Sprawozdanie Etap 1

Przedmiot	Algorytmy metaheurystyczne
Prowadzący	Dr inż. Radosław Idzikowski
Autorzy	Maciej Bazela (261743) Hubert Gruda (261734)
Grupa	Czwartek 13:15-15:00
Kod grupy	K03-66p
Język	Julia

Implementacja

Etap 1 polegał na implementacji heurystyk dla problemu komiwojażera oraz zbadaniu ich niektórych własności (czas działania, PRD).

Badaliśmy cztery heurystyki:

- metoda k-random
- metoda najbliższego sąsiada
- rozszerzona metoda najbliższego sąsiada
- algorytm 2-OPT

Implementacje powyższych algorytmów znajdują się w pliku algorithms.jl.

Badania przeprowadziliśmy na bibliotece TSPLIB, która zawiera przykładowe dane dla symetrycznego problemu komiwojażera, oraz na losowo generowanych grafach.

Funkcje generujące grafy o symetrycznych (euklidesowych) i asymetrycznych (losowych) wagach znajdują się w pliku generate.jl.

Badania

Pierwszy z testów BasicTSPTest polegał na uruchomieniu wybranej heurystyki na podanym datasecie z TSPLIB **k**-razy i wybraniu najlepszej wyliczonej wartości funkcji celu.
Program uruchamiający test wypisywał do konsoli wyliczane po kolei wartości oraz drukował wybraną najlepszą ścieżkę do konsoli oraz pliku w folderze ./plots.

W przypadku problemu komiwojażera, naszą funkcją celu była suma wag pomiędzy kolejnymi węzłami zadanej drogi.

Następne testy polegały na wyliczeniu trzech statystyk dla każdej heurystyki: czas działania danego algorytmu, PRD oraz minimalna wartość funkcji celu dla badanego problemu.

Dla każdej heurystyki wykonywaliśmy k testów na wybranym przez nas zbiorze danych z TSPLIB oraz na losowych grafach z podanego zakresu (i o podanej zmianie wartości ilości miast).

Implementacja badań na predefiniowanym zbiorze plików .tsp znajduje się w funkcji algorithmsTest w pliku testing.jl.

Te same badania, ale na losowych grafach znajdują się w funkcji
randomGraphsTest w tym samym pliku.

Wyniki

Wyniki testów algorithmsTest oraz randomGraphsTest zapisywaliśmy do formatu .json w folderze ./results/jsons.
Rozdzieliliśmy je na 3 różne kategorie:

- wyniki dla problemów z TSPLIB
- wyniki dla losowych grafów o wagach euklidesowych
- wyniki dla losowych grafów o losowych asymetrycznych wagach

Pliki wynikowe zawierają wartości podanych wyżej statystyk (czas, prd, najlepsza wartość funkcji celu) rozdzielone według ilości węzłów dla zadanego problemu.

Przygotowaliśmy dwa programy generujące powyższe pliki .json:

- ./tests/hardcodedTest.jl generujący wyniki dla predefiniowanego zbioru plików z TSPLIB (patrz: wyżej).
- ./tests/randomgraphsTest.jl generujący wyniki dla losowych grafów z zadanego przedziału.

Wykresy

Dla wygenerowanych wyników napisaliśmy program, który sczytuje pliki .json i dla podanych algorytmów tworzy ich wykresy.

Kod źródłowy można znaleźć w pliku ./tests/plotting.jl.

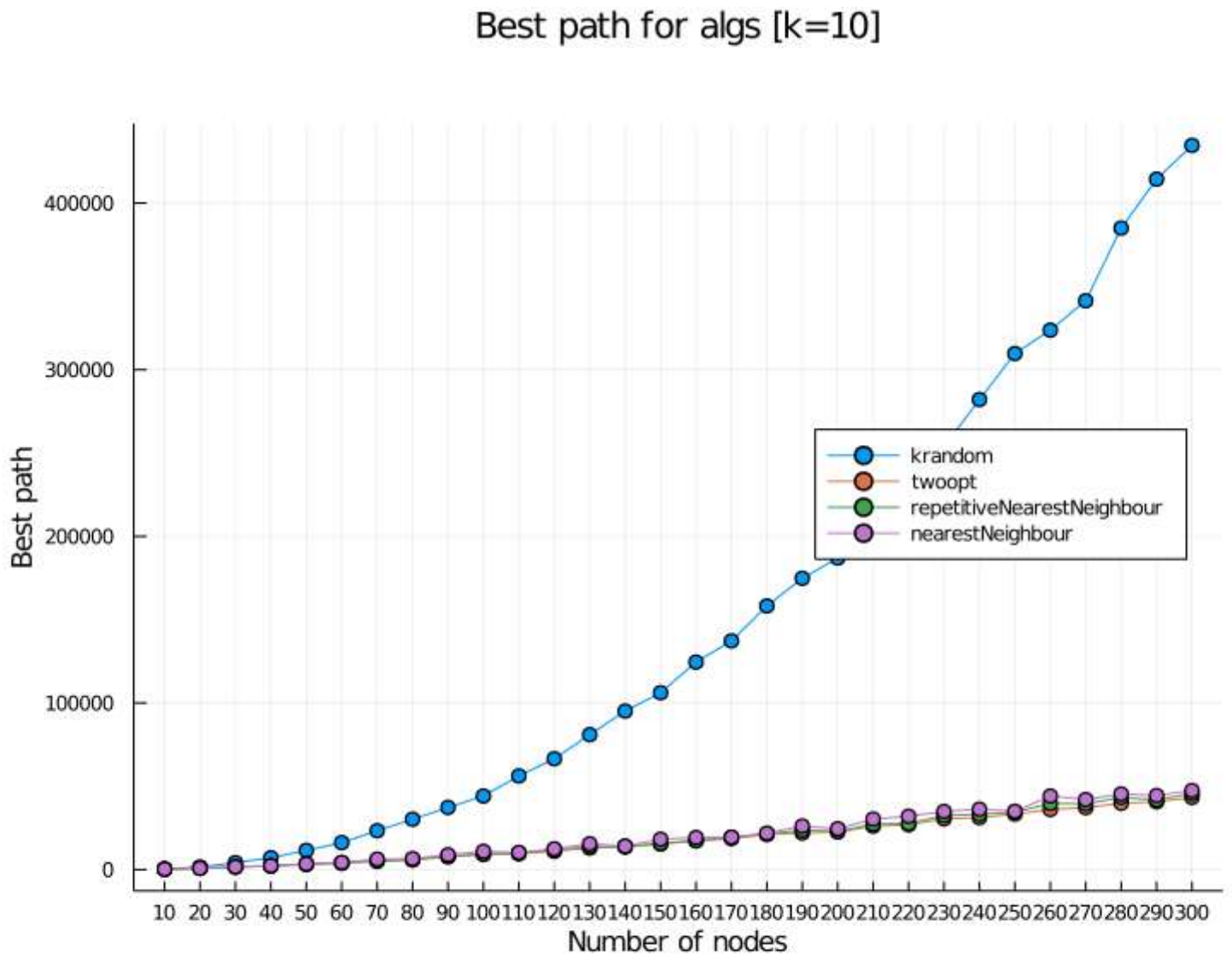
Dla wyników z folderu ./results/jsons wygenerowaliśmy wykresy dla każdego algorytmu. Znajdują się one w folderze ./results/plots.

