ROZDZIAŁ 4

Wyniki

4.1. Zbiór danych

W przeprowadzonych analizach została użyta ogólno dostępna biblioteka TSPlib, zawierająca wiele instancji problemu komiwojażera z różną wielkością miast, różnymi typami problemu komiwojażera oraz różnym rozkładem miast. W skład biblioteki wchodzą różne scenariusze problemu komiwojażera: zarówno symetryczne, gdzie odległości między miastami są identyczne w obie strony, jak i asymetryczne, gdzie odległości mogą się różnić. TSPlib oferuje również instancje wielokryterialne, które biorą pod uwagę różne koszty, czas podróży i inne zmienne decyzyjne. Liczba miast w bibliotece waha się od 14 w najmniejszej instancji, aż do 85 900 w największej, rekordowej instancji.

Każdy przykład w TSPlib jest szczegółowo opisany i ma jednolity format. Zawiera nazwę, opis problemu, specyfikacje takie jak macierz odległości czy lista współrzędnych geograficznych, a także wiele instancji zawiera oczekiwane optymalne rozwiązanie, jeśli jest znane.

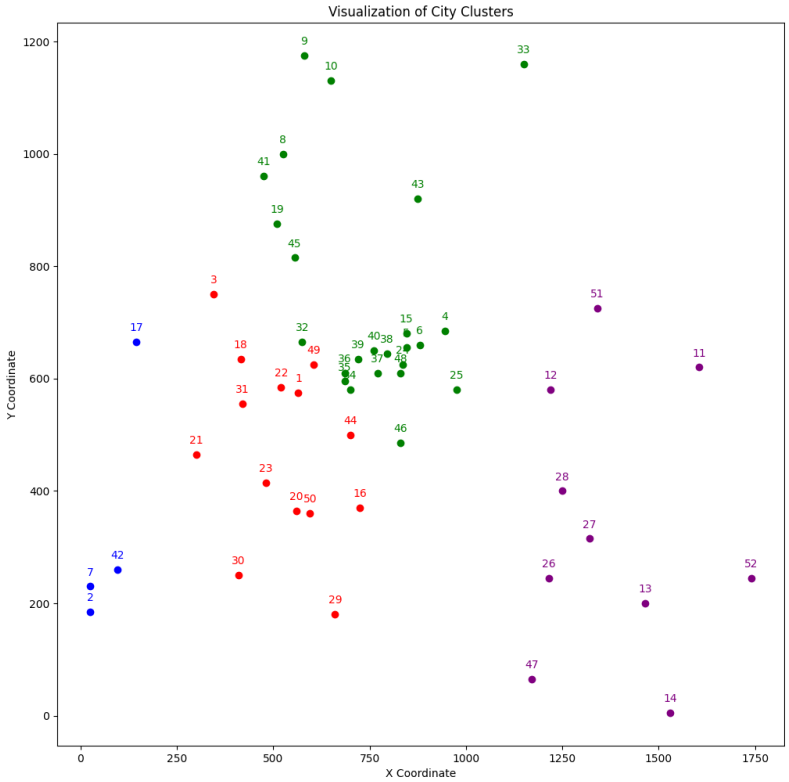
Do przetestowania algorytmu najbliższego sąsiada, mrówkowego, genetycznego i symulowanego wyżarzania, oraz różnych wersji - bez klasteryzacji, oraz z 3 metodami klasteryzacji, w pracy zostały użyte instancje Berlin52, eil76, bier127 i dsj1000.

Dla każdej z testowanych instancji, dokumentowano czas wykonania algorytmu, długość znalezionej trasy oraz porównywano te wyniki z optymalnymi rozwiązaniami podanymi w TSPlib, o ile te były znane.

