Algorytmy metaheurystyczne

Sprawozdanie z problemu komiwojażera

Marek Traczyński (261748)
Kamil Walter (261698)
Rok: 2 Semestr: 4
Grupa: Pt 17:05

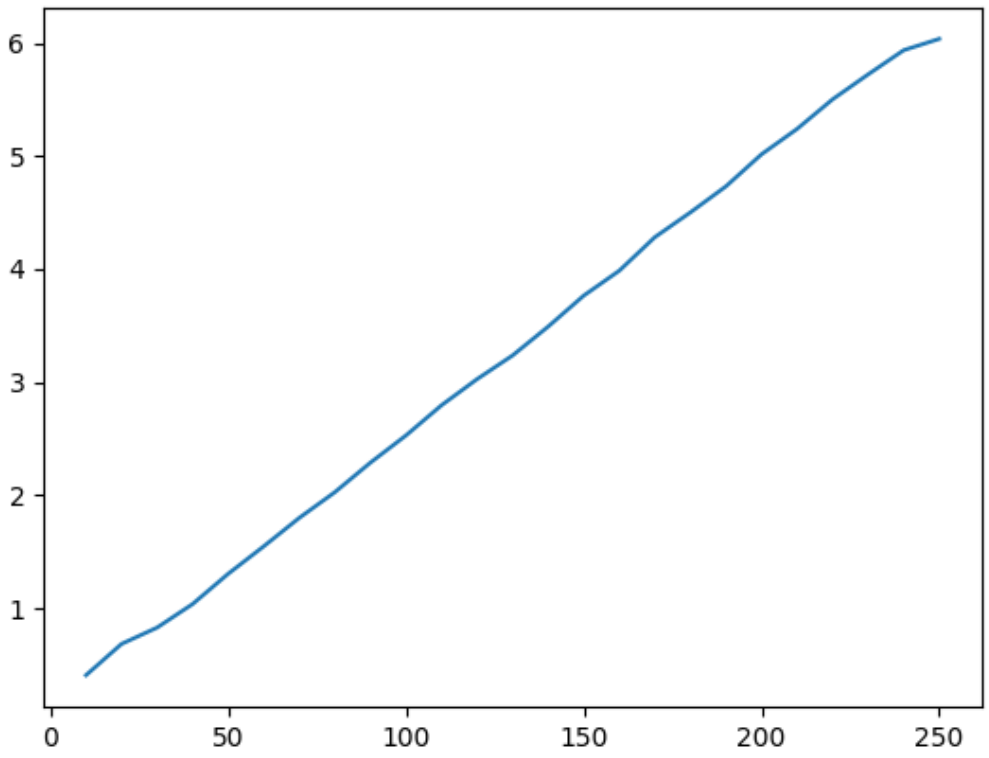
1. Wstęp

Celem pracy na laboratoriach było zbadanie problemu komiwojażera dla czterech algorytmów: k-random, najbliższego sąsiada, rozszerzonego najbliższego sąsiada oraz 2-opt. Głównym zadaniem było napisanie aplikacji umożliwiającej testowanie ww. algorytmów do obliczania najlepszej drogi dla wczytywanych i generowanych instancji. Program został napisany w języku Julia, a wykresy wygenerowane przy użyciu Pythona.

