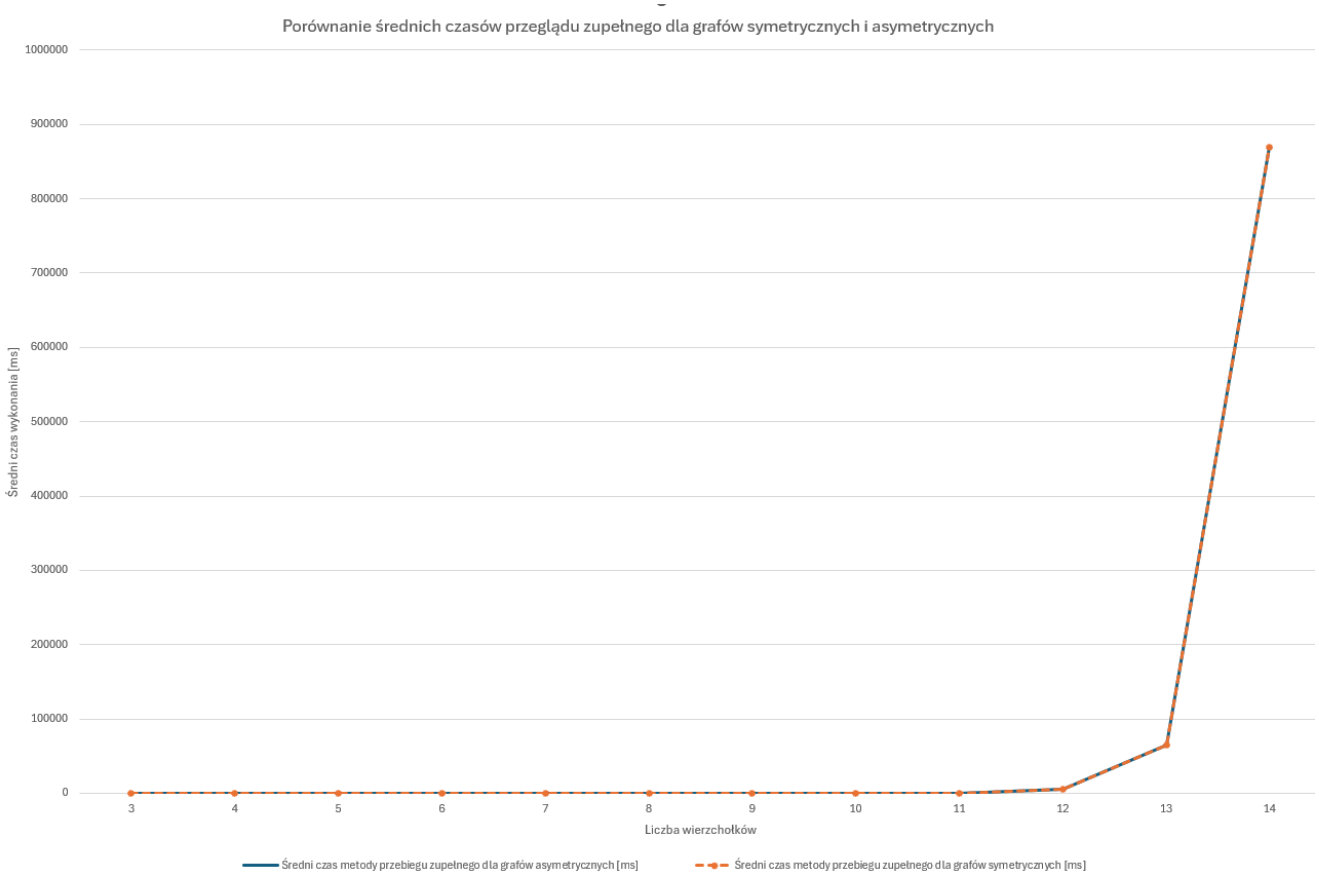
Rysunek 4 wykres funkcji średniego czasu wykonania metody losowej w zależności od liczby wierzchołów grafu wraz z przebiegiem funkcji danej równaniem f(x) = 5x. Hipoteza badawcza dotycząca metody losowej mówiła o znajdowaniu optymalnego rozwiązania w rozsądnym czasie dla grafów z liczbą wierzchołków z przedziału <3;6>. Tutaj wyniki testów pozytywnie mnie zaskoczyły, ponieważ czas wykonania drastycznie wzrósł dopiero dla grafów o 10 wierzchołkach. Później zostały zastosowane ograniczenia dla algorytmu, które widać w Tabeli 1, dlatego też od 11 wierzchołków funkcja ponownie zostaje objęta górnym ograniczeniem wykresem funkcji przedstawionej linią pomarańczową. Dla grafów o 7 wierzchołkach metoda losowa wykonuje się znacznie dłużej w porównaniu do wcześniejszej liczby wierzchołków, a później znowu czas wykonania drastycznie spada. Testy dla tej instancji zostały przeprowadzone ponownie, lecz cały czas ta „anomalia” występuję, co ciężko mi wytłumaczyć w jakikolwiek sposób. Jeżeli chodzi o teoretyczną złożoność obliczeniową, to wykres z Rysunku 4 potwierdza teorię, oprócz instancji o 10 wierzchołkach. Jest to oczywiście spowodowane jedną z hipotez badawczych i moją chęcią do sprawdzenia limitów tej metody dla znajdowania rozwiązania optymalnego. Gdybym szybciej włączył dodatkowe ograniczenia wykonania algorytmu z pewnością wszystkie z wyników znajdowały by się poniżej funkcji oznaczonej linią pomarańczową.