4.2. Klasteryzacja

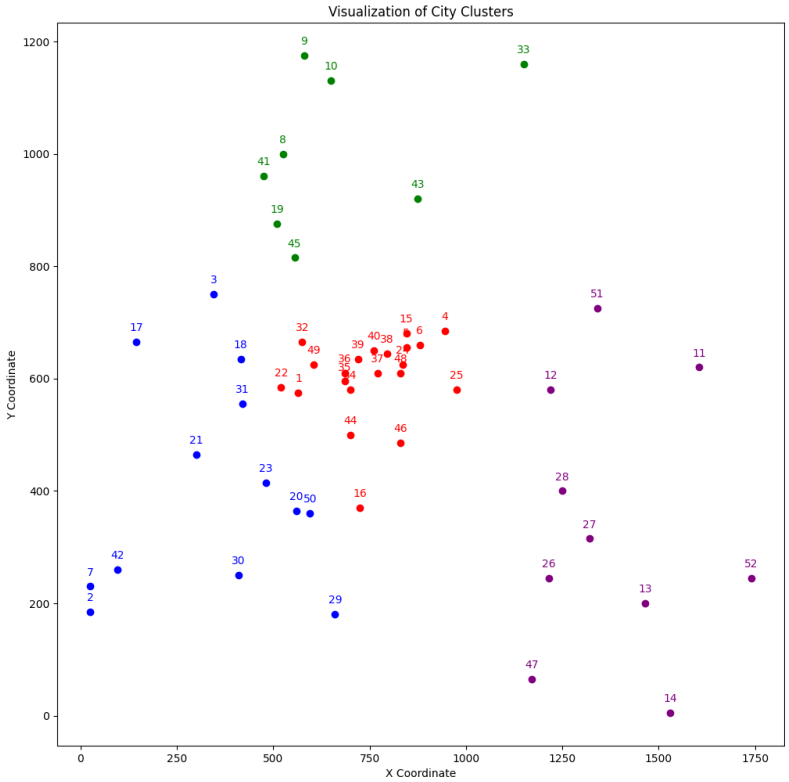
W pracy zostały użyte 3 różne sposoby grupowania danych: k-means, hierarchiczną klasteryzację algomeracyjną oraz Density-Based Spatial Clustering of Applications with Noise. Każda metoda klasteryzacji różniła się od siebie kształtem klastrów, oraz w przypadku DBSCAN ilością tych klastrów różną od k-means i hierarchicznej klasteryzacji algomeracyjnej, w których to liczba klastrów została wybierana odgórnie.

Na rysunku 4.1 został przedstawiony pierwszy zbiór danych wykorzystany w pracy o nazwie Berlin52, który zawierał 52 miasta. Widoczny jest na nim podział na 4 odgórnie wybrane klastry.



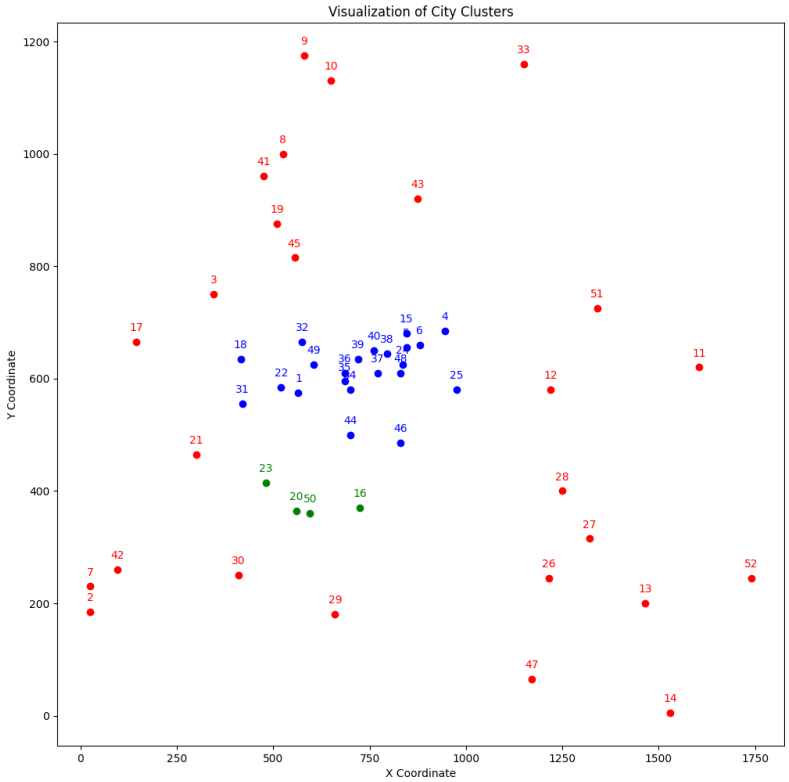
Rysunek 4.1: Klasteryzacja problemu Berlin52 za pomocą metody k-means

