Politechnika Wrocławska

Programowanie efektownych algorytmów

Problem komiwojażera (TSP)

Data oddania: 22.11.2024

Autor: Krzysztof Zalewa 273032

Spis treści

1. Problem komiwojażera (TSP)

Problem komiwojażera (ang Travelling salesman problem) jest problemem obliczeniowym o złożoności NP. W problemie komiwojażera dany jest zbiór miast (Wierzchołków w grafie) i kosztów podróży między każdym z nich. Celem jest znalezienie najtańszej drogi zaczynającej i kończącej się w tym samym mieście (wierzchołku). Obecnie nie istnieje rozwiązanie problemu komiwojażera które wykonało by się w czasie wielomianowym. Mimo tego że optymalne rozwiązanie nie istnieje jest wiele innych dobrych rozwiązań.

2. Specyfikacja sprzętu użytego do badań

Badania zostały wykonanie na komputerze stacjonarnym o specyfikacji:

Procesor: Intel(R) Core(TM) i7-9700K CPU

Zegar: 3.60GHz

Wielkość pamięci RAM: 32,0 GB (dostępne: 31,8 GB)

3.Procedura badawcza

Dla każdego z poniżej zaprezentowanych algorytmów wykonano badania dla:

- Macierzy symetrycznych i niesymetrycznych
- Trzech gęstości macierzy 30%, 60% i 100%
- Macierzy o ilości wierzchołków od 8 do 12 włącznie
- W algorytmie losowym ograniczeń czasowych 10 000 000, 100 000 000 i 1 000 000 000, 10 000 000 000 ns (0.01 , 0.1 , 1 i 10 sekund)

Dla każde z 30 konfiguracji (2*3*5) wykonano 10 powtórzeń by uzyskać wiarygodne wyniki.

4. Badania

4.1. Algorytm losowy (ang. Random)

4.1.1. Opis algorytmu

Algorytm losowy to najprostsze możliwe rozwiązanie problemu komiwojażera. Polega on na wylosowaniu ścieżki i porównaniu jej kosztu z najlepszym znalezionym do tej pory. Jeżeli nowy koszt jest mniejszy to należy ustawić obecny koszt jako najlepszy i kontynuować.

4.1.2. Lista kroków

1. Tworzona jest pierwsza ścieżka 0,1,2,…n-1
2. Ścieżka jest mieszana
3. Zliczana jest wartość ścieżki
4. Jeżeli obliczona wartość jest mniejsza od najlepszej do tej pory otrzymanej należy je zamienić
5. Algorytm powtarza się tak długo aż nie przekroczy wyznaczonego czasu

4.1.3. Założenia badawcze

Dla małych instancji wynik będzie zbliżony (lub równy) do optymalnego. Dla większych instancji wynik prawdopodobnie nie będzie optymalny. Wyniki dla opcji z ograniczeniem do 10s będzie znacznie lepszy niż dla funkcji z mniejszymi ograniczeniami

4.1.4. Wyniki

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0,002015	0,000833	0,001752	0,002821	0,001887	0,000432
9	6,25E-05	0,004514	0,010001	0,00723	0,010001	0,010001
10	0,010001	0,010001	0,006283	0,010001	0,010001	0,010001
11	0,010002	0,010004	0,010001	0,010001	0,010001	0,010001
12	0,010001	0,010001	0,010001	0,010001	0,010001	0,010002

Tabela 1. Czasy wykonania funkcji random [s] dla maksymalnego czasu 0,01 s

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0	0	0	0	0	0
9	0	0	68	0	1554	919
10	13198	4791	0	10964	2192	347
11	N/A	10484	7078	N/A	7613	4501
12	N/A	6728	8220	15537	13322	9072

Tabela 2. Błąd bezwzględny funkcji random dla maksymalnego czasu 0,01 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	29723	22264	12806	28719	14466	18843
9	23923	28513	7197	26942	15282	18508
10	37760	44003	16436	37371	16399	15188
11	N/A	42585	22697	N/A	26730	17562
12	N/A	33634	25236	47456	27049	20744

Tabela 3. Wyniki funkcji random dla maksymalnego czasu 0,01 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0,001975	0,000863	0,001826	0,002925	0,001912	0,000444
9	6,19E-05	0,004554	0,013067	0,007259	0,013034	0,027349
10	0,100001	0,066547	0,006221	0,100001	0,051338	0,05376
11	0,100001	0,100001	0,100001	0,100001	0,100001	0,100001
12	0,100001	0,100001	0,100001	0,100001	0,100001	0,100001

Tabela 4. Czasy wykonania funkcji random [s] dla maksymalnego czasu 0,1 s

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	2335	0	0	272	0	0
11	N/A	1657	2672	7485	3715	1700
12	N/A	6728	5180	8377	6050	3808

Tabela 5. Błąd bezwzględny funkcji random dla maksymalnego czasu 0,1 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	29723	22264	12806	28719	14466	18843
9	23923	28513	7129	26942	13728	17589
10	26897	39212	16436	26998	14207	14841
11	N/A	33758	18291	42497	22832	14761
12	N/A	33634	22196	40296	19777	15480

Tabela 6. Wyniki funkcji random dla maksymalnego czasu 0,1 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0,00198	0,000837	0,001767	0,002812	0,001897	0,000441
9	6,36E-05	0,004542	0,012809	0,007233	0,013208	0,027025
10	0,180671	0,066142	0,00624	0,106923	0,051262	0,054458
11	0,690919	0,385427	1,000001	0,688965	1,000001	0,856074
12	1,000001	1,000001	1,000001	1,000001	1,000001	1,000001