Wnioski

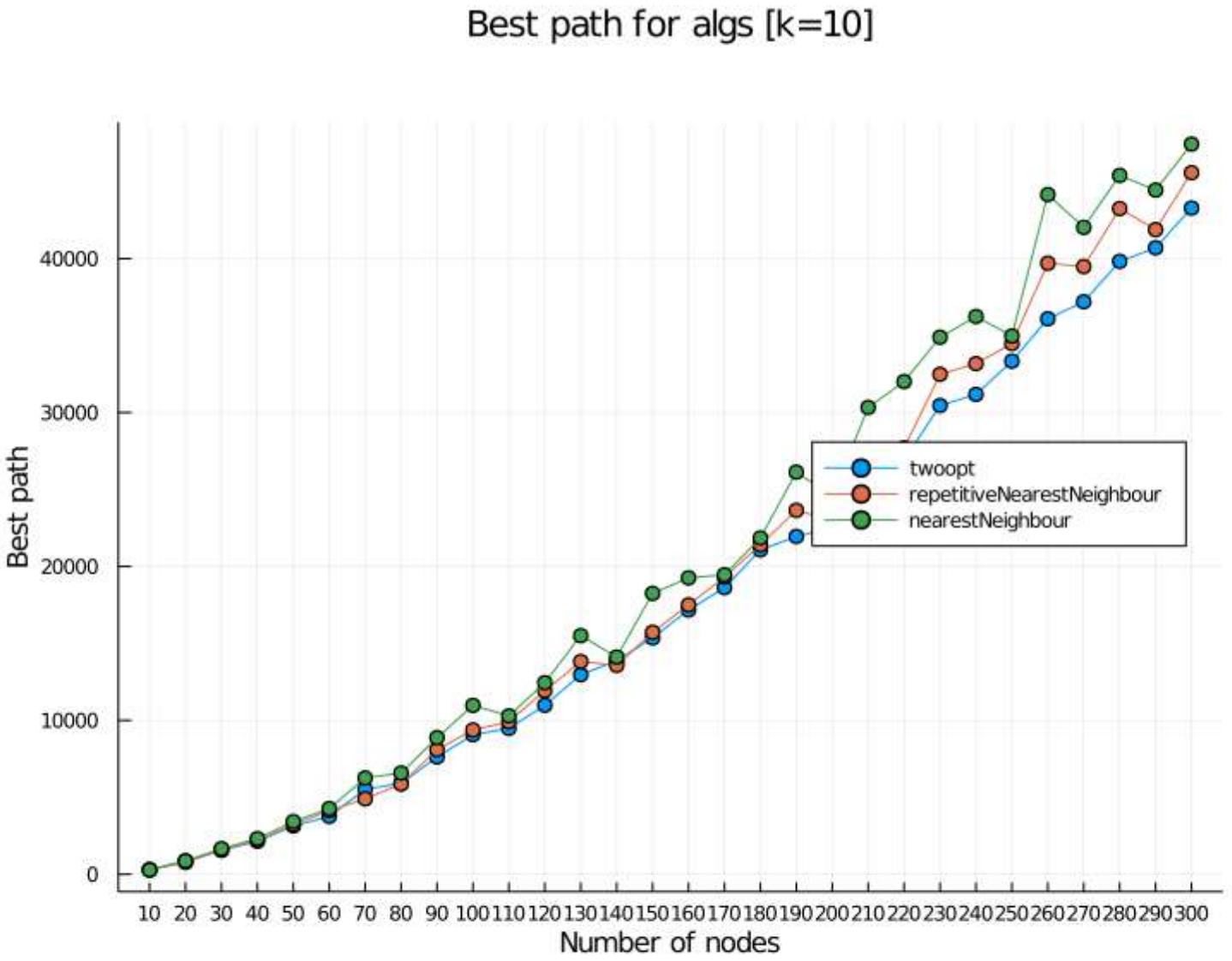
Dla każdej z badanych statystyk wyciągnęliśmy poniższe wnioski:

Najlepsza wartość funkcji celu (best generated path):