Rysunek 4.2 zawiera również 4 klastry stworzone za pomocą algorytmu hierarchicznej klasteryzacji algomeracyjnej dla instancji Berlin52.



Rysunek 4.2: Klasteryzacja problemu Berlin52 za pomocą metody hierarchicznej klasteryzacji algomeracyjnej

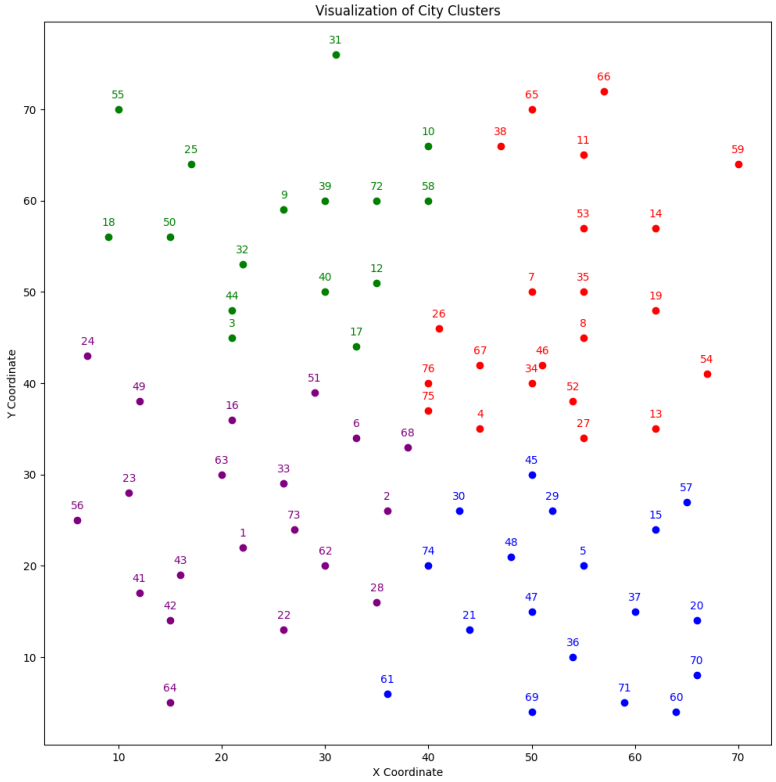
Rysunek 4.3 przedstawia metodę Density-Based Spatial Clustering of Applications with Noise, dzięki której uzyskano 3 klastry dla problemu berlin52



Rysunek 4.3: Rysunek przedstawiający klasteryzację problemu Berlin52 za pomocą metody Density-Based Spatial Clustering of Applications with Noise