Grafy asymetryczne

Tabela 2. Średnia czasów wykonania oraz różnica procentowa wybranych algorytmów dla grafów asymetrycznych

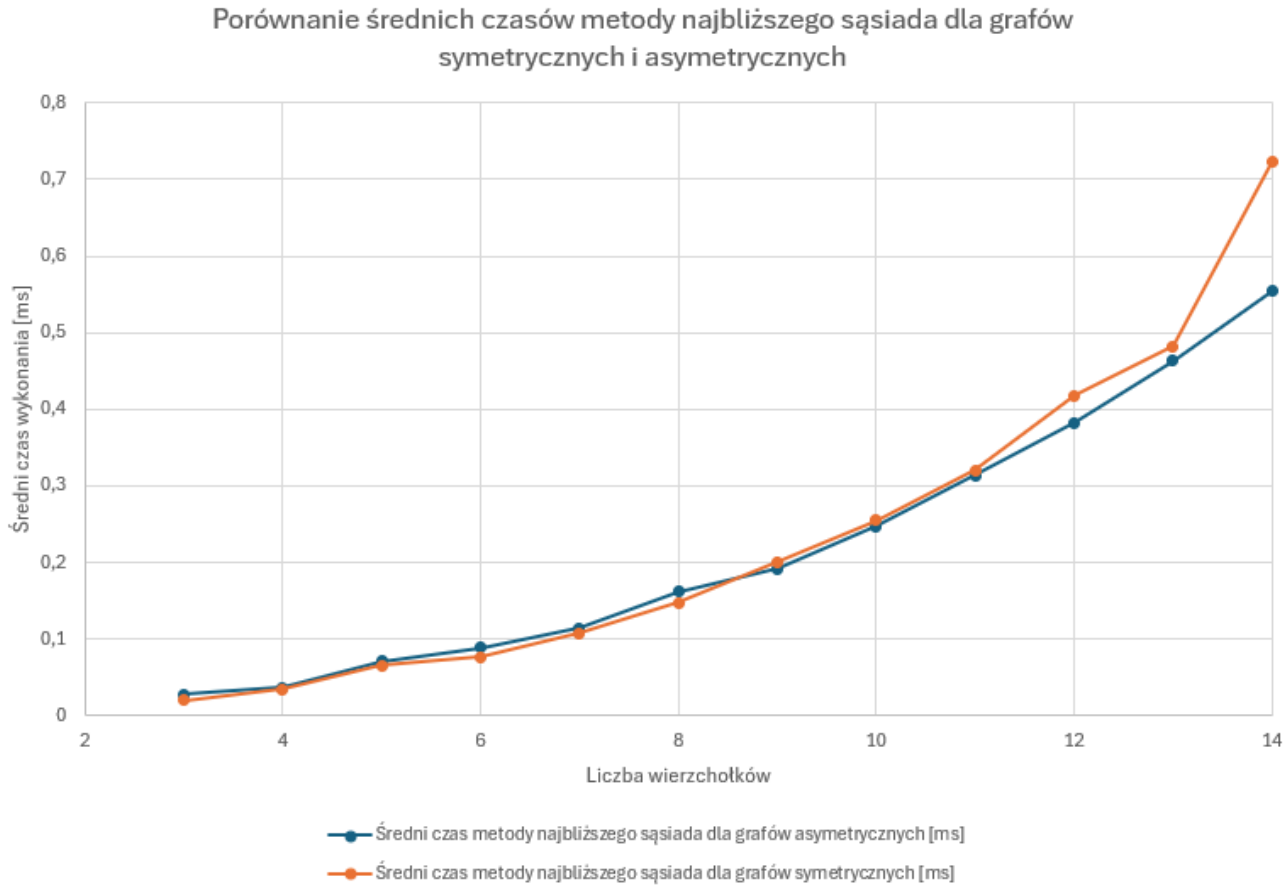
Liczba wierzchołków	Średni czas metody przebiegu zupełnego [ms]	Średni czas metody najbliższego sąsiada [ms]	Średni czas metody losowej [ms]	Różnica procentowa między optymalnym rozwiązaniem a metodą najbliższego sąsiada	Różnica procentowa między optymalnym rozwiązaniem a metodą losową
3	0,0023	0,02834454	0.01477285	0	0,0%
4	0,002415	0,03732081	0.03120336	0	0,0%
5	0,005779	0,07177165	0.09145414	0	0,0%
6	0,017085	0,08879003	0.47997465	0	0,0%
7	0,086834	0,11451289	0.44510046	2%	0,0%
8	0,557467	0,16233761	0.49235	4%	0,0%
9	4,626258	0,19225292	0.54762334	6%	0,0%
10	43,323400	0,24697577	3.5829548	6,30%	0,0%
11	447,00198	0,31379999	7.1242479	7,20%	9,5%
12	5123,6511	0,38241121	1.7281334	9,50%	14,7%
13	65278,275	0,4630979	4.1737234	13,45%	14,9%
14	869813,5	0,55401085	2.0819636	17,64%	19,5%

Tabela 2 przedstawia średnie czasy wykonania metod przeglądu zupełnego, najbliższego sąsiada oraz losowej w zależności od rozmiaru problemu, który jest tutaj wyznaczony jako rozmiar instancji. Grafy na których przeprowadzone zostały testy były grafami asymetrycznymi, o liczbie wierzchołków z przedziału <3,14>. Dla grafów do 10 wierzchołków włącznie metoda losowa działała do momentu znalezienia optymalnego rozwiązania, a dla dalszych procentowa różnica była ustawiona na odpowiednio : 10%, 15%, 15%, 20%. Ponownie zdecydowano się na takie rozwiązanie, by znacząco skrócić czas prowadzony na badania i jednocześnie pokazać, że metoda ta sprawdza się nie najgorzej w momencie gdy nie interesuje nas rozwiązanie optymalne, lecz takie które będzie mu bliskie procentowo. Tutaj tak samo jak poprzednio, średni czas metody najbliższego sąsiada jest krótszy niż 1ms.