Tabela 7. Czasy wykonania funkcji random [s] dla maksymalnego czasu 1 s

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	488	0	1174	0
12	2963	3590	640	4271	3621	2507

Tabela 8. Błąd bezwzględny funkcji random dla maksymalnego czasu 1 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	29723	22264	12806	28719	14466	18843
9	23923	28513	7129	26942	13728	17589
10	24562	39212	16436	26726	14207	14841
11	40077	32101	16107	35012	20291	13061
12	42675	30496	17656	36190	17348	14179

Tabela 9. Wyniki funkcji random dla maksymalnego czasu 1 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0,001976	0,000828	0,001712	0,002814	0,001889	0,000431
9	6,17E-05	0,004498	0,013041	0,007257	0,012914	0,026982
10	0,180213	0,066238	0,006337	0,106698	0,051572	0,056292
11	0,690526	0,385425	1,275675	0,689357	6,300082	0,855919
12	3,532721	10	10	10	2,052986	10

Tabela 10. Czasy wykonania funkcji random [s] dla maksymalnego czasu 10 s

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	837	284	918	0	107

Tabela 11. Błąd bezwzględny funkcji random dla maksymalnego czasu 10 s (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	30%	60%	100%	30%	60%	100%
8	29723	22264	12806	28719	14466	18843
9	23923	28513	7129	26942	13728	17589
10	24562	39212	16436	26726	14207	14841
11	40077	32101	15619	35012	19117	13061
12	39712	27743	17300	32837	13727	11779

Tabela 12. Wyniki funkcji random dla maksymalnego czasu 10 s (N/A oznacza że nie znaleziono ścieżki)

4.1.4 Wnioski

Zaimplementowana metoda random daje optymalne wyniki dla niewielkich macierzy (8- 9 wierzchołków) już w 0,01 sekundy. Wyniki dla większych macierzy poprawiają się wraz z czasem wykonania algorytmu. Dla czasów wykonania 0,01 i 0,1 dla macierzy nie pełnych o liczbie wierzchołków wyniki wychodziły nie poprawne. Prawdopodobnie dla tego że w algorytmie random macierze nie pełne zajmują więcej czasu niż pełne.

4.2. Algorytm najbliższego sąsiada (ang. Nearest neighbour)

4.2.1 Opis algorytmu

Algorytm najbliższego sąsiada jest algorytmem zachłannym. Zaczynając z wierzchołka wybiera się jeden wierzchołek do którego droga jest najkrótsza. Ten proces jest powtarzany tak długo aż przejdziemy przez wszystkie miasta. Początkowy wierzchołek wybierany jest w pętli, więc NN wykona się dla każdego z wierzchołków.

4.2.2. Lista kroków

1. Pierwszy zostanie wybrany wierzchołek 0
2. Dla wybranego wierzchołka sprawdzany jest koszt przejścia do nie odwiedzonych wierzchołków
3. Wybierany jest wierzchołek o najniższym koszcie przejścia
4. Kroki 2 -3 powtarzają się tak długo aż nie zostaną odwiedzone wszystkie wierzchołki grafu
5. Otrzymany wynik porównywany jest z najlepszym do tej pory otrzymanym
6. Jako wierzchołek startowy wybierany jest kolejny wierzchołek 1,2….n-1 jeszcze raz wywoływane są kroki 2 – 6

4.2.3. Założenia badawcze

Jako algorytm zachłanny prowadzi on do „jakiegoś" rozwiązania które nie koniecznie będzie optymalne. Dla grafów nie pełnych rozwiązanie może nie być poprawne.

4.2.4. Wyniki

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	0,000102	0,000119	0,00003131	0,00009777	0,00007901	0,0000317
9	0,000114	0,00017	0,00004568	0,00017193	0,00014056	0,00004516
10	0,000339	0,000239	0,00006631	0,00034104	0,0003382	0,00006679
11	0,00039	0,000386	0,00009162	0,00053767	0,00032088	0,00011535
12	0,000822	0,000707	0,00012198	0,0008222	0,00070706	0,00012296

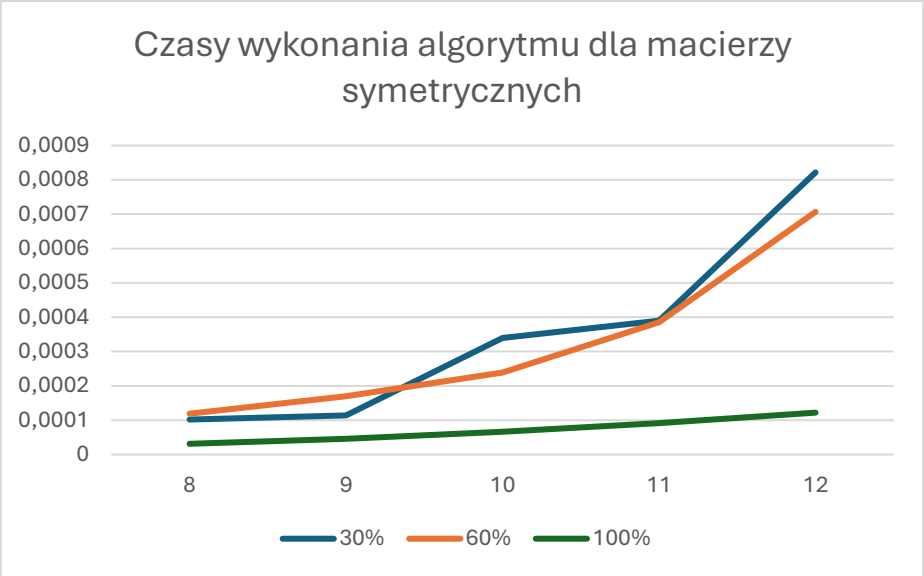
Tabela 13. Czasy wykonania funkcji nearest-neighbour [s] (Na czerwono zaznaczono instancje w których nie udało się znaleźć żadnego wyniku)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	0,00%	N/A	18,93%	33,25%	N/A	15,69%
9	N/A	19,81%	32,09%	N/A	17,25%	0,00%
10	N/A	1,26%	12,98%	N/A	N/A	29,74%
11	N/A	5,16%	25,32%	N/A	1,93%	26,28%
12	N/A	N/A	4,08%	N/A	0,01%	31,38%