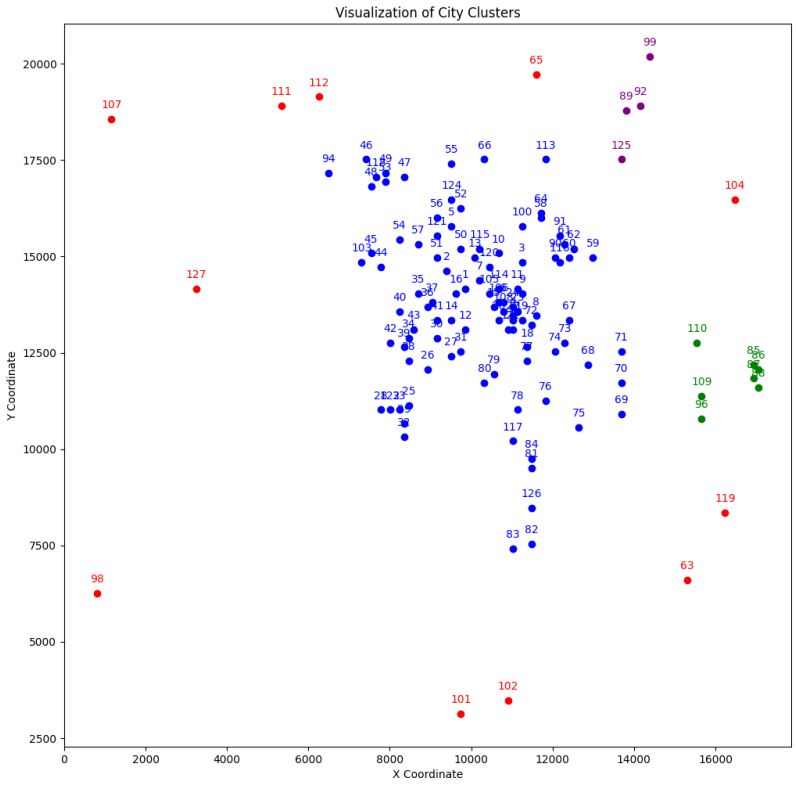
W pracy zostały również wykorzystane instancje o nazwie eil76 o wielkości 76 miast, bier127 o wielkości 127 miast, oraz dsj1000, o wielkości 1000 miast, w którym została wykorzystana tylko metoda k-means. W przypadku instancji dsj1000 nie zostały również osiągnięte rozwiązania przed klasteryzacją z powodu dużej złożoności obliczeniowej, przez którą nie było możliwe rozwiązanie problemu. Wyjątek w nim stanowi algorytm najbliższego sąsiada, który mimo wielkości problemu, osiągnął wynik w bardzo szybkim czasie. Również w przypadku instancji eil76 nie została stworzona wersja po klasteryzacji za pomocą metody DBSCAN z powodu kształtu danych, który niezależnie od zmiennych zawsze dawał jeden klaster.

Na rysunku 4.4 został ukazany rozkład miast problemu eil76 i klasteryzacja tej instancji za pomocą algorytmu k-means.



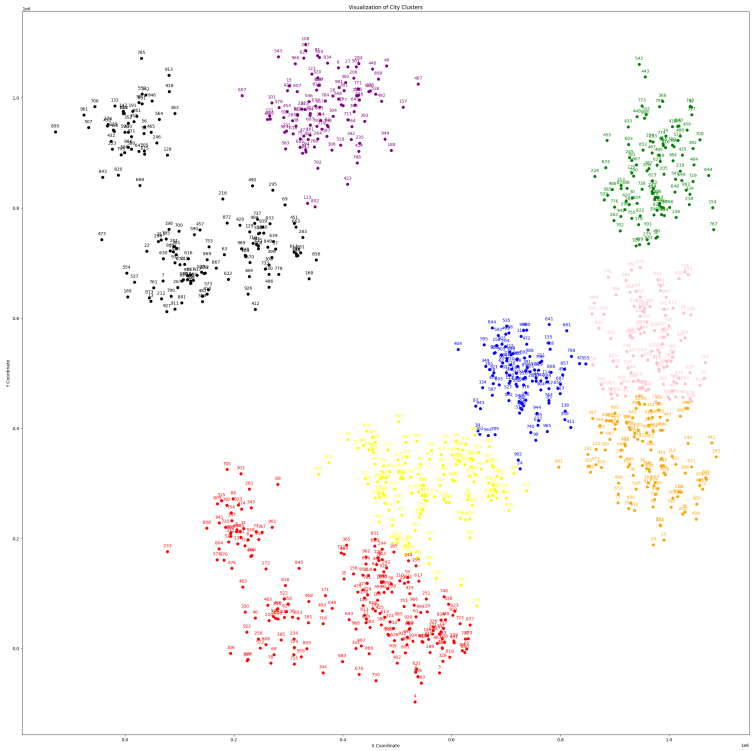
Rysunek 4.4: Klasteryzacja problemu eil76 przy pomocy k-means