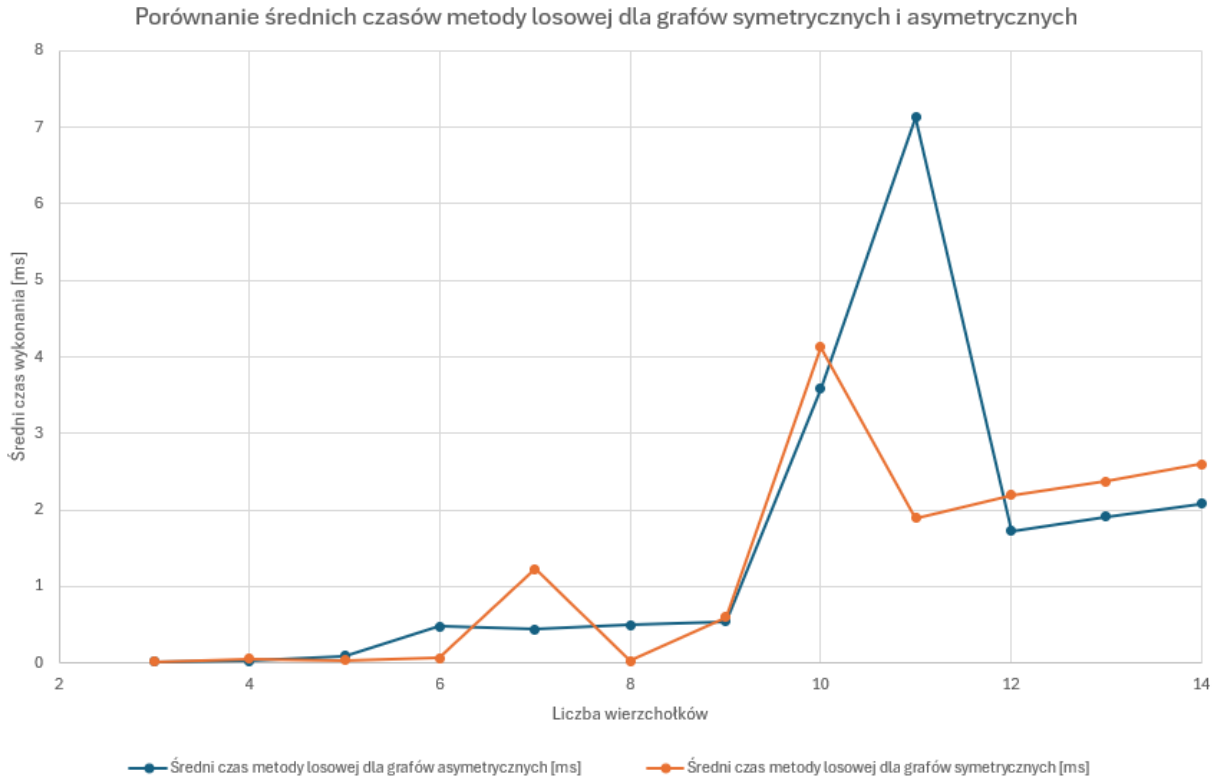
Rysunek 5. Porównanie średnich czasów metody przeglądu zupełnego dla grafów symetrycznych i asymetrycznych

Jak widać na Rysunku 5 wyniki są praktycznie identyczne, o czym świadczą pokrywające się linie. Jest to zgodne z teorią, mianowicie algorytm nie ma wprowadzonej żadnej dodatkowej implementacji dla typu grafu, zatem same wyniki są do siebie bardzo podobne.



Rysunek 6. Porównanie średnich czasów metody najbliższego sąsiada dla grafów symetrycznych i asymetrycznych

Na rysunku 6 przedstawione są średnie czasy działania metody najbliższego sąsiada. Tutaj podobnie jak dla metody przeglądu zupełnego wyniki nie różnią się znacząco. Oznacza to, że symetryczność grafu nie ma znaczenia w przypadku złożoności obliczeniowej dla tej metody.



Rysunek 7. Porównanie średnich czasów metody losowej dla grafów symetrycznych i asymetrycznych

Na rysunku 7 przedstawiono średnie czasy działania metody losowej. Tutaj podobnie, różnice są niewielkie z wyjątkiem instancji o 7 i 11 wierzchołkach. Dla grafów asymetrycznych nie zauważono „anomalii", która pojawiła się na rysunku 4, czyli znacznego wzrostu średniego czasu metody losowej dla grafów o 7 wierzchołkach, co powoduje, że jeszcze trudniej mi wytłumaczyć powód pojawienia się jej dla grafów symetrycznych. Dla grafów o 11 wierzchołkach natomiast, powtórzone są testy sprawdzające ile czasu zajmie znalezienie optymalnego rozwiązania dla metody losowej. Zdecydowano się na taki zabieg, z powodu minimalnie lepszych czasów metody losowej dla grafów asymetrycznych. Widać też, że jest to wzrost liniowy, co znowu potwierdza teoretyczną złożoność obliczeniową. Dla grafów asymetrycznych o wierzchołkach z przedziału <12,14> zastosowano takie same ograniczenia jak dla grafów symetrycznych, co potwierdzają wyniki na wykresie.