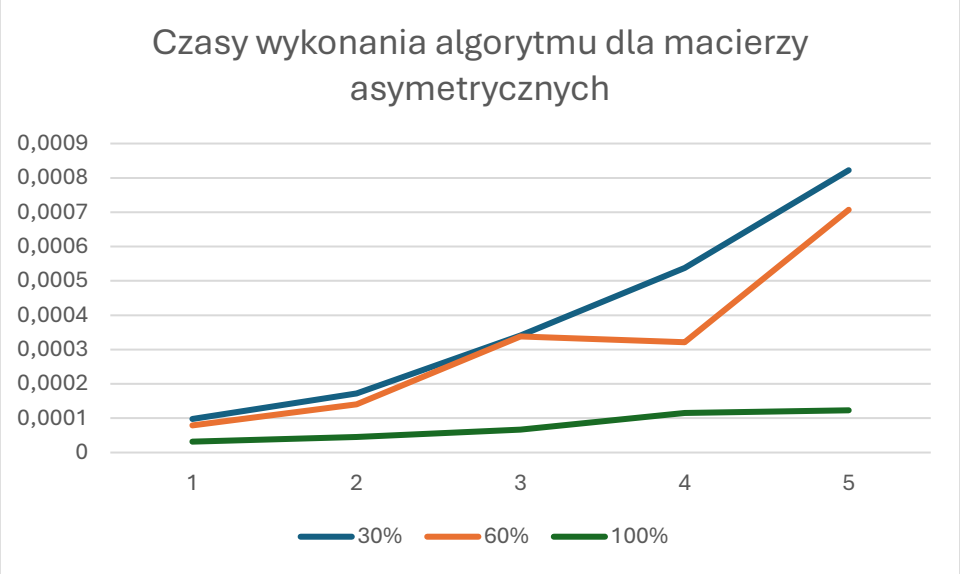
Tabela 14 . Błąd względny funkcji nearest-neighbour (N/A oznacza że nie znaleziono ścieżki)

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	30867	N/A	15230	38267	N/A	24012
9	N/A	34161	9417	N/A	17918	17589
10	N/A	39705	18569	N/A		19255
11	N/A	33758	19573	N/A	19994	17246
12	N/A	N/A	17710	N/A	15781	16216

Tabela 15. Wyniki funkcji nearest-neighbour (N/A oznacza że nie znaleziono ścieżki)



Rysunek 1. Czasy wykonania algorytmu dla macierzy symetrycznych [s]



Rysunek 2. Czasy wykonania algorytmu dla macierzy asymetrycznych [s]

4.2.4 Wnioski

W tej implementacji metody NN otrzymano niewiele optymalnych wyników (2 na 30). Dla macierzy niepełnych otrzymany wynik mógł być nie poprawny. Szansa na otrzymanie poprawnego wyniku maleje wraz z wzrostem gęstości (dla 30% tylko 1 wynik był poprawny, dla 60% to już 3 wyniki)

4.3. Algorytm siłowy (ang. Brute-force)

4.3.1 Opis algorytmu

Algorytm siłowy polega na wygenerowaniu każdej możliwej iteracji rozwiązania. Koszt każdej iteracji jest zliczany i porównywany z najlepszym do tej pory otrzymanym. Wygenerowanie każdej iteracji w mojej implementacji jest osiągane przy użyciu algorytmu heapa. Algorytm heapa generuje iteracje poprzez zmianę i tego elementu z n-1. Następnie jest on wywoływany rekurencyjnie dla zmienionej ścieżki.

4.3.2. Lista kroków

6. Tworzona jest pierwsza iteracja 0,1,2,…n-1
7. Rekurencyjnie wywoływany jest algorytm heapa dla n-1
8. Algorytm heapa zatrzymuje się gdy n=1
9. Należy zliczyć wynik otrzymanej iteracji i porównać z najlepszym do tej pory otrzymanym wynikiem. I zamienić jeżeli jest mniejszy.
10. Algorytm cofa się do poprzedniej iteracji algorytmu heapa i zamienia i-ty element z elementem n-1 (gdzie i 0,1,2,…n-1)
11. Jeszcze raz rekurencyjnie wywoływany jest algorytm heapa dla n-1
12. Algorytm heapa zakończy się gdy wytworzy wszystkie możliwe iteracje

4.3.3. Założenia badawcze

Brute force na pewno da optymalny wynik. Ale będzie się wykonywał dłużej niż pozostałe algorytmy.