Rysunek 4.5 przedstawia problem bier127 podzielony na 4 klastry przy użyciu algorytmu DBSCAN.



Rysunek 4.5: Klasteryzacja problemu bier127 metodą DBSCAN

Największa instancja użyta w pracy o nazwie dsj1000 została podzielona na 8 klastrów o różnej ilości miast przy użyciu metody k-means. Na rysunku 4.6 został ukazany wygląd poszczególnych klastrów w tym problemie.



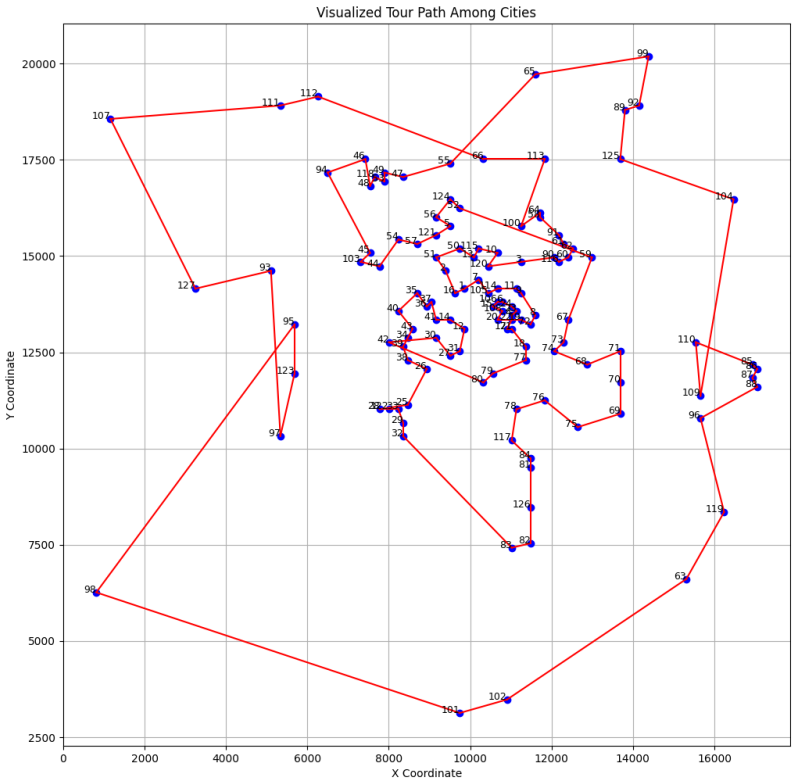
Rysunek 4.6: Klasteryzacja problemu dsj1000 metodą k-means

W przypadku różnych instancji problemu komiwojażera, przy pomocy metod klasteryzacji k-means i HKA została wybierana taka sama liczba klsatrów, która w jak najlepszy sposób grupowała dane. W przypadku metody DBSCAN były manipulowane parametry dla każdej instancji tak, żeby miasta rozkładały się jak najsensowniej na poszczególne grupy.