Zadania drugiej kategorii (B&B)

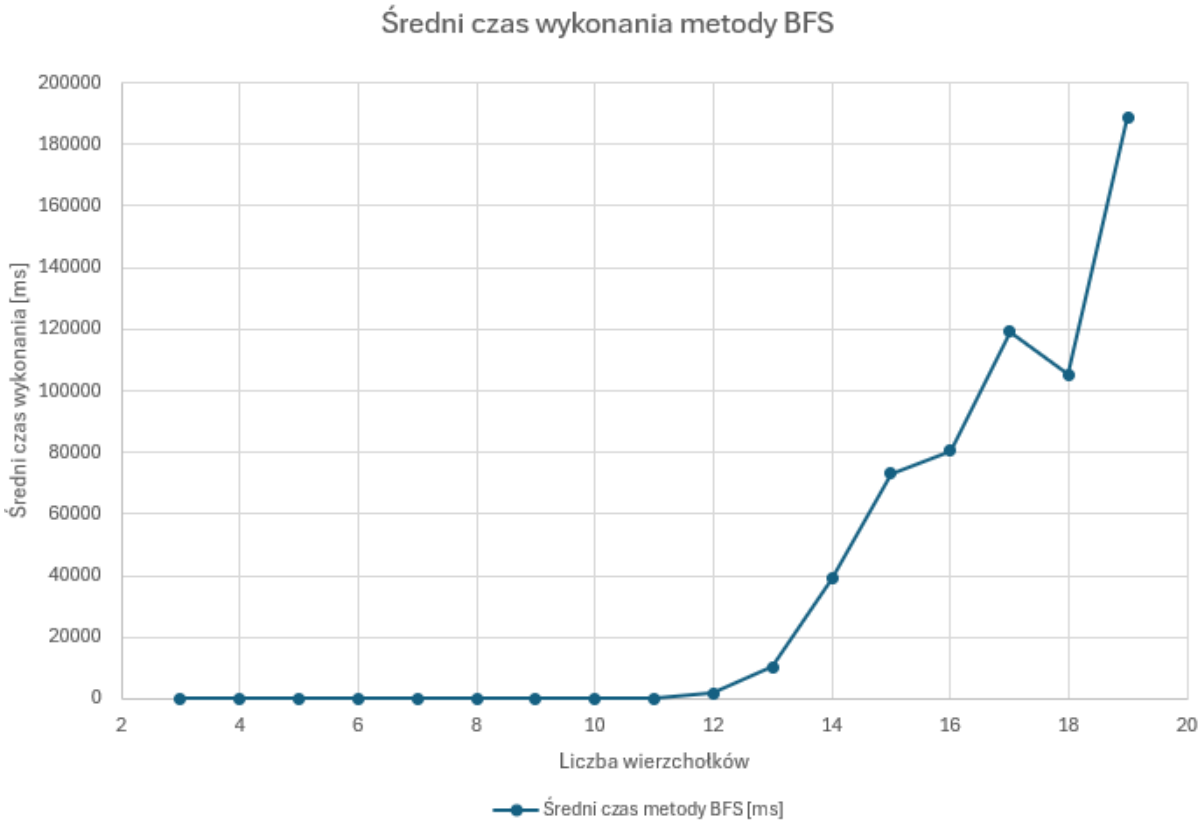
Grafy symetryczne

Tabela 3. Średnie czasy wykonania poszczególnych metoda dla grafów symetrycznych

Liczba wierzchołków	Średni czas metody BFS [ms]	Średni czas metody DFS [ms]	Średni czas LC [ms]	Procent błędów BFS	Procent błędów DFS	Procent błędów LC
3	0,00713252	0,016089921	0,00721664	0	0	0
4	0,02215422	0,034346099	0,02644532	0	0	0
5	0,05589835	0,073710446	0,10481127	0	0	0
6	0,2636167	0,21418404	0,44058655	0	0	0
7	0,12284958	0,285350634	2,0799737	0	0	0
8	2,3469969	0,586445505	3,786387474	0	0	0
9	6,7585737	2,533996633	20,27574667	0	0	0
10	6,4594125	6,004994747	46,53813571	0	0	0
11	108,0256375	19,07608257	126,270078	0	0	0
12	1927,214964	19,92031782	816,57236	0	0	0
13	10386,99488	31,2904396	3495,49344	1	0	0
14	39029,48481	147,0201139	14281,43221	3	0	0
15	73082,832	468,3222931	37585,9258	10	0	0
16	80422,91	982,0299327	81167,91429	32	0	30
17	119347,68	3381,618209	176130,125	50	0	40
18	105152,3	12778,21362	221532	80	0	80
19	188830	20310,52484	300000	90	0	100
20	x	32911,02745	x	x	0	x
21	x	45279,17568	x	x	0	x
22	x	103727,8667	x	x	16	x
23	x	172580,8	x	x	25	x
24	x	208417	x	x	33	x
25	x	232732,3333	x	x	40	x

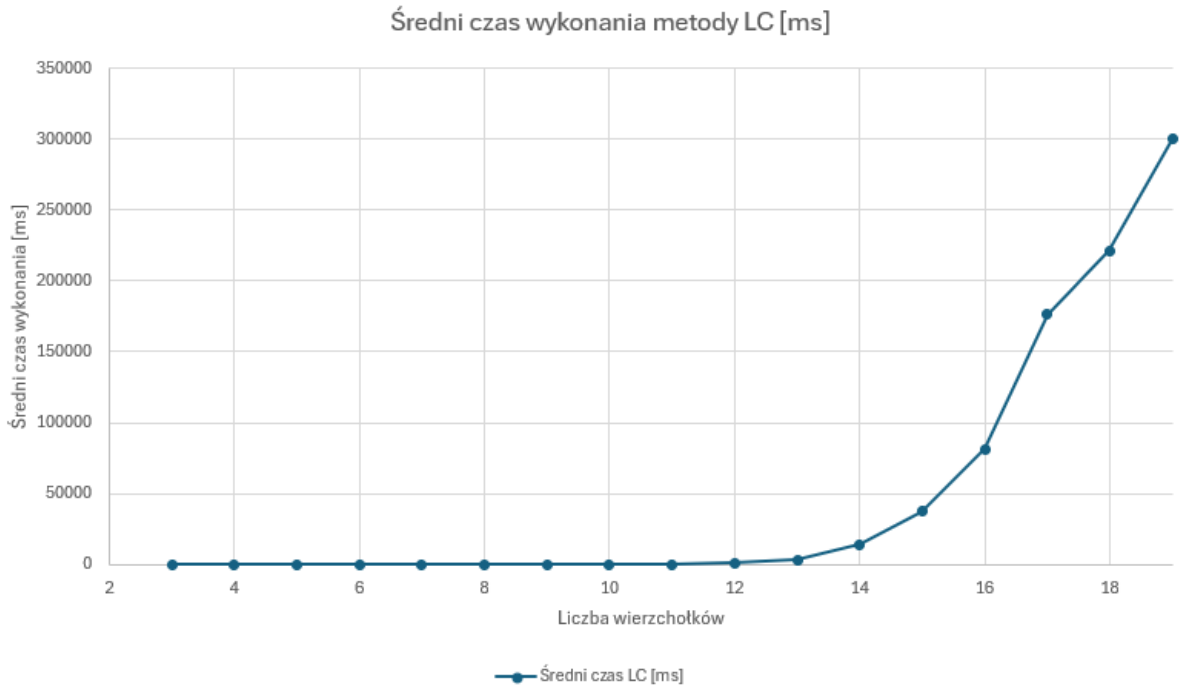
Tabela 3 przedstawia średnie czasy wykonania algorytmu podziału i ograniczeń dla różnych metod przeszukiwania przestrzeni rozwiązań. Tabela ta posiada także informacje ile iteracji zostało przerwanych z powodu przekroczenia czasu (limit 5 minut). Wartości „x” w tabeli oznaczają brak pomiaru. Tak jak wcześniej było wspomniane, metody BFS i LC zostały