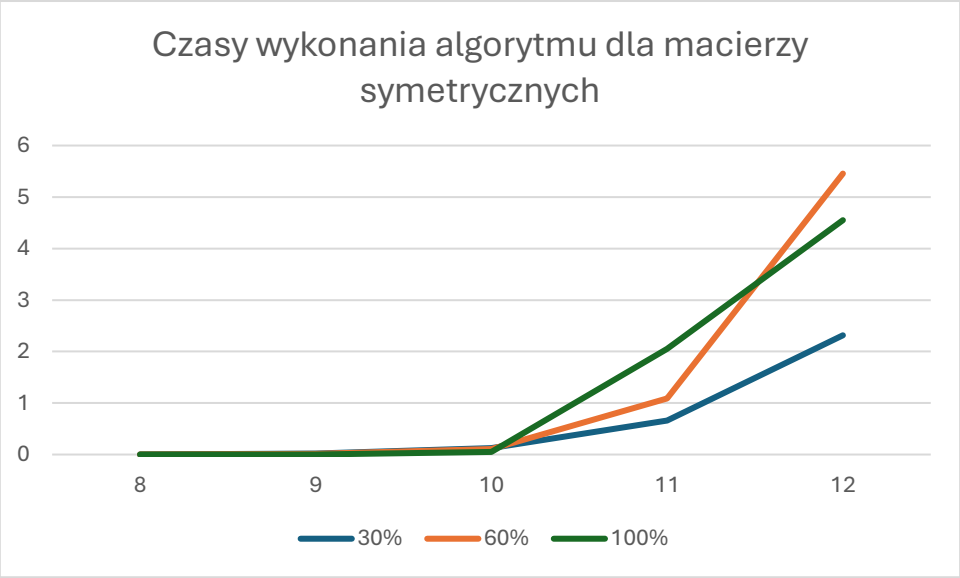
4.3.4. Wyniki

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	0,000584	0,001455	0,002964	0,000594	0,00593	0,006764
9	0,022613	0,012452	0,000671	0,007114	0,047743	0,002322
10	0,125065	0,106813	0,049782	0,38329	0,47834	0,565129
11	0,658866	1,088163	2,050599	4,200745	5,931383	0,191078
12	2,315414	5,455512	4,550668	50,45979	63,29362	25,10583

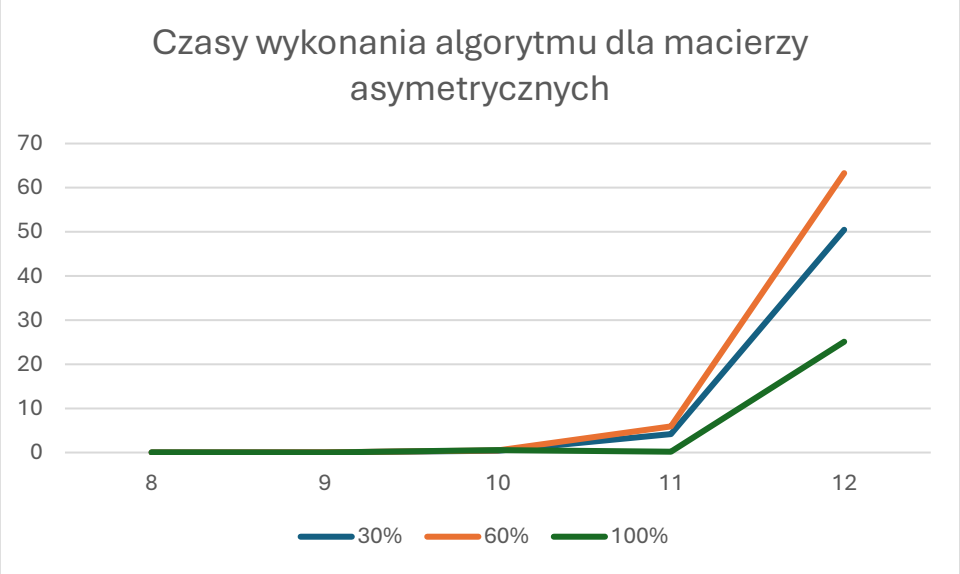
Tabela 16. Czasy wykonania funkcji brute-force [s]

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	30867	22264	12806	28719	16035	18843
9	23923	28513	7129	26942	13728	17589
10	24562	39212	16436	26726	14207	14841
11	40077	32101	15619	35236	19117	13657
12	39712	26906	17016	31919	13727	12343

Tabela 17. Wyniki funkcji brute-force



Rysunek 3. Czasy wykonania algorytmu dla macierzy symetrycznych [s]



Rysunek 4. Czasy wykonania algorytmu dla macierzy asymetrycznych [s]

4.2.4 Wnioski

Dla niewielkich instancji wyniki dla różnych gęstości był zbliżone. Przy większych macierzach wyniki się rozwarstwiają. Zakładałem że czasy wykonania dla gęstości 100% będą największe, jednakże na wykresach widać że to gęstość 60% wykonywała się naj dłużej. Prawdopodobnie jest to kwestia danych użytych do badania.

4.4. Metoda podziału i ograniczeń (ang. Branch and bound)

Metoda branch and bound polega na systematycznym przeszukiwaniu przestrzeni rozwiązań (Punkty 4.5 – 4.7). Najważniejszym elementem tej metody jest ograniczanie liczby przypadków do sprawdzenia poprzez zastosowanie podziału (branch) i ograniczeń (bound) .