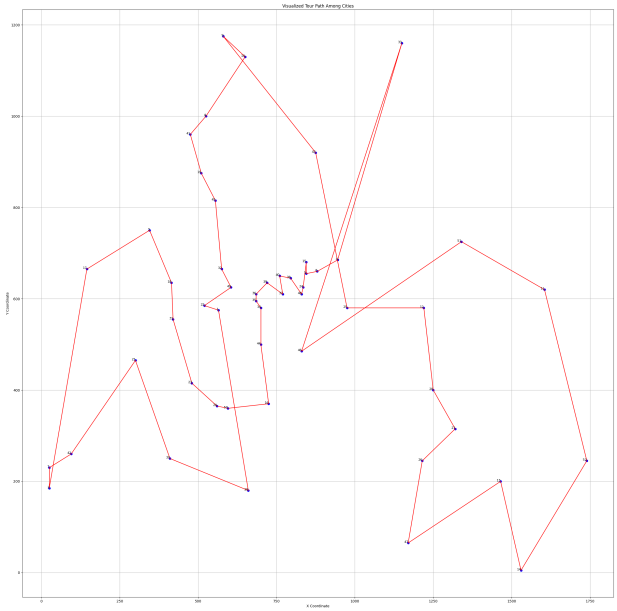
4.3. Rozwiązwanie problemu komiwojażera

Aby osiągnąć jak najlepszy wynik dla każdej wersji poszczególnych instancji problemu komiwojażera, tych przed klasteryzacją i po różnych metodach klasteryzacji, były modyfikowany parametry każdego algorytmu tak, aby osiągnąć możliwie najlepszy wynik. W przypadku algorytmu najbliższego sąsiada nie były wymagane żadne ingerencje w parametry, gdyż działanie tego algorytmu niezależnie od problemu jest takie same.

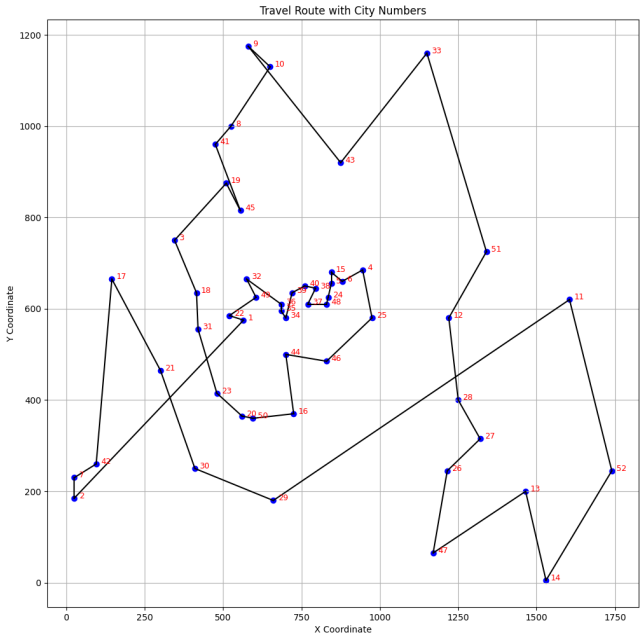
W przypadku algorytmu mrówkowego były manipulowane takie parametry jak alpha, beta, rho, Q i elite factor. Liczba mrówek była taka sama co ilość miast. Po znalezieniu najlepszych parametrów dla danej instancji, parametry były powielane dla każdej wersji w poszczególnym problemie, aby otrzymać jak najbardziej zbliżone wyniki, które miały ukazać różnice w wynikach pomiędzy różnymi metodami klasteryzacji. W przypadku wersji bez klasteryzacji parametry były inne, osiągane również manipulacją parametrów tak, aby osiągnąć jak najlepszy wynik. Rysunek 4.7 przedstawia rozwiązany problem komiwojażera dla instancji bier127 za pomocą algorytmu mrówkowego po klasteryzacji za metodą DBSCAN. Trasa osiągnęła długość 139 128,60 jednostek, a algorytm działał 11,59 sekund. Ta sama instancja bier127 rozwiązana przed klasteryzacją danych osiągnęła wynik 126 700,79 jednostek w 17,33 sekundy. Osiągnięta trasa po klasteryzacji jest dłuższa o około 9,81% w czasie krótszym o około 33,12%



Rysunek 4.7: Rozwiązanie problemu komiwojażera dla problemu bier127 w wersji z klasteryzacją DBSCAN za pomocą algorytmu mrówkowego.