testowane dla instancji o liczbie wierzchołków z przedziału <3,19>, a metoda DFS dodatkowo była testowana dla grafów o liczbie wierzchołków z przedziału <20,25>.



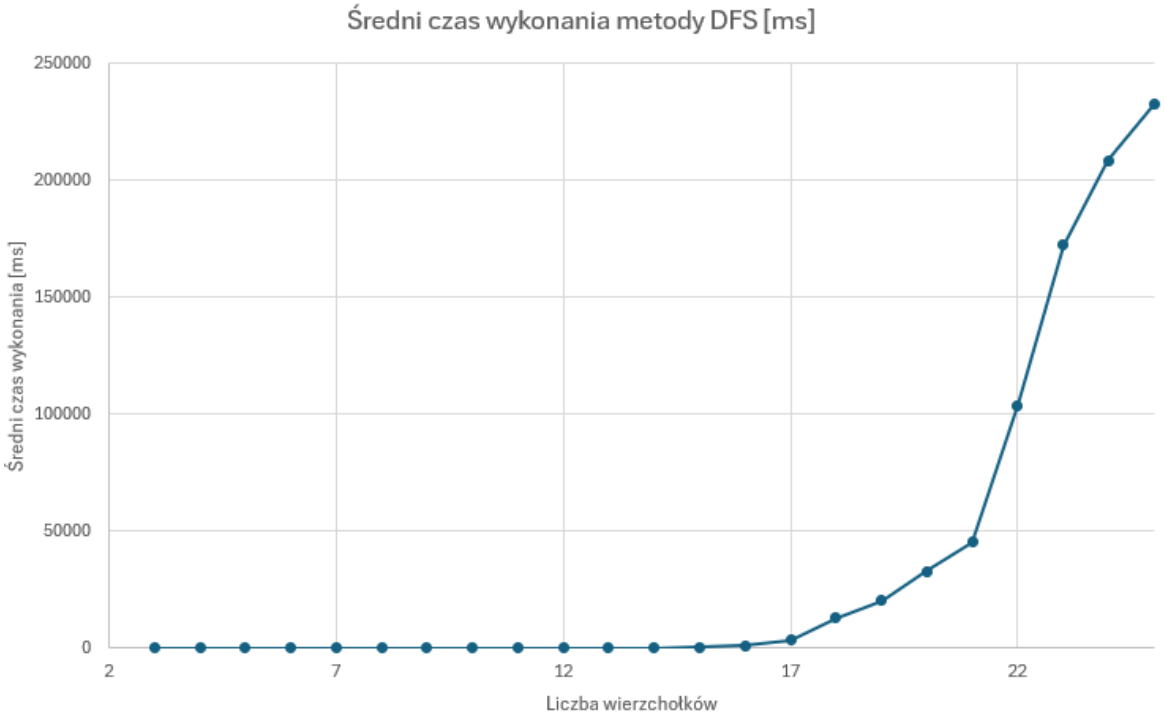
Rysunek 8. Średni czas wykonania metody BFS dla grafów symetrycznych

Na Rysunku 8 widać wykres funkcji średniego czasu wykonania algorytmu metody wszerz. Dla grafów z liczbą wierzchołków z przedziału <3;13> czasy wykonania są nieznaczne, natomiast później rosną gwałtownie, tak że dla grafów o 19 wierzchołkach średni czas trwania wyniósł ponad 3 minuty. Dodatkowo jak widać w Tabeli 3 aż 90% wszystkich prób zostało przerwanych z powodu przekroczenia limitu 5 minut.



Rysunek 9. Średni czas wykonania metody LC dla grafów symetrycznych

Rysunek 9 przedstawia wykres funkcji średniego czasu wykonania algorytmu metody LC. Tutaj, średni czas wykonania zaczął gwałtownie rosnąc dla grafów o większej liczbie wierzchołków niż dla metody BFS, lecz ostateczny średni czas wyniósł o prawie 2 minuty więcej niż w metodzie poprzedniej. Tutaj bowiem, żadna z prób wykonania algorytmu dla grafów i liczbie wierzchołków równej 19 nie zakończyła się sukcesem, wszystkie zostały zakończone z powodu przekroczenia limitu czasu.