4.5. Przeszukiwanie w głąb (ang. Depth first search)

4.5.1 Opis algorytmu

Przeszukiwanie w głąb polega na przejściu przez wszystkie wierzchołki w odpowiedniej kolejności. Zaczynając od wierzchołka 0 wybieramy jedno z jego dzieci. Następnie dla wybranego dziecka wybieramy jego dziecko (wnuk wierzchołka 0). Proces ten powtarzamy tak długo aż nie trafimy na liścia. Jeżeli wybrany wierzchołek jest liściem należy zliczyć koszt wytworzonej trasy i porównać z najlepszym otrzymanym do tej pory. Jeżeli obecny koszt jest lepszy należy go zamienić. Następnie cofamy się do poprzednio odwiedzonego wierzchołka i wybieramy inne dziecko niż to z którego przszliśmy. DFS zwykle wykonywany jest przy pomocy stosu.

4.5.2. Lista kroków

1. Na stos wrzucany jest wierzchołek 0
2. Wyliczamy ograniczenie dolne oraz ograniczenie górne (za pomocą nearest neighbour)
3. Tak długo jak stos nie jest pusty
4. Zdejmowany jest wierzchołek na szczycie
5. Jeżeli jego granica dolna jest większa lub równa ograniczeniu górnemu pomijamy ten wierzchołek
6. Jeżeli ścieżka przypisana do wierzchołka jest równa n to zliczana jest jego wartość.
13. Wartość porównywana jest z najlepszym znalezionym rozwiązaniem. I zamieniana jeżeli jest mniejsza.
7. Jeżeli ścieżka jest krótsza od n należy znaleźć nowy wierzchołek i który nie był odwiedzony (i = 0,1,2…n-1)
8. Dla tego wierzchołka należy wyliczyć ścieżka która do niego prowadzi i jego granicę
9. Jeżeli granica jest mniejsza od najlepszego znalezionego rozwiązania wierzchołek dodawany jest na stos.

4.5.3. Założenia badawcze

Depth first search będzie szybszy od brute forca da dobry wynik który na pewno będzie poprawny ale może nie być optymalny.

4.5.4. Wyniki

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	0,000186	0,000265	0,000635	0,000166	0,000288	0,000634
9	0,000363	0,000598	0,00083	0,000751	0,00088	0,000655
10	0,000106	0,002801	0,004232	0,000523	0,000546	0,00486
11	0,000469	0,007211	0,020066	0,012145	0,003791	0,024452
12	0,000246	0,007022	0,02514	0,034566	0,015293	0,020256

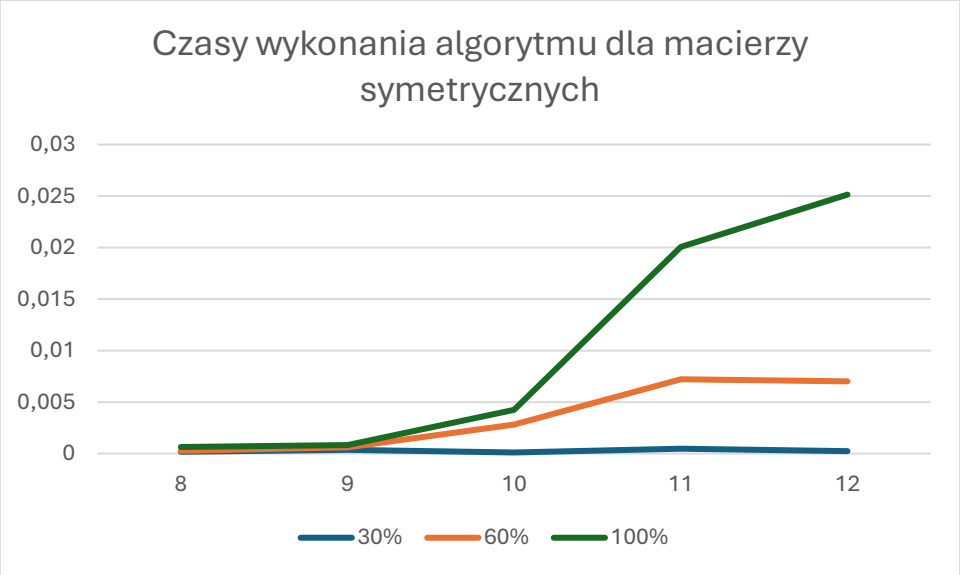
Tabela 18. Czasy wykonania funkcji depth first search [s]

Rozmiar Instancji /Gęstość	Symetryczne			Asymetryczne		
	30%	60%	100%	30%	60%	100%
8	3,71%	0,00%	0,00%	0,00%	11,47%	0,00%
9	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
10	0,00%	0,00%	0,00%	8,46%	0,00%	0,00%
11	0,00%	0,00%	0,00%	10,61%	0,00%	0,00%
12	0,00%	0,00%	0,00%	0,00%	0,00%	4,57%

