

Algorytmy Metaheurystyczne

Laboratorium1 - Problem komiwojażera

Autorzy: Zofia Stypułkowska, Bartosz Grelewski

Repozytorium GitHub: <https://github.com/Sophie1227/Metaheurystyka1>

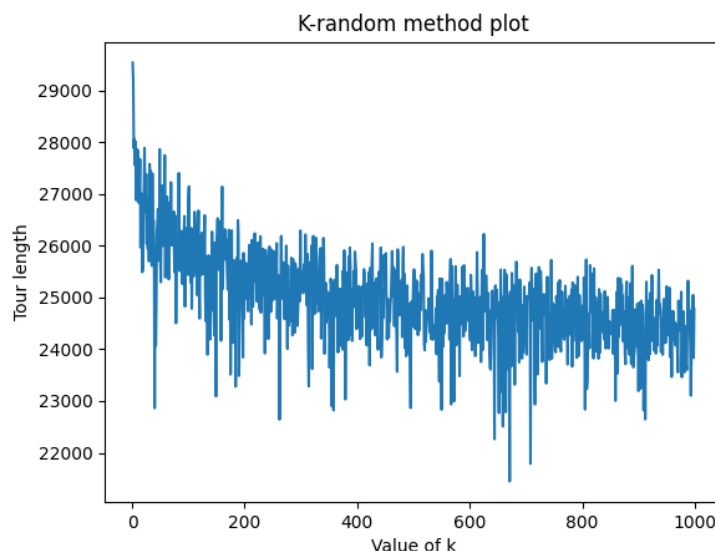
Rozważany Problem

Problem komiwojażera to jedno z najlepiej znanych zagadnień optymalizacyjnych polegające na znalezieniu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Innymi słowy mając zadany układ miast wraz z odległościami między nimi, sprzedawca musi wykonać kurs, odwiedzając każde z nich dokładnie raz i wracając do miasta wyjściowego. Warunkiem jest jednak pokonanie takiej trasy możliwie najmniejszym kosztem, czyli pokonując możliwie najmniejszą trasę. Największa trudność polega na dużej liczbie danych do analizy, a co za tym idzie czasie potrzebnym na znalezienie rozwiązania. W przypadku symetrycznym dla n miast liczba możliwych permutacji wynosi $\frac{(n-1)!}{2}$. Więc przykładowo dla danych z bays29 mamy $\frac{(29-1)!}{2} = 1,5 * 10^{29}$. Z powodu tak dużej liczby danych do sprawdzenia rozwiązując problem komiwojażera możemy to zrobić na 3 sposoby, skupiając się na najlepszym wyniku, najszybszym czasie pracy programu lub starając się znaleźć balans między nimi.