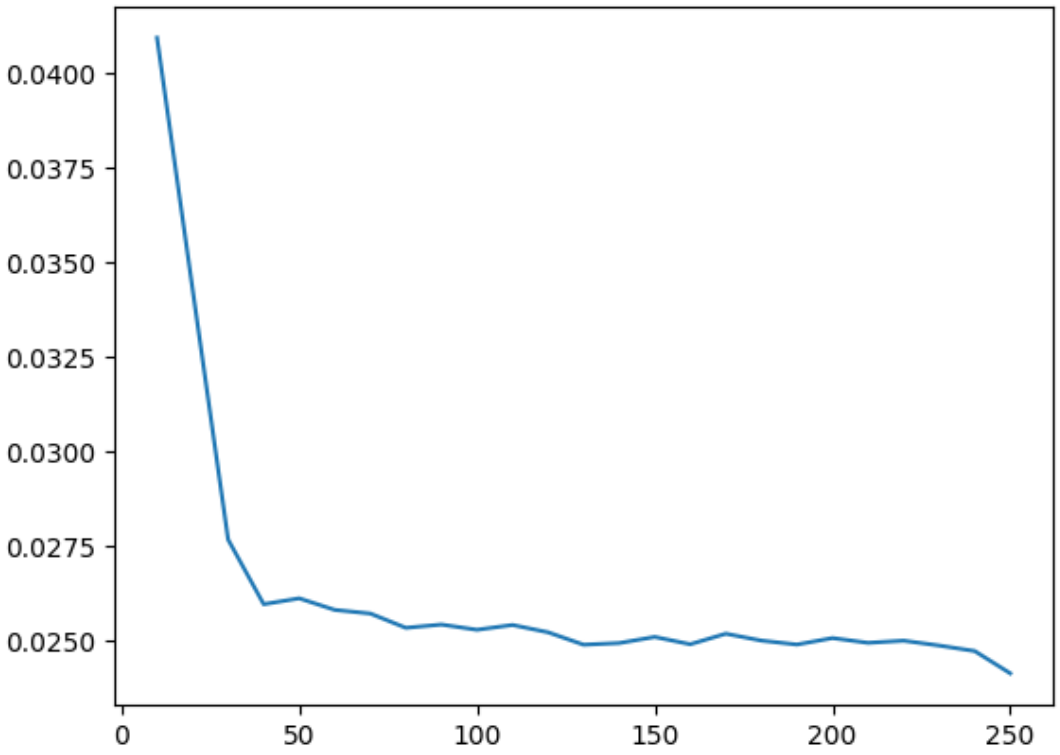
2. Złożoność obliczeniowa algorytmów

Algorytmy były testowane na losowo wygenerowanych instancjach typu EUC_2D.

- k-random (dla wartości k = 100 000):



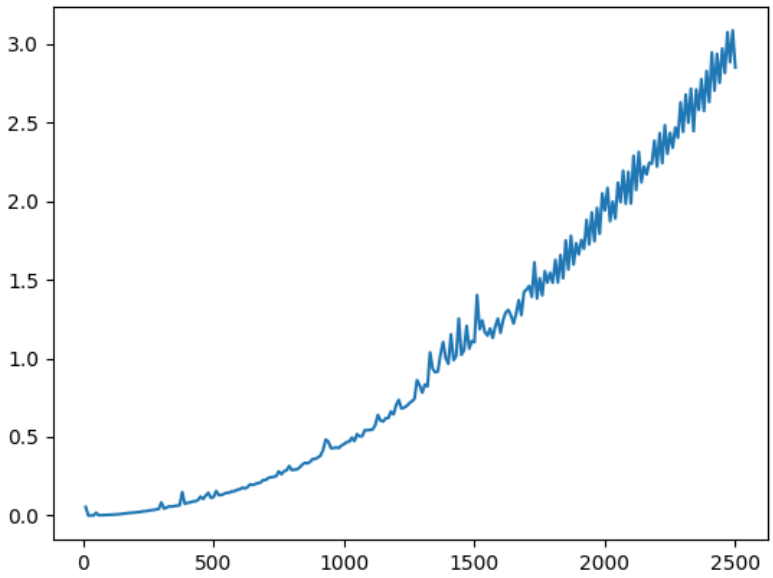
Wykres zależności czasu działania w [s] od wielkości instancji (10-250 node'ów)