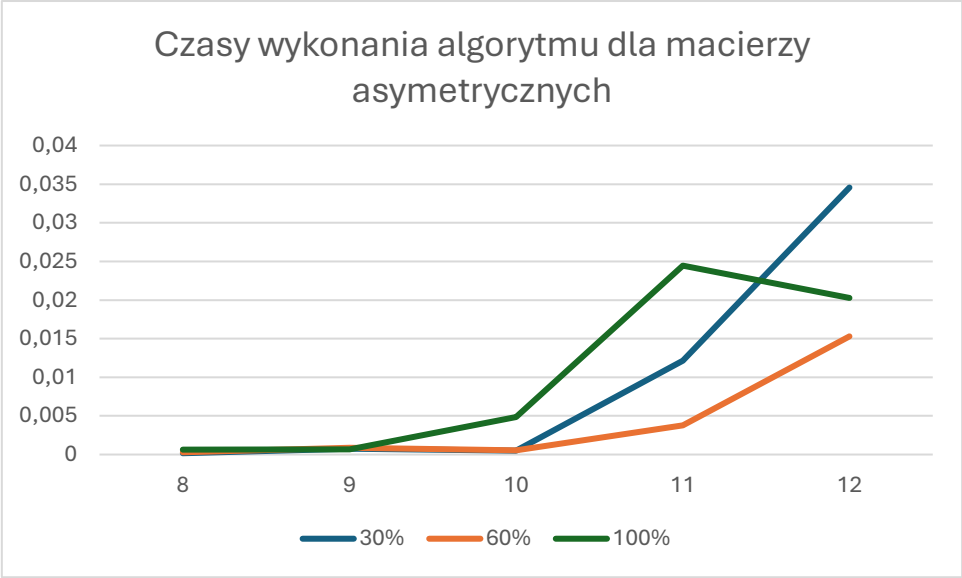
Tabela 19. Błąd względny funkcji depth first search

Rozmiar Instancji /Gęstość	Symetryczne			Asymetryczne		
	30%	60%	100%	30%	60%	100%
8	29723	22264	12806	28719	14466	20755
9	23923	28513	7129	26942	15282	17589
10	24562	39212	16436	26830	16789	14841
11	40077	32101	15619	35395	20388	13657
12	39712	26906	17016	32837	15782	11779

Tabela 20. Wyniki funkcji depth first search



Rysunek 5. Czasy wykonania algorytmu dla macierzy symetrycznych [s]



Rysunek 6. Czasy wykonania algorytmu dla macierzy asymetrycznych [s]

4.5.4 Wnioski

Druga pod względem szybkości wykonania metoda. Nie wszystkie otrzymane wyniki były optymalne. Problemem może być obliczanie dolnej granicy. Podobnie jak w BF dla macierzy asymetrycznych widać spadek czasu wykonania dla macierzy o 12 wierzchołkach.

4.6. Przeszukiwanie w szerz (ang. Breadth first search)

4.6.1 Opis algorytmu

Przeszukiwanie w szerz jest podobne do przeszukiwania w głąb. Zaczynamy w wierzchołku 0 i po kolei wybieramy wszystkie jego dzieci. Następnie dla tych dzieci wybieramy wszystkie ich dzieci. Tą procedurę powtarzamy aż do momentu gdy dotrzemy do liści. Następnie zliczamy koszt wytworzonej trasy i porównać z najlepszym otrzymanym do tej pory. Jeżeli obecny koszt jest lepszy należy go zamienić. BFS wykonywany jest przy pomocy kolejki.

4.6.2. Lista kroków

1. Na kolejkę wrzucany jest wierzchołek 0
2. Wyliczamy ograniczenie dolne oraz ograniczenie górne (za pomocą nearest neighbour)
3. Tak długo jak kolejka nie jest pusta
4. Zdejmowany jest wierzchołek na szczycie
5. Jeżeli jego granica dolna jest większa lub równa ograniczeniu górnemu pomijamy ten wierzchołek
6. Jeżeli ścieżka przypisana do wierzchołka jest równa n to zliczana jest jego wartość.
7. Wartość porównywana jest z najlepszym znalezionym rozwiązaniem. I zamieniana jeżeli jest mniejsza.
8. Jeżeli ścieżka jest krótsza od n należy znaleźć nowy wierzchołek i który nie był odwiedzony (i = 0,1,2…n-1)
9. Dla tego wierzchołka należy wyliczyć ścieżka która do niego prowadzi i jego granicę
10. Jeżeli granica jest mniejsza od najlepszego znalezionego rozwiązania wierzchołek dodawany jest na kolejka.

4.6.3. Założenia badawcze

Breadth first search będzie wykonywał się dłużej od brute forca. Wynik będzie poprawny i na pewno optymalny.