Wykorzystane Metody

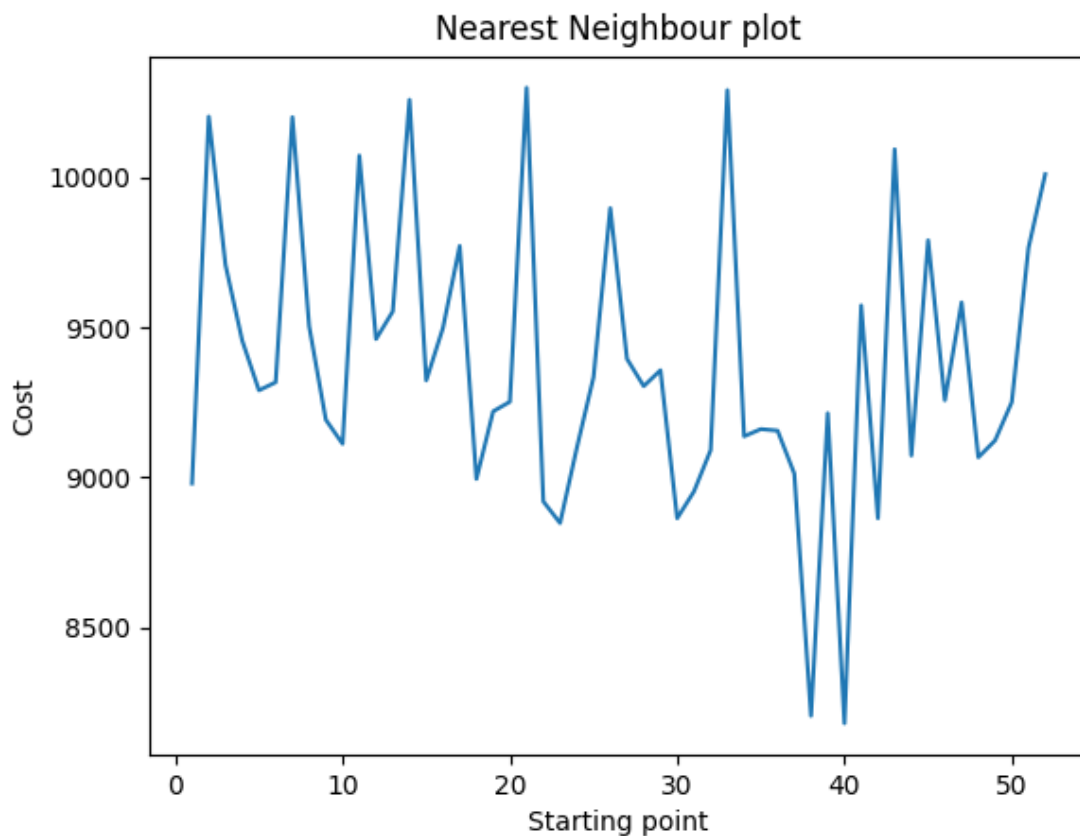
Metoda k-random

Generujemy k losowych dopuszczalnych rozwiązań i szukamy najlepszego z nich. Taka metoda choć prosta do implementacji ma jednak swoje wady. W podstawowej wersji takiego podejścia generujemy jedynie jedną permutację co sprawia, że rozwiązanie, które otrzymujemy może być bardzo słabe, a nawet najgorsze z możliwych. Aby zredukować szanse na taką sytuację generujemy k możliwych rozwiązań zamiast jednego. Losowo próbkujemy zbiór rozwiązań i zwracamy najlepsze z nich. Teoretycznie, gdy $k \rightarrow \infty$ prawdopodobieństwo znalezienia optymalnego rozwiązania jest równe 1. Jest to jednak bardzo nieoptyczne pod względem czasu potrzebnego na otrzymanie wyniku. Metoda ta pozwala jednak na proste balansowanie pomiędzy czasem działania i jakością otrzymanego wyniku.



Metoda najbliższego sąsiada

Jest to algorytm zachłanny bazujący na założeniu, że lokalnie optymalne decyzje są również globalnie optymalne. Szybkość tego podejścia polega na małej ilości danych w każdym kroku decyzyjnym. Zamiast przeszukiwać wszystkie możliwe rozwiązania dla każdego punktu sprawdzamy odległości do pozostałych punktów w grafie. Więc dla każdego punktu wyjściowego mamy łącznie $\sum_{i=1}^{n-1} i$ rozwiązań do sprawdzenia. Zaczynamy od wybrania w dowolny sposób wierzchołka wyjściowego i przeszukujemy pozostałe, nieodwiedzone jeszcze wierzchołki w poszukiwaniu najbliższego do niego sąsiada. Robimy to w pętli, aż odwiedzimy wszystkie miasta. Wadą takiego podejścia jest zależność jakości rozwiązania od wyboru punktu wyjściowego, ponieważ po wyborze danego miasta „usuwamy” część krawędzi, które mogły być w kolejnych krokach lokalnie najlepsze.



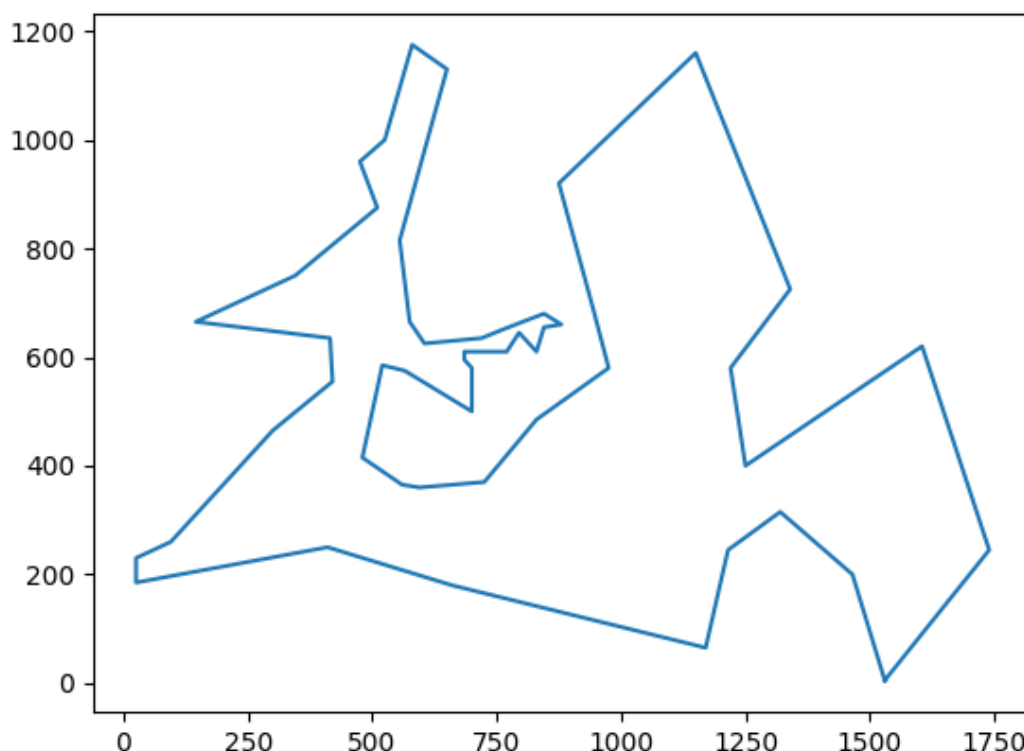
Rysunek 1- Zależność długości trasy od punktu wyjściowego dla berlin52