Rysunek 10. Średni czas wykonania metody DFS dla grafów symetrycznych

Jak widać na rysunku 10, metoda DFS cechuje się najlepszymi czasami wśród trzech testowanych metod podziału i ograniczeń. Tutaj bowiem, czasy powyżej 5 minut zaczęły się pojawiać dopiero dla grafów o liczbie wierzchołków równej 22. Dla grafów o liczbie wierzchołków z przedziału <3,21> algorytm ten wykonuje się bardzo szybko. Dla grafów o 25 wierzchołkach średni czas wyniósł prawie 4 minuty, gdzie 40% przypadków testowania algorytmu zakończyło się z powodu przekroczeniu limitu ustawionego na 5 minut. Uważam, że jest to nienajgorszy wynik, szczególnie w porównaniu z dwoma pozostałymi metodami B&B.

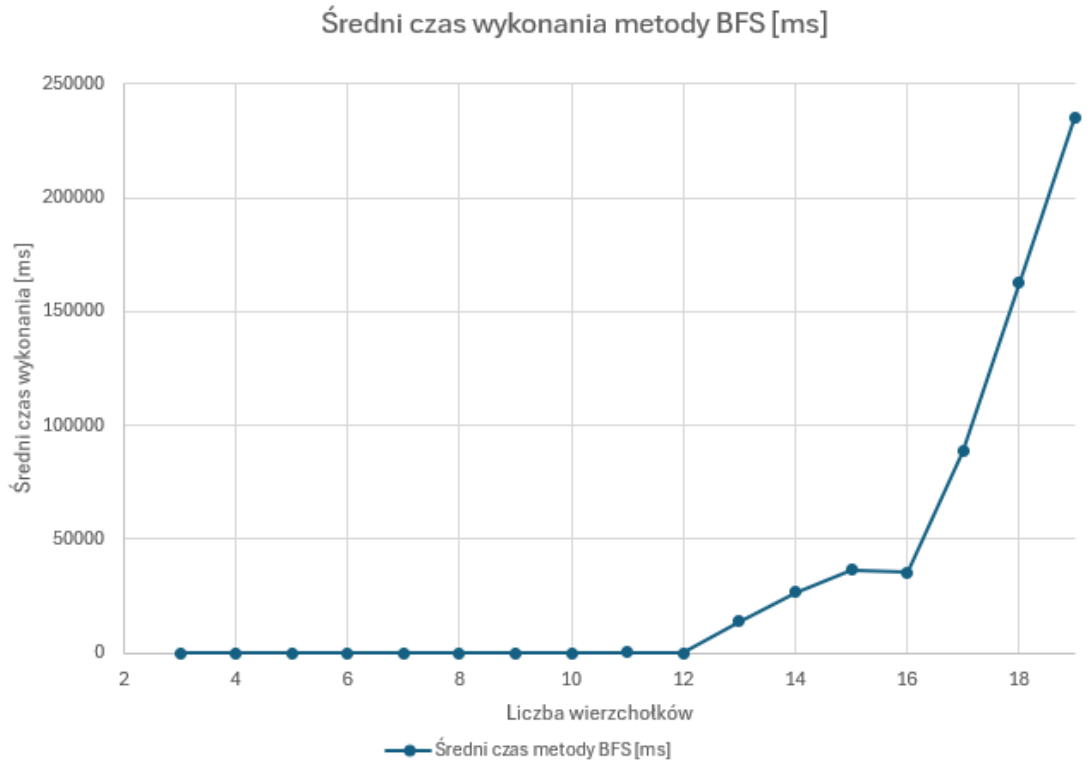
Wszystkie wykresu funkcji dla metod podziału i ograniczeń są podobne do siebie wyglądem, jedynym aspektem, w którym się różnią są same wyniki. Metoda DFS bowiem osiągnęła czasy o rzędy wielkości mniejsze niż metoda BFS i LC. Warto też wspomnieć, że metoda ta wykazała się także najlepszą złożonością pamięciową, ta wyniosła od kilku do kilkunastu MB w zależności od rozmiaru instancji. Dla metody BFS złożoność ta dla największych instancji wyniosła nawet kilka GB, a metoda LC około 700MB.

Grafy asymetryczne

Tabela 4. Średnie czasy wykonania poszczególnych metoda dla grafów asymetrycznych