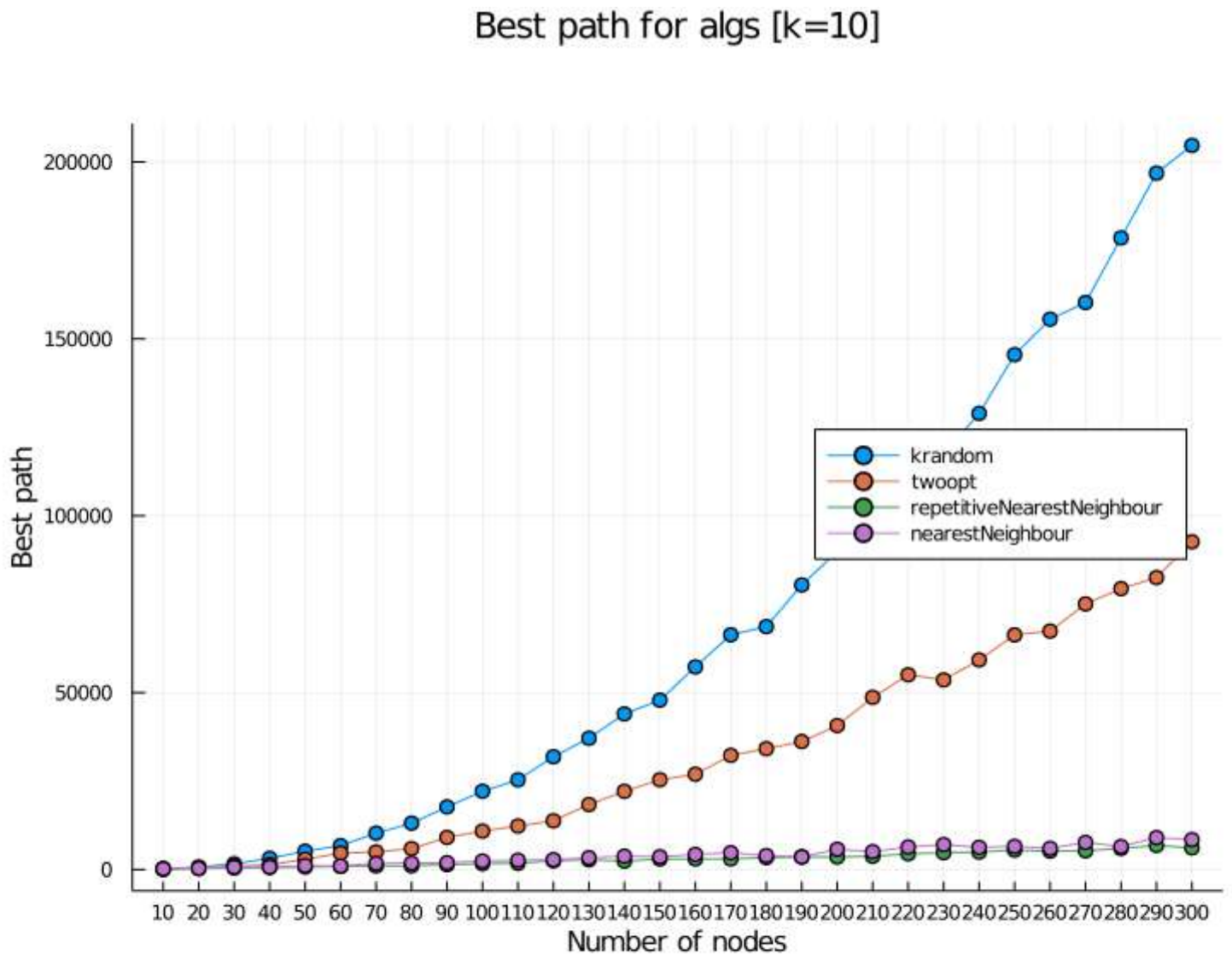
- przyapdek symetryczny:
  - wszystkie heurystyki:


Best path for algs [k=10]
Best path
Number of nodes
krandom
twoopt
repetitiveNearestNeighbour
nearestNeighbour
0
100000
200000
300000
400000
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300

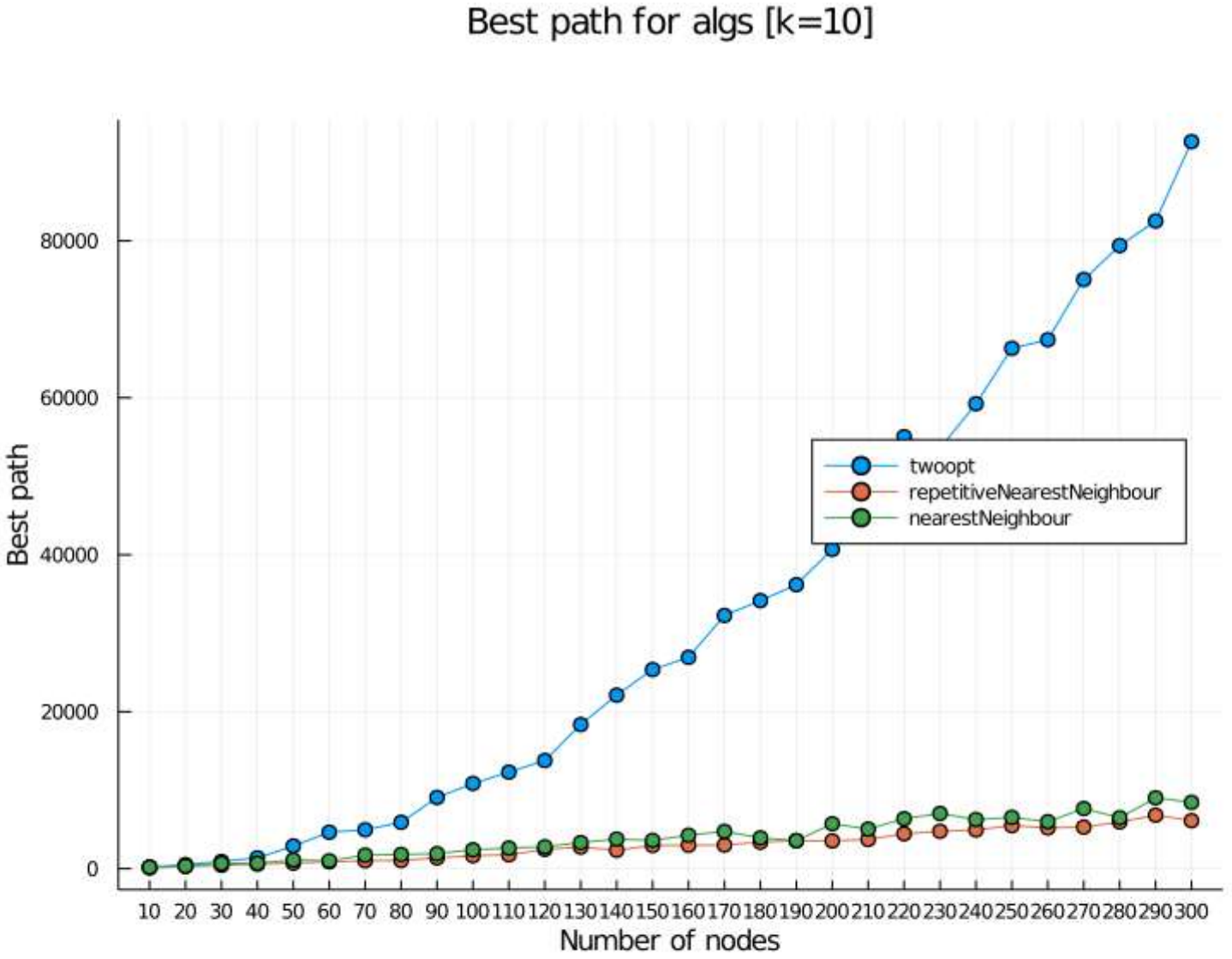
- bez k-random:



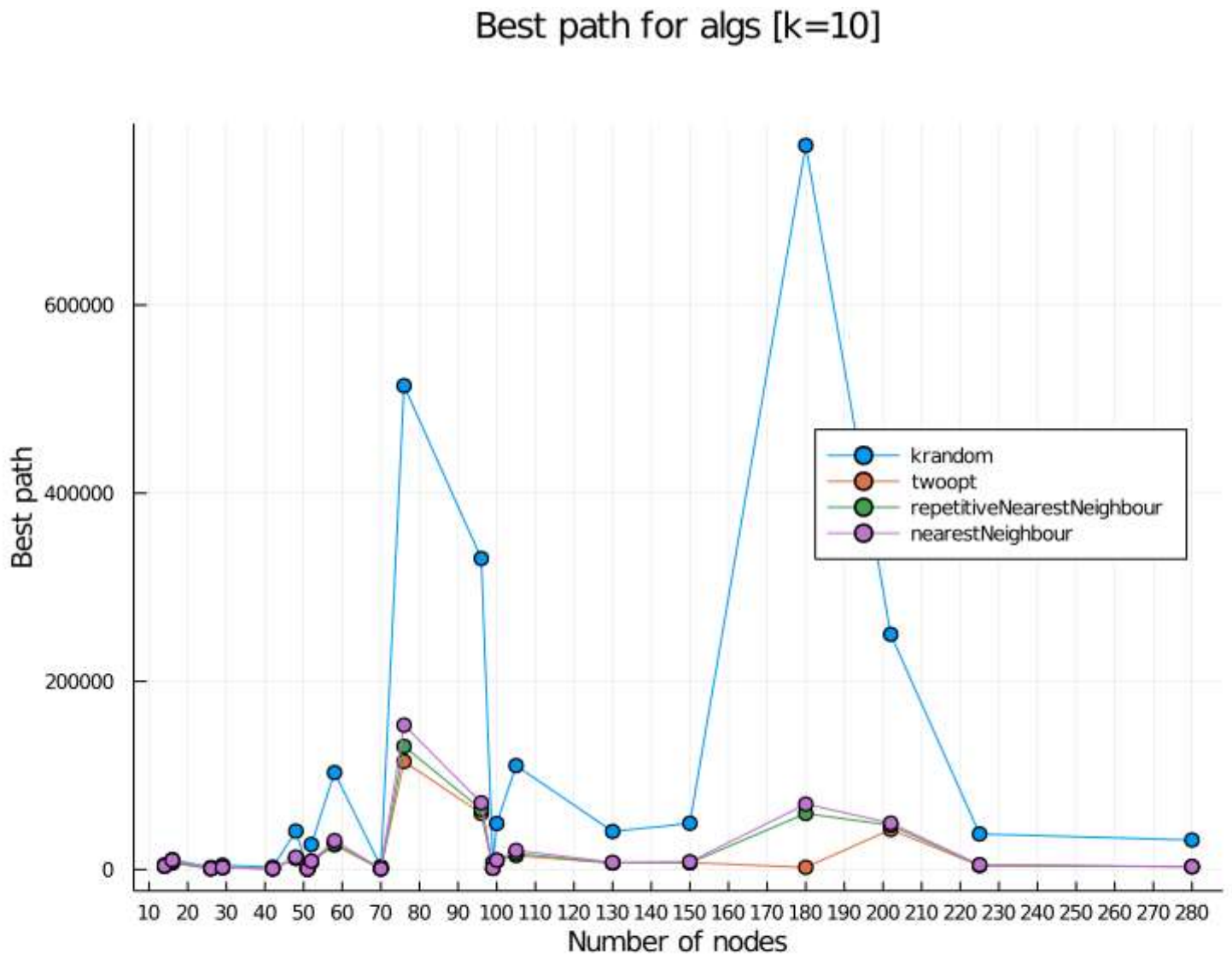
- przypadek asymetryczny:
  - wszystkie heurystyki:


Best path for algs [k=10]
Best path
Number of nodes
krandom
twoopt
repetitiveNearestNeighbour
nearestNeighbour
0
50000
100000
150000
200000
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300

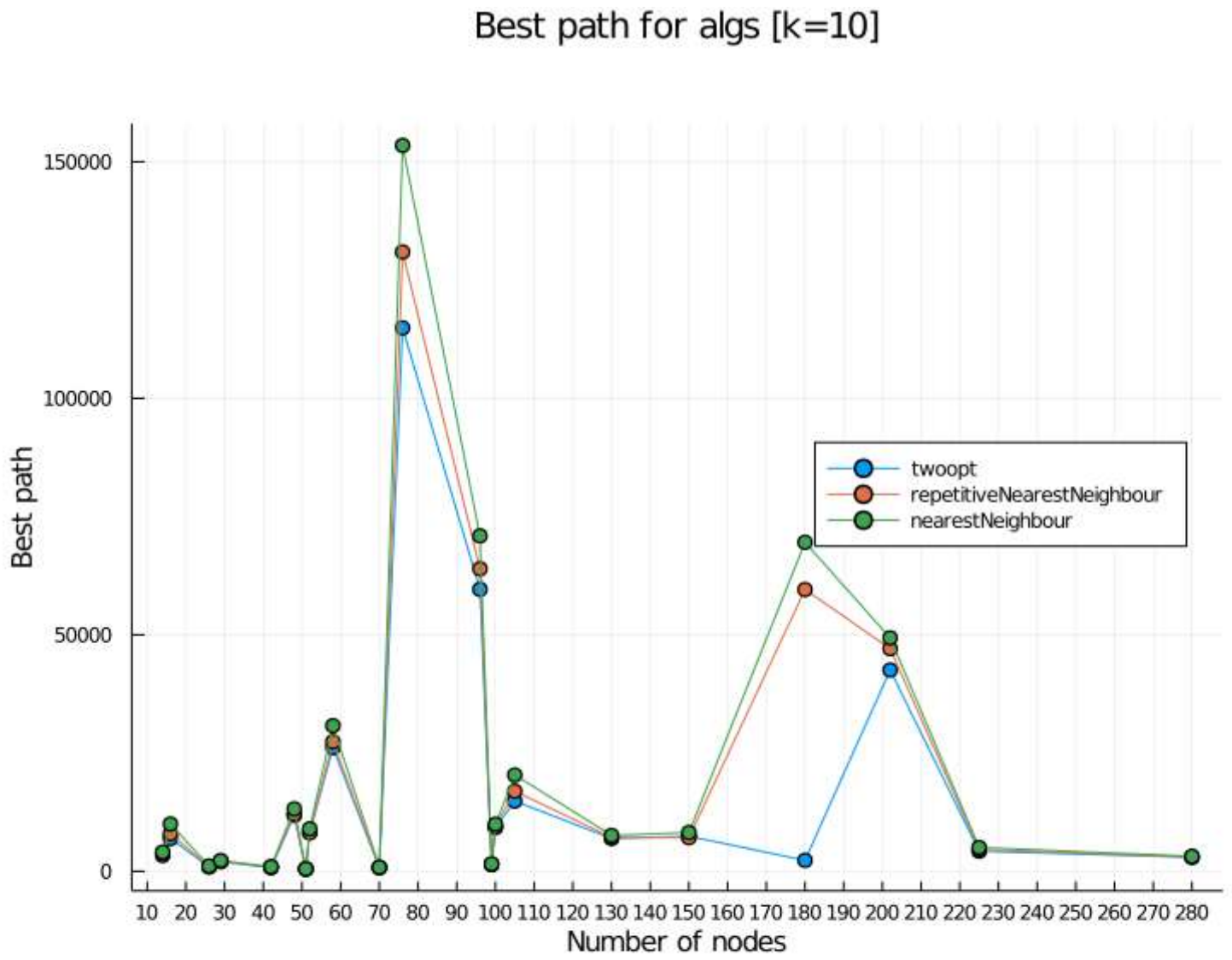
- bez k-random:



- problemy TSPLIB:
  - wszystkie heurystyki:


Best path for algs [k=10]
Best path
600000
400000
200000
0
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280
Number of nodes
krandom
twoopt
repetitiveNearestNeighbour
nearestNeighbour

- bez k-random:

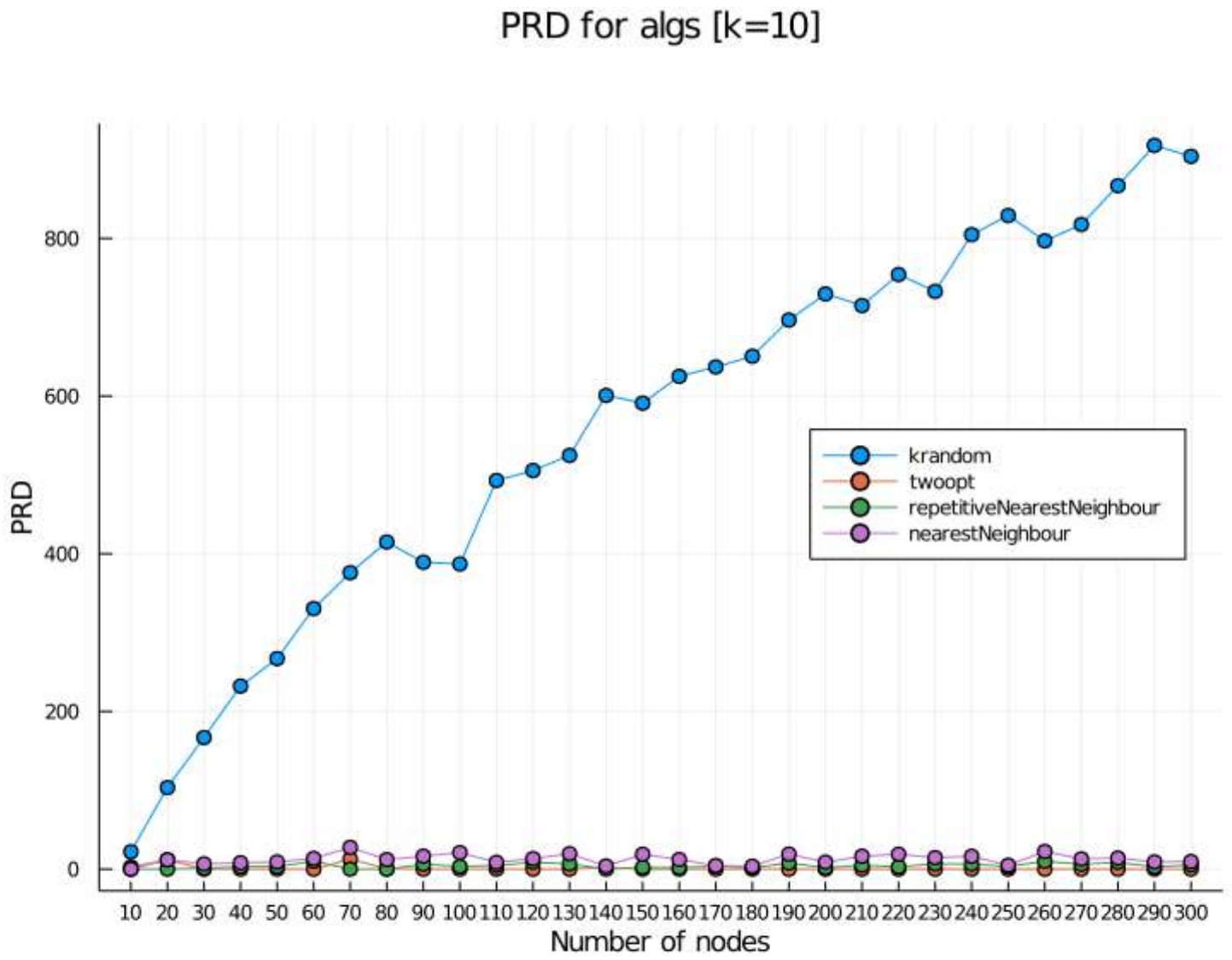


Dla algorytmów k-random i 2-opt przy większej ilości testów otrzymujemy lepsze rozwiązania.
Wynika to z zaimplementowanej losowości w obu algorytmach (k-random dosłownie generuje losową ścieżkę, 2opt ulepsza wcześniej losowo wygenerowaną ścieżkę).