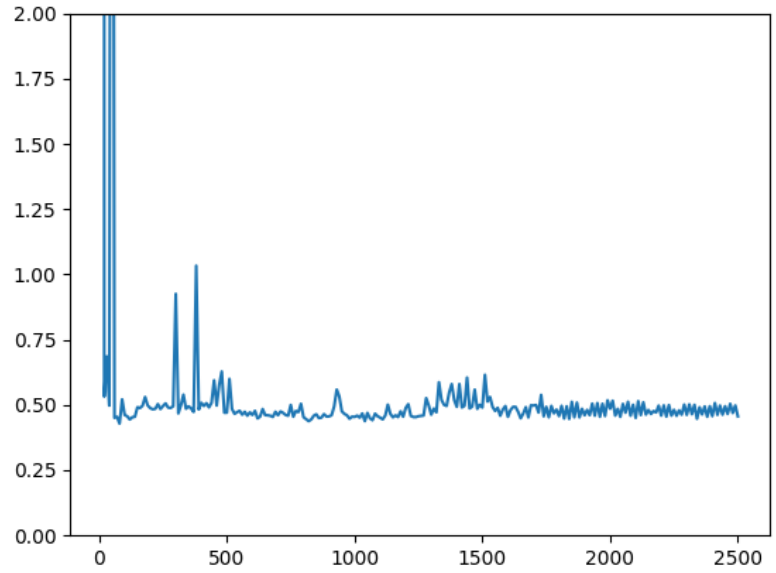
Wykres zależności czasu działania w [s] od wielkości instancji po podzieleniu przez wielkość (f(x)/x)

Nasz algorytm k-random posiada złożoność obliczeniową równą O(k*n), gdzie k oznacza liczbę losowań, a n – wielkość instancji problemu. Pierwszy wynik wyraźnie odstaje od reszty, lecz jest to spowodowane błędem pomiarowym wywoływanym przez sposób w jaki Julia mierzy czas za pomocą makra @elapsed (Błąd ten powtarza się również w reszcie algorytmów). Liniowa złożoność wynika z tego, że długość działania algorytmu jest uzależniona tylko od wielkości tablicy (czyli instancji), którą permutujemy.

- Najbliższy sąsiad:



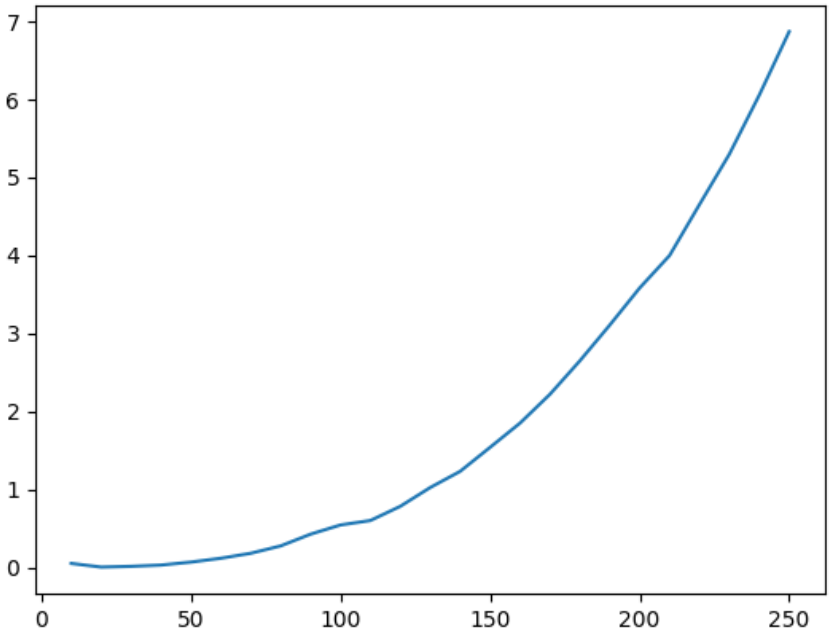
Wykres zależności czasu działania w [s] od wielkości instancji (10-250 node'ów)`