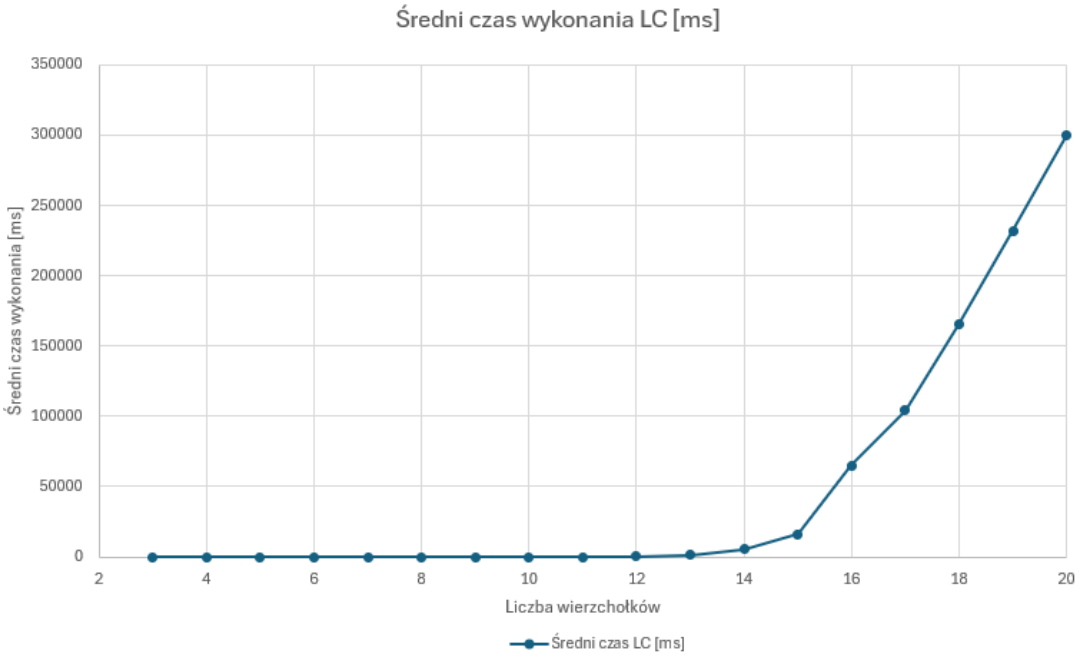
Liczba wierzchołków	Średni czas metody BFS [ms]	Średni czas metody DFS [ms]	Średni czas LC [ms]	Procent błędów BFS	Procent błędów DFS	Procent błędów LC
3	0,01517205	0,016123782	0,01499207	0	0	0
4	0,03304046	0,035657208	0,03818165	0	0	0
5	0,11155036	0,055778	0,12788832	0	0	0
6	0,19519207	0,274882416	0,39009418	0	0	0
7	1,830780816	0,211604416	1,371585625	0	0	0
8	0,552443566	0,77830199	4,619414719	0	0	0
9	7,2276498	0,456261178	10,74147573	0	0	0
10	42,8539055	0,745363406	37,716372	0	0	0
11	249,3856335	7,582802128	34,78156684	0	0	0
12	48,5710099	14,39626261	269,612818	0	0	0
13	13736,19574	27,892348	1213,54132	0	0	0
14	26602,62314	46,48223069	5441,5507	16	0	0
15	36549,19725	75,4280802	16331,54191	40	0	0
16	35404,79346	267,5723218	64921,855	64	0	0
17	88769,94056	567,1547627	103934,9911	70	0	10
18	162610	1848,825137	165250,6467	86	0	40
19	235165,9	3507,585255	232123	90	0	92
20	x	11421,59541	300000	x	0	100
21	x	34570,43275	x	x	0	x
22	x	78652,6496	x	x	0	x
23	x	87822,8	x	x	8	x
24	x	93585,3	x	x	24	x
25	x	96624,5125	x	x	40	x

Tabela 4 przedstawia średnie czasy wykonania algorytmu podziału i ograniczeń dla różnych metod przeszukiwania przestrzeni rozwiązań. Oprócz tego posiada ona także informacje o tym jaki procent iteracji został przerwany ze względu na przekroczenie limitu czasowego jakim było 5 minut. Wartości „x” w tabeli oznaczają brak pomiaru. Grafy, które testowane były w tej części sprawozdania są grafami asymetrycznymi



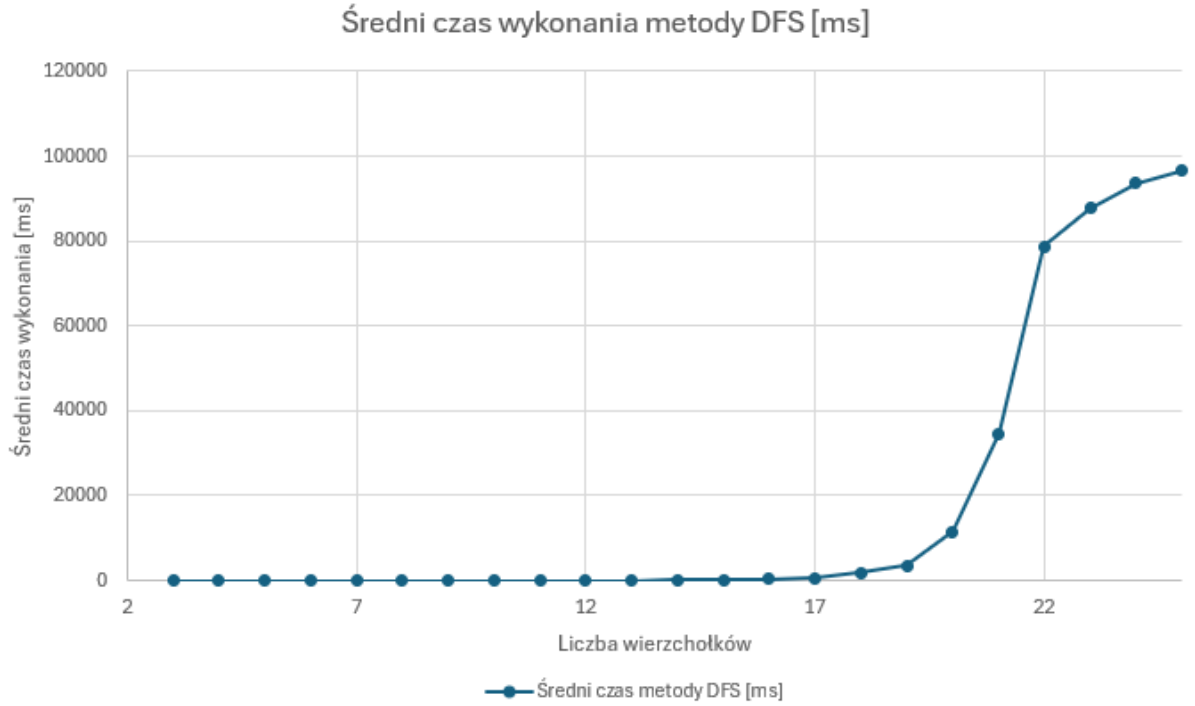
Rysunek 11. Średni czas wykonania metody BFS dla grafów asymetrycznych

Rysunek 11 przedstawia wykres funkcji średniego czasu wykonania algorytmu wszerz dla grafów asymetrycznych. Zauważalne wydłużenie średniego czasu wykonania pojawia się tutaj dla instancji o 2 wierzchołki więcej niż dla grafów symetrycznych. Końcowy średni czas wykonania dla grafów o 19 wierzchołkach wzrósł jednak o około 1 minutę, w porównaniu z wynikami dla grafów symetrycznych.