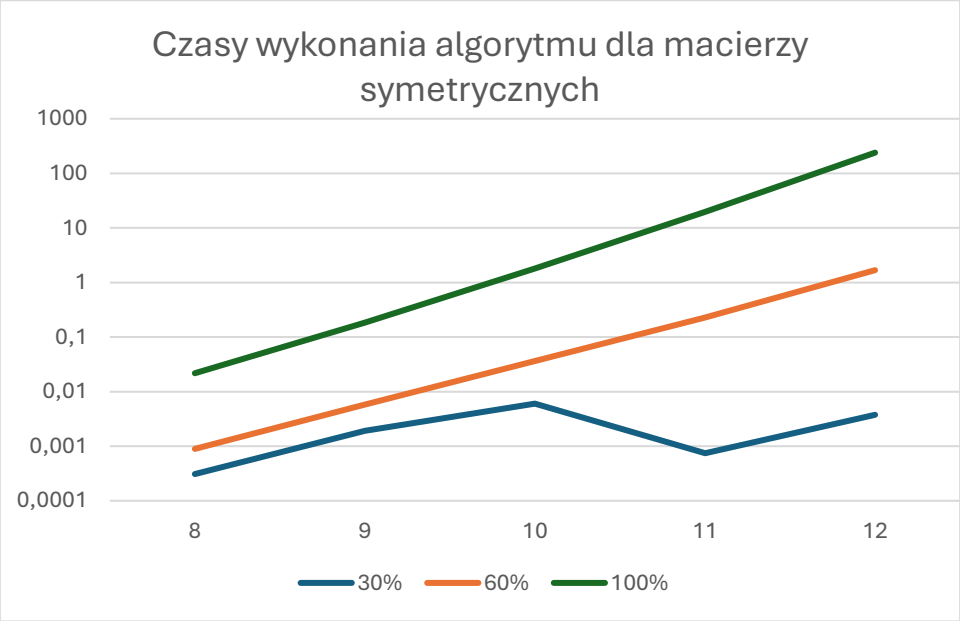
4.6.4. Wyniki

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	0,000309	0,000893	0,021742	0,000316	0,011754	0,027044
9	0,001921	0,00582	0,183279	0,002084	0,065804	0,216008
10	0,006027	0,03631	1,813179	0,006508	0,469833	1,932505
11	0,000742	0,2272	19,78348	0,042772	7,037906	20,38759
12	0,00375	1,682285	239,4432	0,295299	25,02463	239,5641

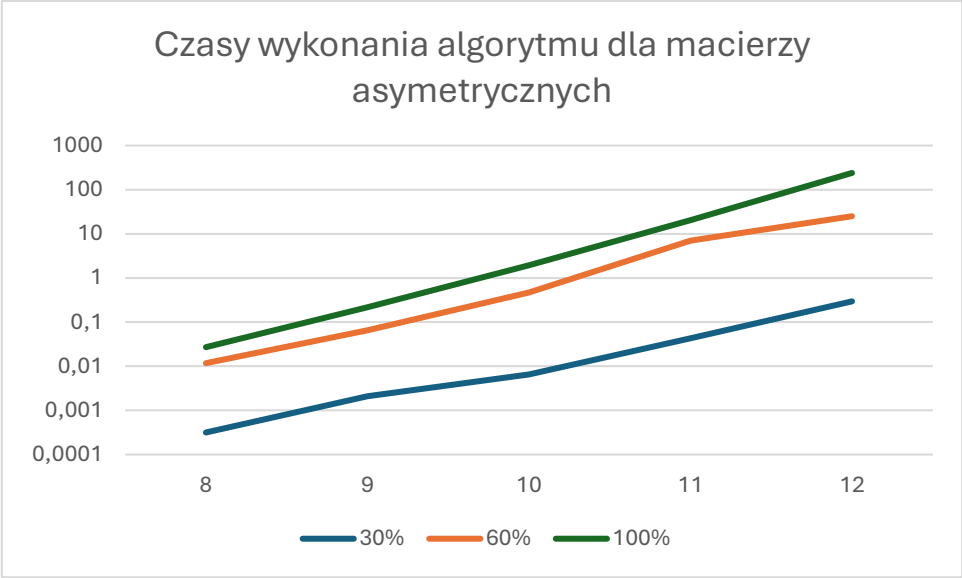
Tabela 21. Czasy wykonania funkcji breadth first search [s]

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	30867	22264	12806	28719	16340	18843
9	23923	28513	7129	26942	13728	17589
10	24562	39212	16436	26726	14207	14841
11	40077	32101	15619	39594	19117	13061
12	39712	26906	17016	32837	13727	12343

Tabela 22. Wyniki funkcji breadth first search



Rysunek 7. Czasy wykonania algorytmu dla macierzy symetrycznych [s]



Rysunek 8. Czasy wykonania algorytmu dla macierzy asymetrycznych [s]

4.6.4 Wnioski

By wyniki były lepiej widoczne na wykresach użyto skali logarytmicznej. Czasy wykonania były większe nawet od BF (Dla gęstości 100% ~ 240s gdzie w BF ten sam graf rozwiązano w ~5s). Otrzymane wyniki były optymalne.

4.7 Przeszukiwanie przy minimum kosztów (ang. Least cost)

4.7.1 Opis algorytmu

Przeszukiwanie przy minimum kosztów zaczyna się podobnie do DFS i BFS. Zaczynając w wierzchołku 0 dla każdego dziecka wyliczamy granicę według pewnej funkcji. Wybieramy dziecko o najniższym koszcie. Proces ten powtarzany jest aż do momentu gdy dotrzemy do liści. LC wykonywany jest przy pomocy kolejki priorytetowej.

4.7.2. Lista kroków

1. Na kolejkę priorytetową wrzucany jest wierzchołek 0
2. Wyliczamy ograniczenie dolne oraz ograniczenie górne (za pomocą nearest neighbour)
3. Tak długo jak kolejka priorytetowa nie jest pusta
4. Zdejmowany jest wierzchołek na szczycie
5. Jeżeli jego granica dolna jest większa lub równa ograniczeniu górnemu pomijamy ten wierzchołek
6. Jeżeli ścieżka przypisana do wierzchołka jest równa n to zliczana jest jego wartość.
7. Wartość porównywana jest z najlepszym znalezionym rozwiązaniem. I zamieniana jeżeli jest mniejsza.
8. Jeżeli ścieżka jest krótsza od n należy znaleźć nowy wierzchołek i który nie był odwiedzony (i = 0,1,2…n-1)
9. Dla tego wierzchołka należy wyliczyć ścieżka która do niego prowadzi i jego granicę
10. Jeżeli granica jest mniejsza od najlepszego znalezionego rozwiązania wierzchołek dodawany jest na kolejkę priorytetową.

4.7.3. Założenia badawcze

Least cost powinien być najszybszym zaimplementowany algorytmem i powinien zwracać poprawne i optymalne wyniki.