(a) Rozwiązanie problemu Berlin52 z k-means (9123,06 jednostek)



(b) Rozwiązanie problemu Berlin52 bez klasteryzacji (8980,92 jednostek)

Rysunek 4.8: Porównanie rozwiązań za pomocą algorytmu najbliższego sąsiada

Tabela 4.1: Porównanie wyników dla różnych metod

Metoda	**Berlin52**		**eil76**		**bier127**		**dsj1000**	
	Dystans	**Czas**	**Dystans**	**Czas**	**Dystans**	**Czas**	**Dystans**	**Czas**
Optymalna	7544.36	–	–	–	118282.00	–	–	–
NN	8980.92	0.001	711.99	0.002	135751.78	0.006	24630960	0.39
ACO	7703.97	0.68	587.91	1.15	126700.79	17.33	–	–
GA	8998.25	14.97	689.64	17.71	190722.02	27.96	–	–
SA	8391.01	0.98	680.25	1.92	130216.98	3.61	–	–
NN k-means	9123.06	0.002	726.19	0.007	167961.20	0.021	24718246	1.47
ACO k-means	8994.33	1.35	635.57	2.23	131925.77	15.94	21913721	82.11
GA k-means	9189.10	1.69	608.80	3.40	137613.25	7.66	24297561	59.25
SA k-means	9825.73	1.36	644.17	2.23	136554.87	2.39	22975253	117.16
NN HKA	9012.15	0.003	700.87	0.007	153840.28	0.018	–	–
ACO HKA	8633.76	1.03	619.82	2.71	132875.56	3.23	–	–
GA HKA	9017.44	1.67	617.57	4.34	146857.15	7.78	–	–
SA HKA	9255.47	1.31	618.44	2.29	136955.98	2.58	–	–
NN DBSCAN	10188.74	0.0029	–	–	142677.86	0.01	–	–
ACO DBSCAN	8575.21	2.07	–	–	139128.60	11.59	–	–
GA DBSCAN	8730.44	1.85	–	–	166484.37	15.93	–	–
SA DBSCAN	9070.68	1.22	–	–	146532.98	3.52	–	–

Tabela 4.2: Wpływ różnych metod klasteryzacji na wyniki

	Berlin52		**eil76**		**bier127**	
Metoda	**Długość**	**Czas**	**Długość**	**Czas**	**Długość**	**Czas**
Bez klasteryzacji	1.00	1.00	1.00	1.00	1.00	1.00
k-means	1.09	0.26	1.02	0.38	1.02	0.53
HKA	1.05	0.24	1.04	0.45	1.02	0.28
DBSCAN	1.07	0.31	–	–	0.98	0.63

4.4. Analiza wyników

W pracy udało się sprawdzić wpływ różnych metod klasteryzacji na wyniki pracy algorytmów w rozwiązywaniu problemu komiwojażera i czas w jakim te algorytmy osiągały swój wynik. Klasteryzacja danych pogarszała wyniki pracy algorytmów w zakresie od 2% do 9%, a w jednym przypadku dzięki użyciu metod grupowania miast poprawiły się wyniki o 2%. Kosztem nieznacznie dłuższej trasy, wynik był osiągany w znacząco krótszym czasie, który poprawił się średnio o 61%, a w jednym przypadku o 76% na problemach o wielkości od 52 do 127 miast. Klasteryzacja danych na późniejszym etapie okazała się konieczną techniką w poszukiwaniu jak najbardziej optymalnej trasy. W przypadku dużej instancji, takiej jak dsj1000, w której to osiągniecie wyniku pracy algorytmów w wersji bez klasteryzacji było nieosiągalne, z wyjątkiem algorytmu Nearest Neighbour. Użycie tego podejścia skutecznie redukuje złożoność obliczeniową w przypadku problemu komiwojażera dzieląc je na większą ilość mniejszych problemów.