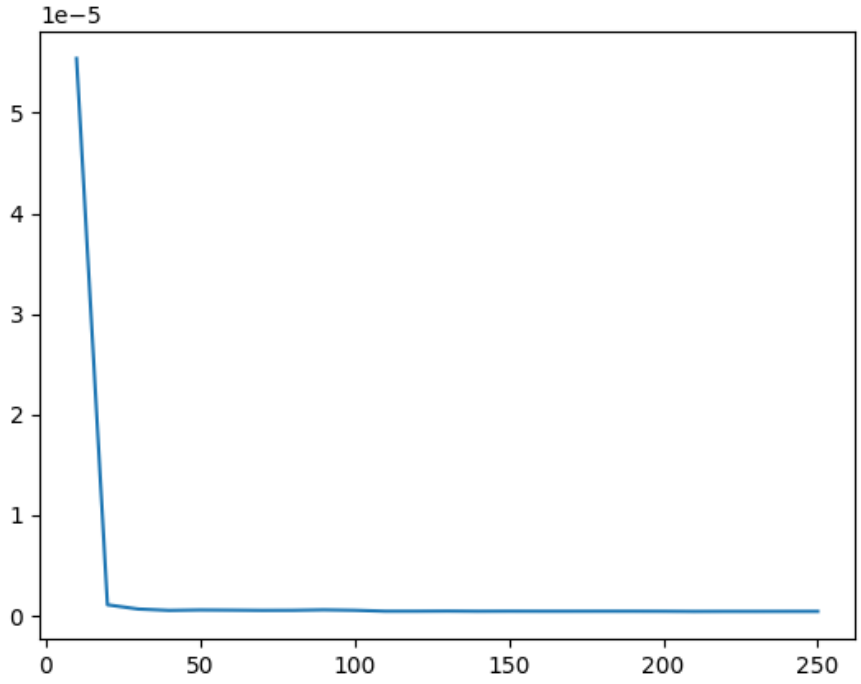
Wykres zależności czasu działania w [s] od wielkości instancji po podzieleniu przez wielkość do kwadratu (f(x)/x^2)

Po podzieleniu uzyskanych wartości przez wielkość podniesioną do kwadratu widać, że algorytm najbliższego sąsiada ma złożoność wielkości O(n^2). Taka złożoność wynika z tego, że algorytm w kolejnych krokach wykonuje n-1, n-2, n-3, …, 1 porównań.

- Rozszerzony najbliższy sąsiad:



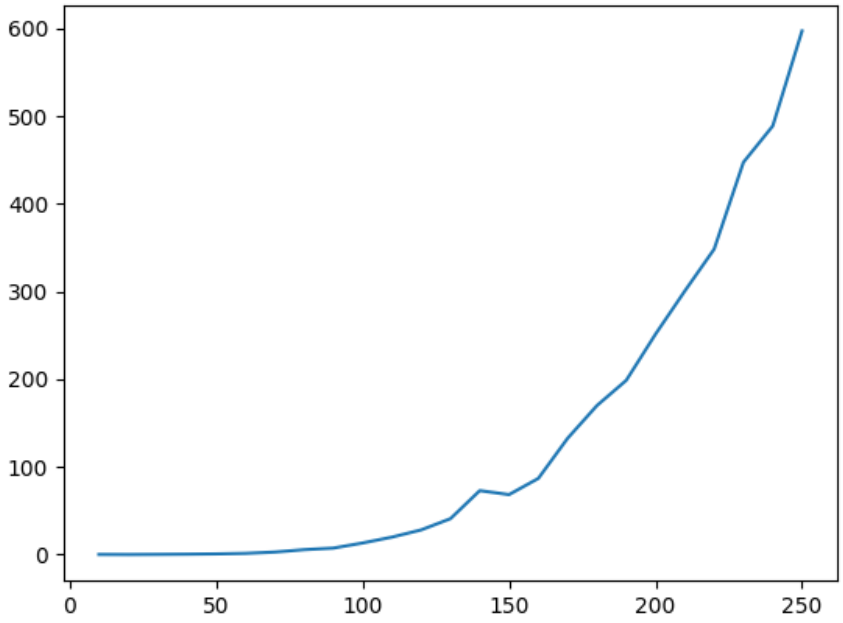
Wykres zależności czasu działania w [s] od wielkości instancji (10-250 node'ów)`