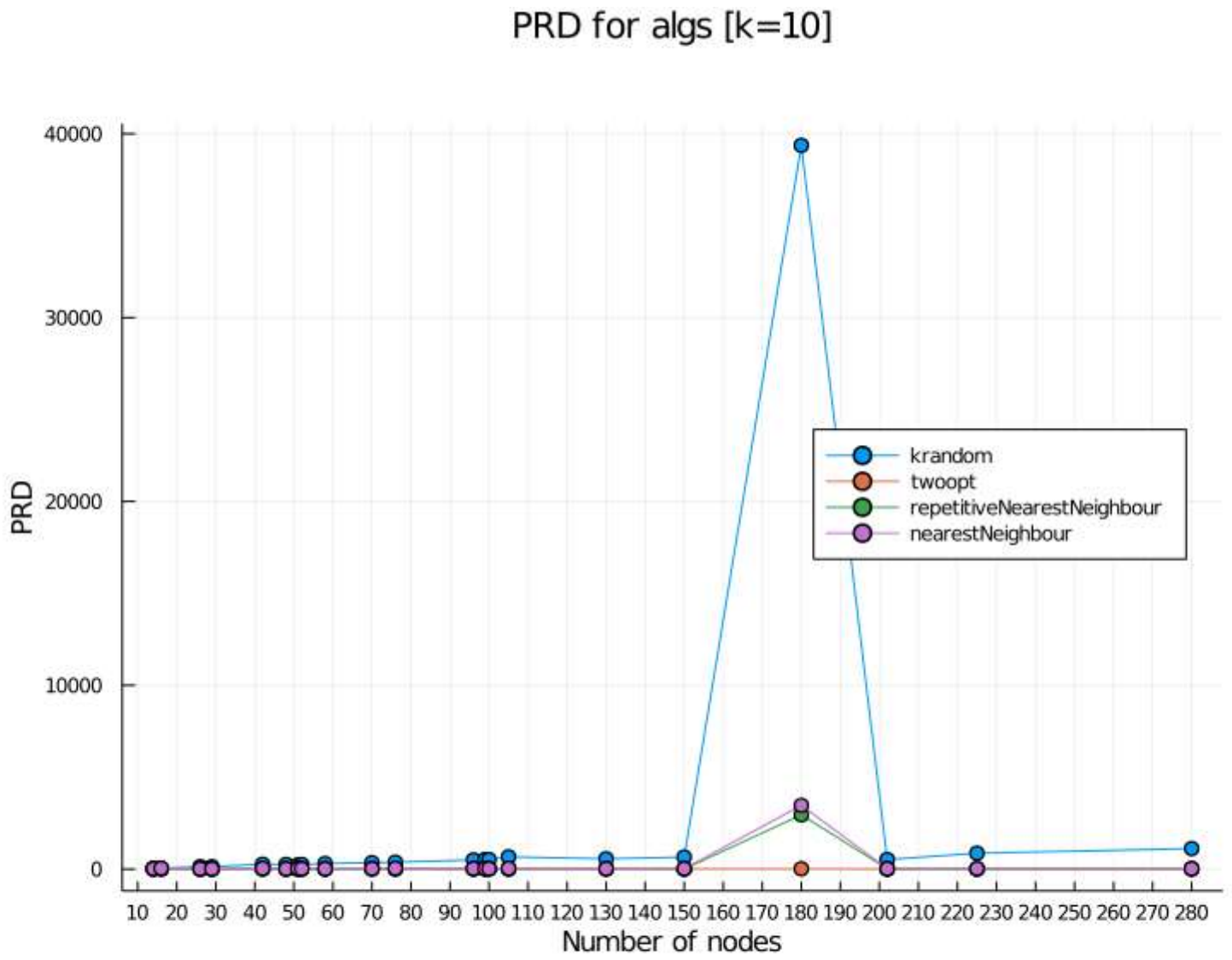
Dla obu metod najbliższego sąsiada ilość testów nie wpływa na generowanie lepszego rozwiązania, gdyż nie zawierają one losowości.
Dla każdego algorytmu ilosc węźłów nie wpływa w żaden sposób na generowanie najkrótszej drogi.
Z wykresów można wywnioskować, że najlepsze trasy dla przypadku symetrycznego generował algorytm 2opt, na drugim miejscu rozszerzona metoda sąsiada, będąca minimalnie lepsza od zwykłej metody sąsiada.
Dla przypadku asymetrycznego algorytm 2-opt jest dużo gorszy niż algorytmy najbliższego sąsiada.
W obu przypadkach nieporównywalnie gorsze wyniki otrzymaliśmy od metody k-random.

PRD:

- przyapdek symetryczny:
  - wszystkie heurystyki:



- problemy TSPLIB:
  - wszystkie heurystyki:



PRD określone jest wzorem: (p_gen - p_best) / p_best * 100%, gdzie:

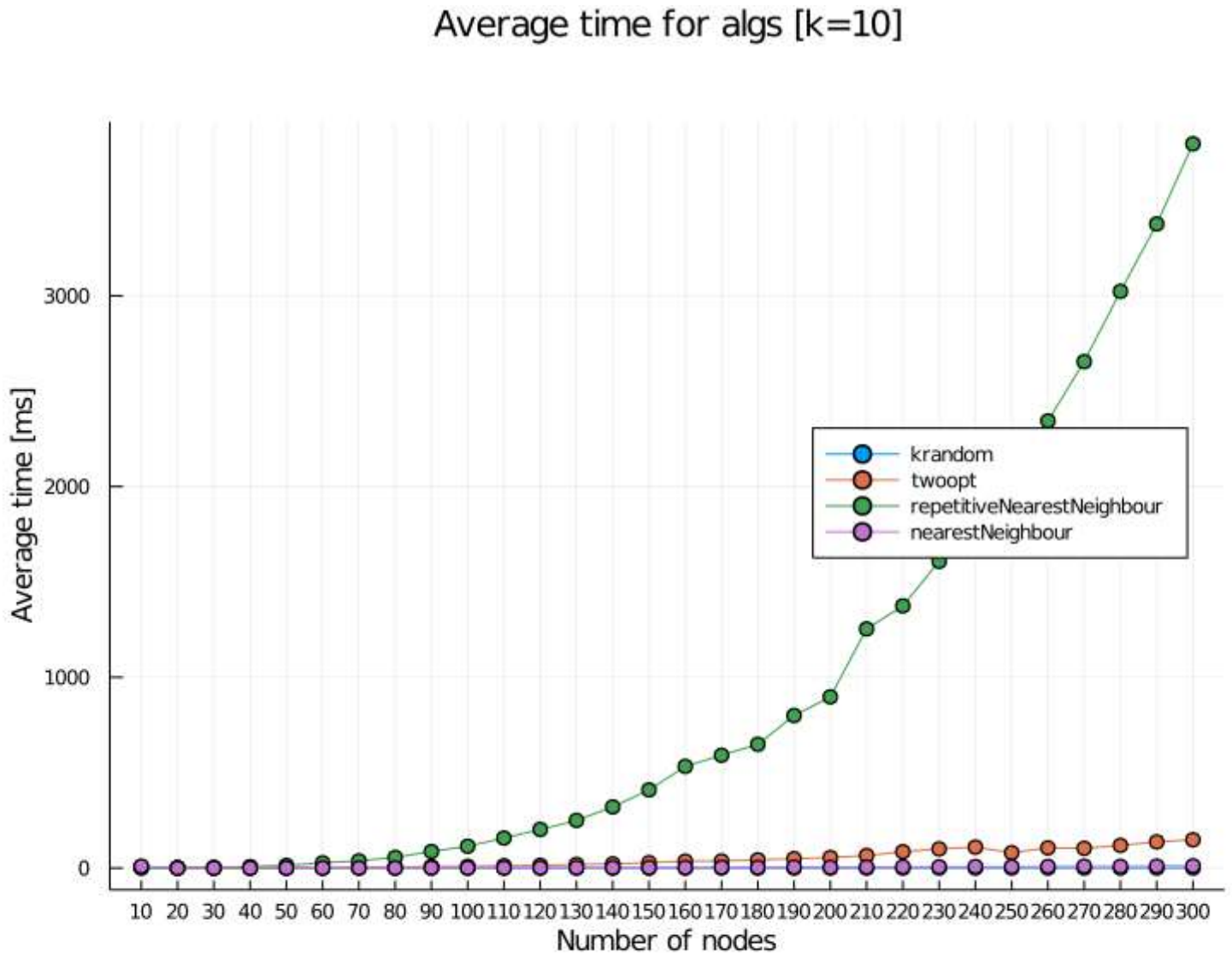
- p_gen to najlepsza wartość funkcji celu dla danej heurystyki,