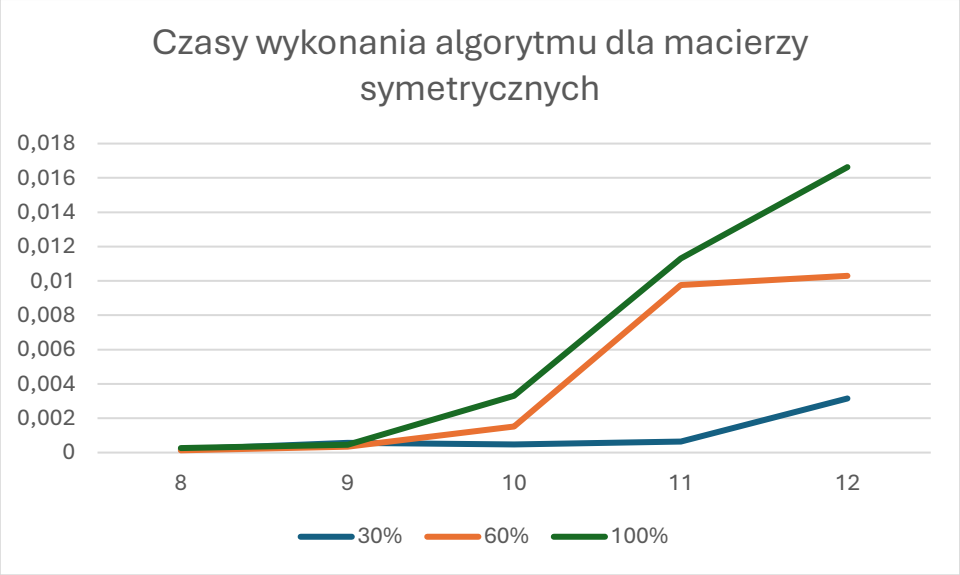
4.7.4. Wyniki

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	0,000182	0,00013	0,000262	0,000268	0,00008125	0,000387
9	0,000557	0,000336	0,000447	0,000148	0,0008032	0,001481
10	0,000469	0,001513	0,003302	0,001321	0,00155468	0,002506
11	0,000641	0,009767	0,011304	0,009657	0,00279568	0,008128
12	0,003151	0,010295	0,016628	0,009071	0,00313619	0,002091

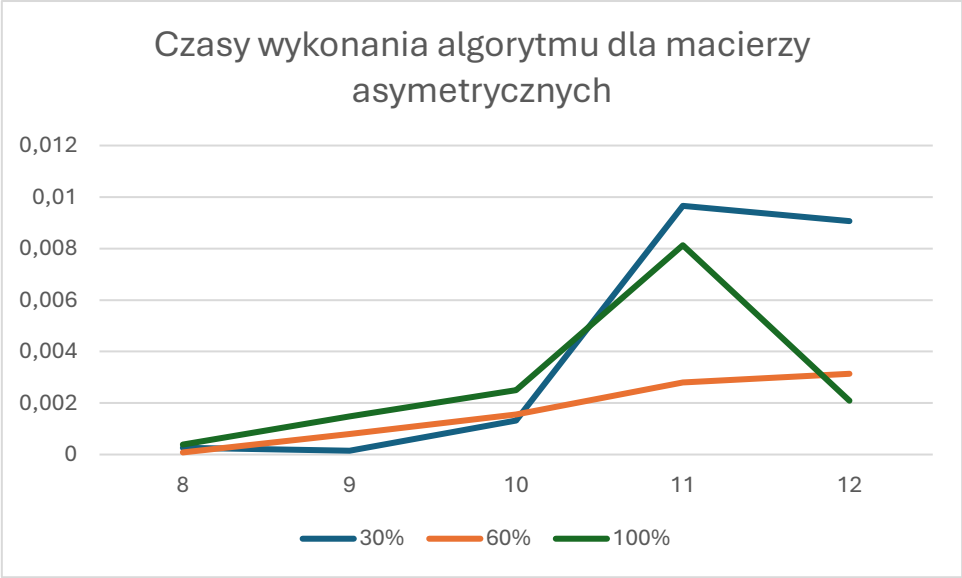
Tabela 23. Czasy wykonania funkcji least cost [s]

	Symetryczne			Asymetryczne		
Rozmiar Instancji /Gęstość	**30%**	**60%**	**100%**	**30%**	**60%**	**100%**
8	30867	22264	12806	28719	16340	20755
9	23923	28513	7129	26942	15282	17589
10	24562	39212	16436	29310	16789	14841
11	40077	32101	15619	39594	20388	13657
12	39712	26906	17016	32837	15782	12343

Tabela 24. Wyniki funkcji least cost



Rysunek 9. Czasy wykonania algorytmu dla macierzy symetrycznych [s]



Rysunek 10. Czasy wykonania algorytmu dla macierzy asymetrycznych [s]

4.7.4 Wnioski

Jest to najszybszy z zaimplementowanych algorytmów. Wyniki były optymalne. Na Rys 10 widać spadek czasu wykonania który widoczny był też w metodzie BF. Możliwe że macierz w z 12 wierzchołkami ułożona jest w taki sposób że algorytmy szybko trafiają na wynik optymalny.

4. Źródła

[1]. https://www.ceas3.uc.edu/ret/archive/2016/ret/docs/abstract/reading33.pdf
[2]. http://seor.vse.gmu.edu/~khoffman/TSP_Hoffman_Padberg_Rinaldi.pdf
[3]. https://eduinf.waw.pl/inf/alg/001_search/0109.php
[4]. https://eduinf.waw.pl/inf/alg/001_search/0110.php