Rozszerzona metoda najbliższego sąsiada

Podejście to jest bardzo podobne do podstawowego algorytmu najbliższego sąsiada, usuwa jednak zależność od punktu wyjściowego i znajduje globalnie najlepsze rozwiązanie dla swojej podstawowej wersji. Nie usuwa jednak problemu związanego z brakiem dostępu do wykorzystanych już krawędzi, które mogły okazać się lokalnie lepsze.

Metoda 2-OPT

Jest to heurystyka dedykowana dla problemu komiwojażera, jednak przy drobnych zmianach może znaleźć zastosowanie dla innych zagadnień. Jest to algorytm należący do algorytmów popraw. Losujemy pierwsze rozwiązanie, a algorytm próbuje je poprawić. Poprzez lokalne przeszukiwanie tworzymy „sąsiadów” aktualnego rozwiązania i wybieramy najlepszego z nich. Sprawdzamy, czy jest ono lepsze od naszego aktualnego rozwiązania i jeśli jest to aktualizujemy najlepsze rozwiązanie i powtarzamy dla niego proces. Jeśli nie jest ono lepsze to kończymy pracę algorytmu. W naszym rozwiązaniu korzystamy z metody invert, czyli zamiany miejscami 2 wierzchołków. Pozwala to na „rozplątanie” nieoptymalnych pętli i poprawienie wylosowanego rozwiązania, często do poziomu otrzymania globalnie najlepszej trasy. Jest to jednak metoda należąca do NP-trudnych, czyli niedająca gwarancji znalezienia optymalnego rozwiązania w rozsądnym czasie wykonywania pętli algorytmu.



Rysunek 2- Wizualizacja przykładowej trasy dla rozwiązania 2-OPT z wykorzystaniem danych berlin52

Wnioski

Każdy z powyższych algorytmów skupia się na innych celach. Mimo, że usiłują one rozwiązać ten sam problem to każdy z nich ma inne priorytety dotyczące rozwiązania. Mimo, że teoretycznie algorytmy rozszerzonego najbliższego sąsiada i 2-OPT dają najlepsze rozwiązania, to są one niewydajne czasowo. Z kolei algorytm k-random jest szybki dla niskiej wartości k, jednak nie daje on dobrych wyników. W naszym programie policzyliśmy jednak również rozwiązania dla wyników metody k-ranom i najbliższego sąsiada po rozplątaniu przez 2-OPT. Dla losowej permutacji długość trasy wyniosła 27076, a po rozplątaniu za pomocą algorytmu 2-OPT zmniejszyła się do 8111. Z kolei dla rozwiązania znalezionej metodą najbliższego sąsiada, wstępna trasa wyniosła 8980, a po rozplątaniu 7842.

Widzimy więc, że zastosowanie algorytmu 2-OPT nałożonego na algorytm najbliższego sąsiada daje nam najlepsze wyniki, ale jest on również dość wydajny czasowo. Jeśli jednak zależy nam szczególnie na czasie, to warto rozważyć zastosowanie czystej metody najbliższego sąsiada lub rozszerzonej metody najbliższego sąsiada, które dają stosunkowo dobre wyniki i mają mniejszą złożoność obliczeniową od 2-OTP.