Algorytm NN osiągnął średnio wyniki o 7% gorsze w porównaniu z działaniem algorytmów ACO, GA i SA, ale był to zdecydowanie najszybszy i najprostszy w działaniu algorytm nie wymagający żadnej ingerencji w parametry. Algorytm mrówkowy z kolei osiągał najlepsze wyniki w rozwiązywaniu problemu komiwojażera, średnio o 9% lepsze od pozostałych algorytmów, i był to również łatwy algorytm w dobieraniu parametrów.

Tabela 4.3: Średni wynik poszczególnych algorytmów

Algorytm	**berlin52**	**eil76**	**bier127**
NN	9326,217	713,016	150057,78
ACO	8476,82	614,43	132657,68
GA	8983,81	638,67	160419,20
SA	9135,72	647,62	137565,20

Największe problemy przysporzył algorytm genetyczny i algorytm symulowanego wyżarzania, które były znacznie trudniejsze w skonfigurowaniu tak, aby dawały równie dobre wyniki. Radziły sobie lepiej od NN w przypadku instancji eil76, która charakteryzowała się bardzo zbliżoną odległością pomiędzy każdym miastem, ale już w przypadku bardziej zróżnicowanej bazy danych jak berlin52, albo dużej jak bier127, dawały wyniki zbliżonego do najprostszego algorytmu, a nawet jak w przypadku algorytmu genetycznego w bier127, gorsze.

4.5. Wnioski

Najlepszym algorytmem w pracy okazał się algorytm mrówkowy, który dla 3 zbiorów danych: Berlin52, eil76 i bier127 i łącznie 4 wersji algorytmu: bez klasteryzacji i 3 metodami klasteryzacji, osiągnął średnio najlepsze wyniki w rozwiązywaniu problemu komiwojażera. Był to również algorytm łatwy w implementacji i konfigurowaniu parametrów algorytmu. Algorytm genetyczny z kolei dobrze sprawdził się w stosunkowo małych instancjach, takich jak berlin52 i eil76, ale poradził sobie najgorzej spośród algorytmów dla instancji bier127. Mogło to wynikać z konieczności dostosowania tego algorytmu do większych instancji. Algorytm symulowanego wyżarzania sprawdził się stosunkowo dobrze w każdej instancji osiągając podobne wyniki w każdej instancji. Algorytm najbliższego sąsiada, który zgodnie z przewidywaniami wypadł najgorzej. Był to najprostszy w działaniu algorytm, który rozwiązując problem komiwojażera zawsze wybierał najbliższe następne miasto, aż do ukończenia trasy. Mimo gorszych wyników w każdej instancji, jest to niezwykle szybki algorytm nie wymagający zastosowania metod klasteryzacji w celu przyspieszenia działania tego algorytmu.

Klasteryzacja zgodnie z oczekiwaniami znacznie przyspieszyła rozwiązywanie problemu komiwojażera poprzez redukcję złożoności problemu do mniejszych podproblemów, ale jest to osiągane kosztem długości trasy. Pozytywny wpływ technik klasteryzacji był już zauważalny w znacznym skróceniu czasu pracy algorytmów na niewielkich instancjach liczących od 52 do 127 miast. W przypadku większych instancji stosowanie klasteryzacji jest koniecznym procesem do zastosowania użytych w pracy algorytmów heurystycznych.

Najskuteczniejszą techniką grupowania danych w przypadku problemu komiwojażera okazała się metoda hierarchicznej klasteryzacji aglomeracyjnej. Dzięki niej uzyskano najkrótszy czas pracy algorytmów, jednocześnie osiągając najmniejsze pogorszenie długości trasy.