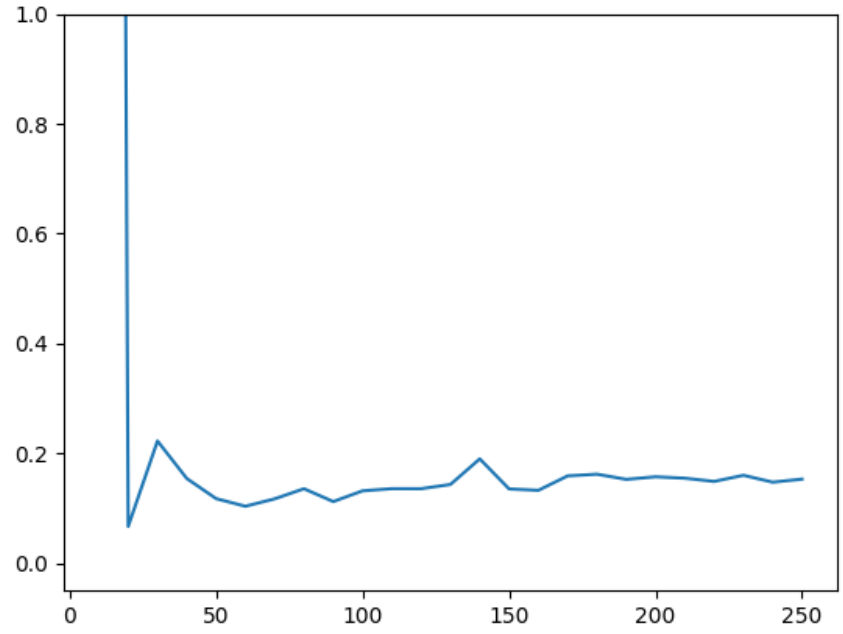
*Wykres zależności czasu działania w [s*10^-5] od wielkości instancji po podzieleniu przez wielkość do kwadratu (f(x)/x^3)*

Rozszerzenie najbliższego sąsiada działa na zasadzie wywołania algorytmu najbliższego sąsiada dla wszystkich wierzchołków początkowych i wybraniu najlepszego wyniku. Skoro najbliższy sąsiad ma złożoność obliczeniową O(n^2), to wywołanie go n-krotnie powoduje, że otrzymujemy złożoność O(n^3).

- 2-opt:



Wykres zależności czasu działania w [s] od wielkości instancji (10-250 node'ów)`



Wykres zależności czasu działania w [s] od wielkości instancji po podzieleniu przez wielkość do kwadratu (f(x)/x^4)

Algorytm polega na porównywaniu sąsiedztw permutacji danych przez transpozycję. Testy wspomagane drugim wykresem wykazały, że nasz algorytm posiada złożoność obliczeniową równą O(n^4).

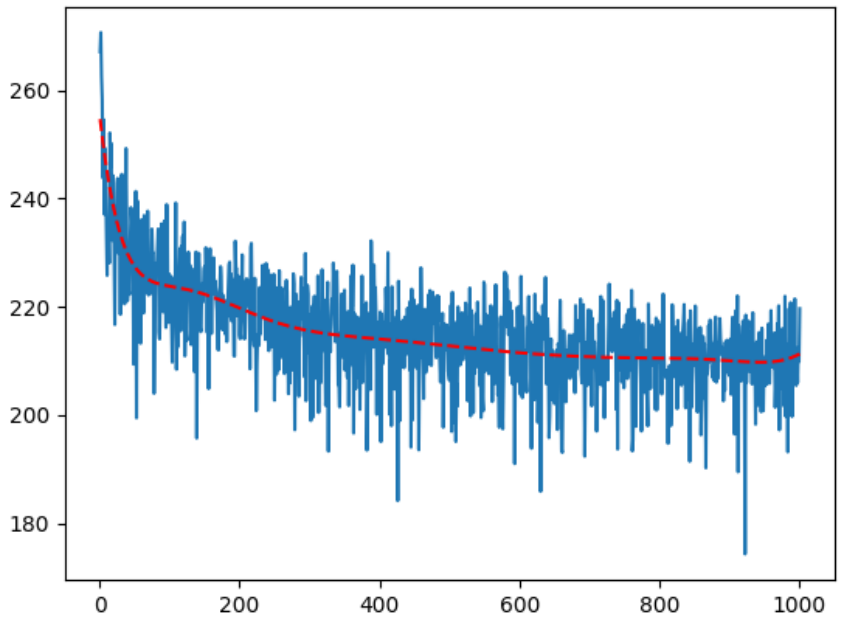
3. Porównanie algorytmów

- Najbliższy sąsiad vs rozszerzony najbliższy sąsiad:

Jak napisaliśmy wyżej, rozszerzenie wspomnianego algorytmu, polega na wywołaniu wersji podstawowej dla każdego wierzchołka początkowego. Stąd dla każdej instancji problemu pewnym jest, że rozszerzony najbliższy sąsiad wygeneruje nam krótszą lub równą trasę.

- K-random:

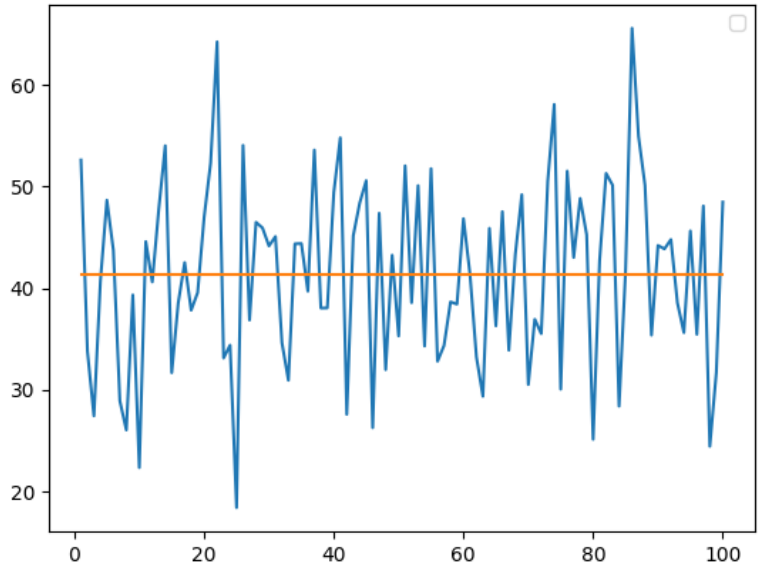
Algorytm k-random jest algorytmem bardzo prostym i naiwnym. Dla instancji problemu o rozmiarze n, otrzymujemy n! potencjalnych permutacji. Aby algorytm k-random generował dostatecznie dobre wyniki, potrzeba aby liczba losowań była proporcjonalna do n!. Jednak z wykresu poniżej wynika, że w praktyce uzyskane wyniki różnią się od siebie co raz mniej, pomimo zwiększania ilości losowań.



Wykres zależności PRD w %, od liczby losowań k w algorytmie k-random dla instancji berlin52.tsp. Czerwona linia to wielomianowe przybliżenie otrzymanych wyników.

- 2-opt:

Algorytm 2-opt ma największą złożoność obliczeniową, jednak daje on najlepsze wyniki. Mimo to otrzymane PRD waha się mocno w zależności od permutacji startowej (w naszym przypadku generowanej losowo). Średnie PRD jakie uzyskaliśmy dla 100 uruchomień algorytmu wynosi około 40%.



Wykres zależności PRD w %, od liczby uruchomień algorytmu 2-opt dla instancji berlin52.tsp

4. Wnioski

Z wyżej uzyskanych wyników wyciągamy wniosek, że algorytm k-random jest nieopłacalny, ponieważ wymaga bardzo dużej liczby powtórzeń i finalnie daje i tak gorsze wyniki, niż pozostałe algorytmy. Najbliższy sąsiad jest jednoznacznie przyćmiewany przez jego rozszerzony wariant. Dla odpowiednio dużych problemów algorytm rozszerzony dostarcza nieznacznie gorsze wyniki do algorytmu 2-opt, a działa w znacznie krótszym czasie. Jednakże, przy pomocy mocy obliczeniowej najnowszych komputerów, algorytm 2-opt wykonuje się w akceptowalnym czasie, więc naszym zdaniem jest najlepszym z wyżej testowanych algorytmów.