Rysunek 12. Średni czas wykonania metody LC dla grafów asymetrycznych

Rysunek 12 przedstawia wykres funkcji średniego czasu wykonania algorytmu LC dla grafów asymetrycznych. Widać tutaj znaczne polepszenie w porównaniu do grafów symetrycznych, ponieważ dla grafów o 18 wierzchołkach algorytm ten dwa razy częściej kończył się poprawnym odnalezieniem rozwiązania. Dodatkowo dla 19 wierzchołków także udało się tej metodzie znaleźć rozwiązania w czasie szybszym niż 5 minut.



Rysunek 13. Średni czas wykonania metody DFS dla grafów asymetrycznych

Rysunek 13 potwierdza, że najlepszą z metod podziału i ograniczeń badanych w tym sprawozdaniu jest metoda w głąb. W porównaniu z grafami symetrycznymi, dla grafów asymetrycznych osiąga ona znacznie lepsze wyniki, ponieważ dla grafów o 25 wierzchołkach wykonuje się ona średnio o ponad 2 minuty szybciej, co jest znaczną poprawą. Także częstotliwość zakończenia iteracji ze względu na limit czasowy zmniejszyła się zauważalnie.

Wnioski

Analiza metod rozwiązania problemu komiwojażera (TSP) przeprowadzona w sprawozdaniu pozwala na wyciągniecie istotnych wniosków dotyczących efektywności różnych podejść optymalizacyjnych i heurystycznych. Problem TSP, będący problemem NP-trudnym, stawia wysokie wymagania przed algorytmami stosowanymi do jego rozwiązania. Badania objęły zarówno metody gwarantujące rozwiązanie optymalne, jak i heurystyki oferujące szybsze, lecz nie zawsze precyzyjne wyniki.

Podstawową metodą testowaną w tym sprawozdaniu jest metoda przeglądu zupełnego. Potwierdzono jej teoretyczną złożoność obliczeniową O(n!), co sprawia, że użycie jej sprawdza się w niewielkich instancjach, mniej więcej do 10 wierzchołków w grafie. Pod względem praktyczności lepiej sprawdza się metoda najbliższego sąsiada, ta bowiem zapewniła średnie czasy rozwiązania poniżej 1ms. Generuje ona jednak rozwiązania suboptymalne, więc nie sprawdza się we wszystkich przypadkach, szczególnie gdy naszym wymaganiem jest otrzymanie rozwiązania optymalnego. Metoda losowa pozytywnie zaskoczyła faktem, że zwracała optymalne wyniki dla grafów o liczbie wierzchołków z przedziału <3,10> i <3,11> dla grafów odpowiednio symetrycznych i asymetrycznych. Zakładano, że będzie ona działać tak dobrze maksymalnie dla grafów i liczbie wierzchołków równej 6, lecz badania potwierdziły, że nadaje się ona także do minimalnie większych instancji.

Szczególnie interesującą grupą metod były techniki podziału i ograniczeń (Branch and Bound), w tym algorytmy przeszukiwania wszerz (BFS), w głąb (DFS) oraz podejścia minimalizacji kosztu (LC). Zastosowanie tych metod pozwoliło znacząco poprawić efektywność obliczeń w porównaniu do prostych heurystyk. DFS wyróżniał się na tle pozostałych dzięki najlepszej wydajności czasowej i minimalnemu zużyciu pamięci. Był on też w stanie zwrócić wynik optymalny w limicie czasu 5 minut dla grafów o 25 wierzchołkach, gdy pozostałe nie były w stanie tego zrobić ani razu dla grafów o 20 wierzchołkach.

Badania wykazały także różnice w efektywności algorytmów podziału i ograniczeń w zależności od typu grafu. Dla grafów asymetrycznych, gdzie odległość między miastami nie jest równa w obu kierunkach, metody B&B osiągały lepsze wyniki w porównaniu do grafów symetrycznych. Szczególnie zauważalne było to w przypadku metod LC i DFS, gdzie w tym drugim największa różnica wyniosła ponad 2 minut.

Nieprzewidzianym elementem okazała się zaobserwowana anomalia w wynikach metody losowej. Dla grafów symetrycznych o 7 wierzchołkach odnotowano niespodziewany wzrost czasu wykonania, którego nie udało się w pełni wyjaśnić. Ponowne testy wykazały powtarzalność tego zachowania, co może wskazywać na nietypowe cechy testowanych instancji, bądź specyficzne interakcje w algorytmie. Potwierdza to tezę o tym, że algorytmy heurystyczne mogą w niektórych przypadkach działać niestabilnie, co ogranicza ich przydatność.

Podsumowując, wybór odpowiedniej metody rozwiązania problemu komiwojażera (TSP) powinien być uzależniony od charakterystyki instancji oraz wymagań dotyczących czasu i dokładności obliczeń. W momencie gdy rozmiar instancji jest mały, a zależy nam na optymalnym wyniku warto użyć metody przeglądu zupełnego, z powodu prostoty jej

implementacji i jednocześnie szybkim czasie wykonania dla takich rozmiarów grafów. Proste heurystyki, takie jak najbliższy sąsiad, sprawdzają się dla małych i średnich grafów, gdzie szybkość działania jest kluczowa, a dokładność wyników mniej istotna. Dla większych instancji bardziej opłacalne są metody podziału i ograniczeń, zwłaszcza DFS, który oferuje korzystny czas wykonania, małe zużycie pamięci i dobrą jakość wyników. Eksperymenty przeprowadzone w sprawozdaniu stanowią wartościowy wkład w praktyczne rozważania nad efektywnym rozwiązaniem problemów NP.-trudnych w rzeczywistych zastosowaniach.