- p_best to rozwiązanie optymalne dla zadanego problemu (w przypadku braku informacji o optymalnej ścieżce przyjmowaliśmy p_best jako wartość funkcji celu dla najlepszej heurystyki)

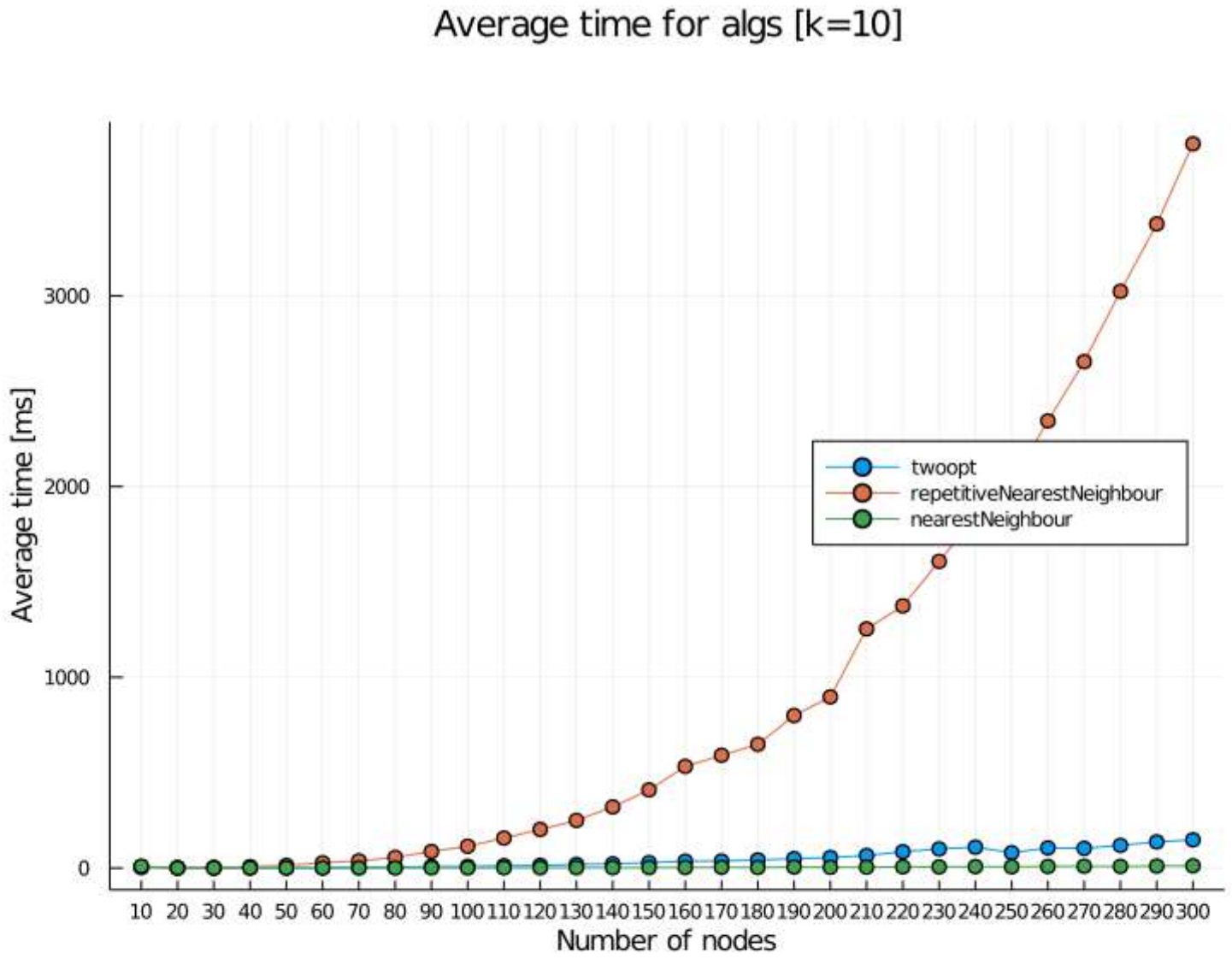
Z badań wynika, że ilość węzłów nie wpływa na PRD. Identycznie jak dla wartości funkcji celu dla najlepszej wygenerowanej ścieżki, najlepszy jest alogrytm 2-opt, druga jest rozszerzona metoda sąsiada, trzecia zwykła metoda sąsiada, a na samym końcu k-random.

Time Complexity:

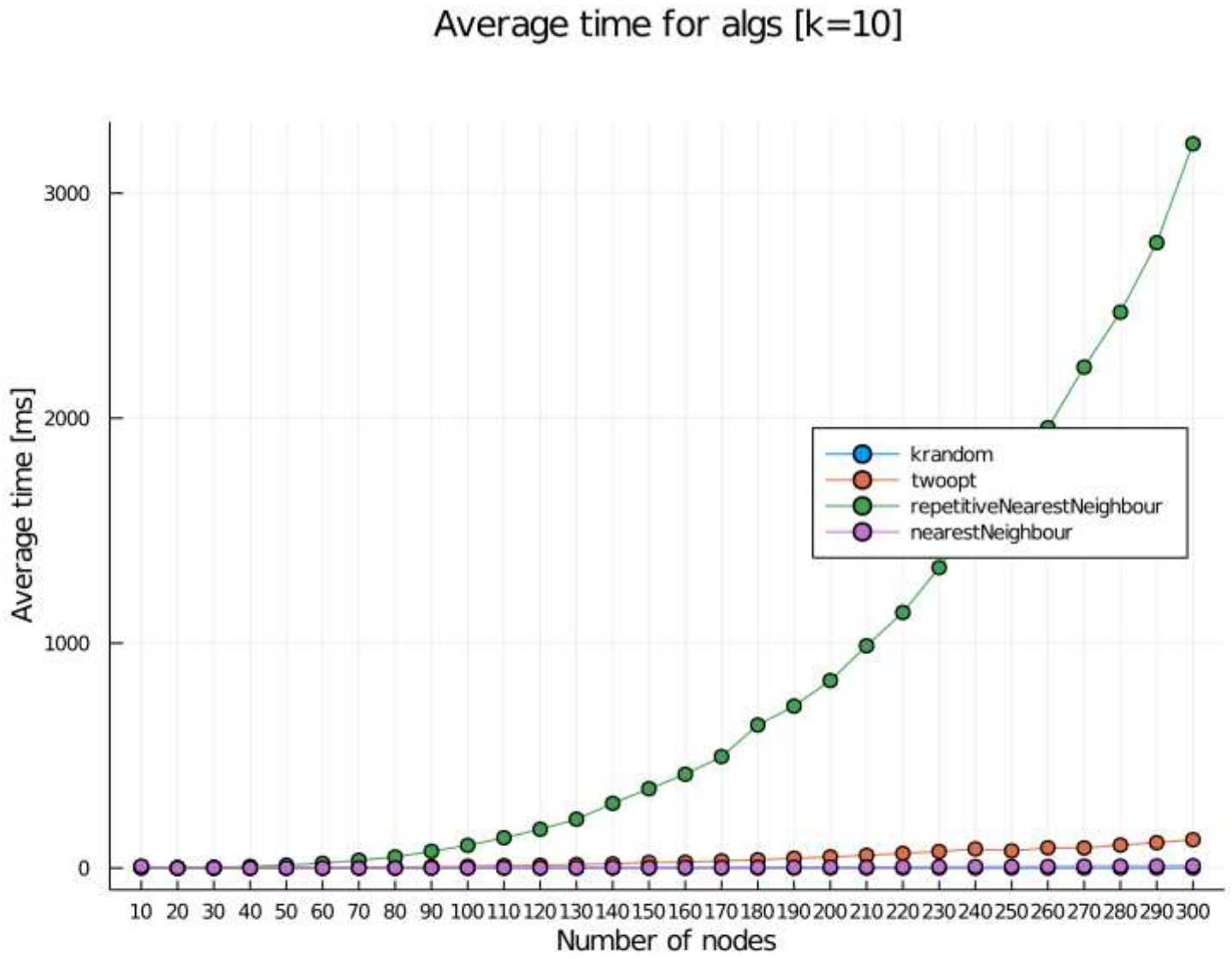
- przyapdek symetryczny:
  - wszystkie heurystyki:


Average time for algs [k=10]
Average time [ms]
0
1000
2000
3000
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300
Number of nodes
krandom
twoopt
repetitiveNearestNeighbour
nearestNeighbour

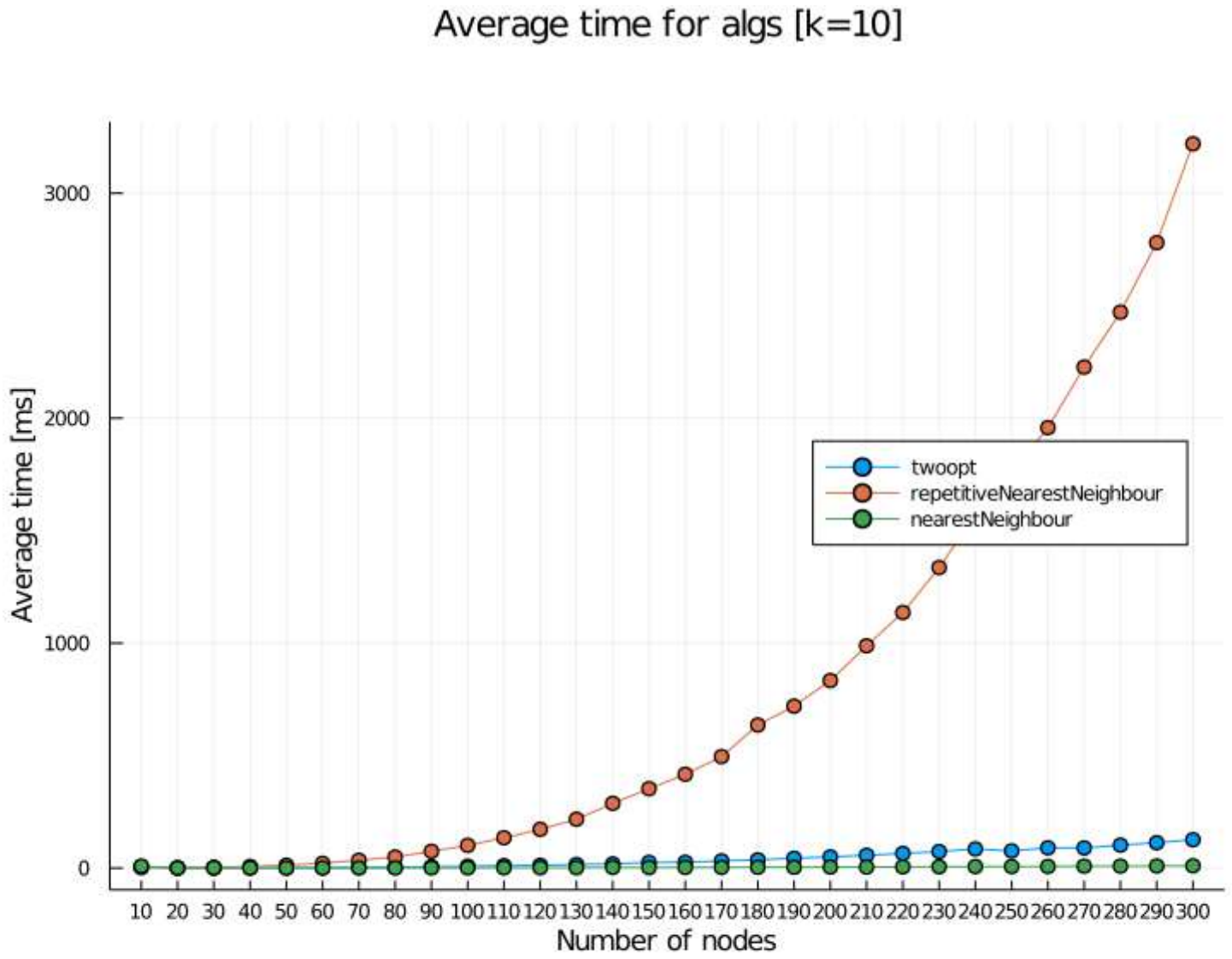
- bez k-random:



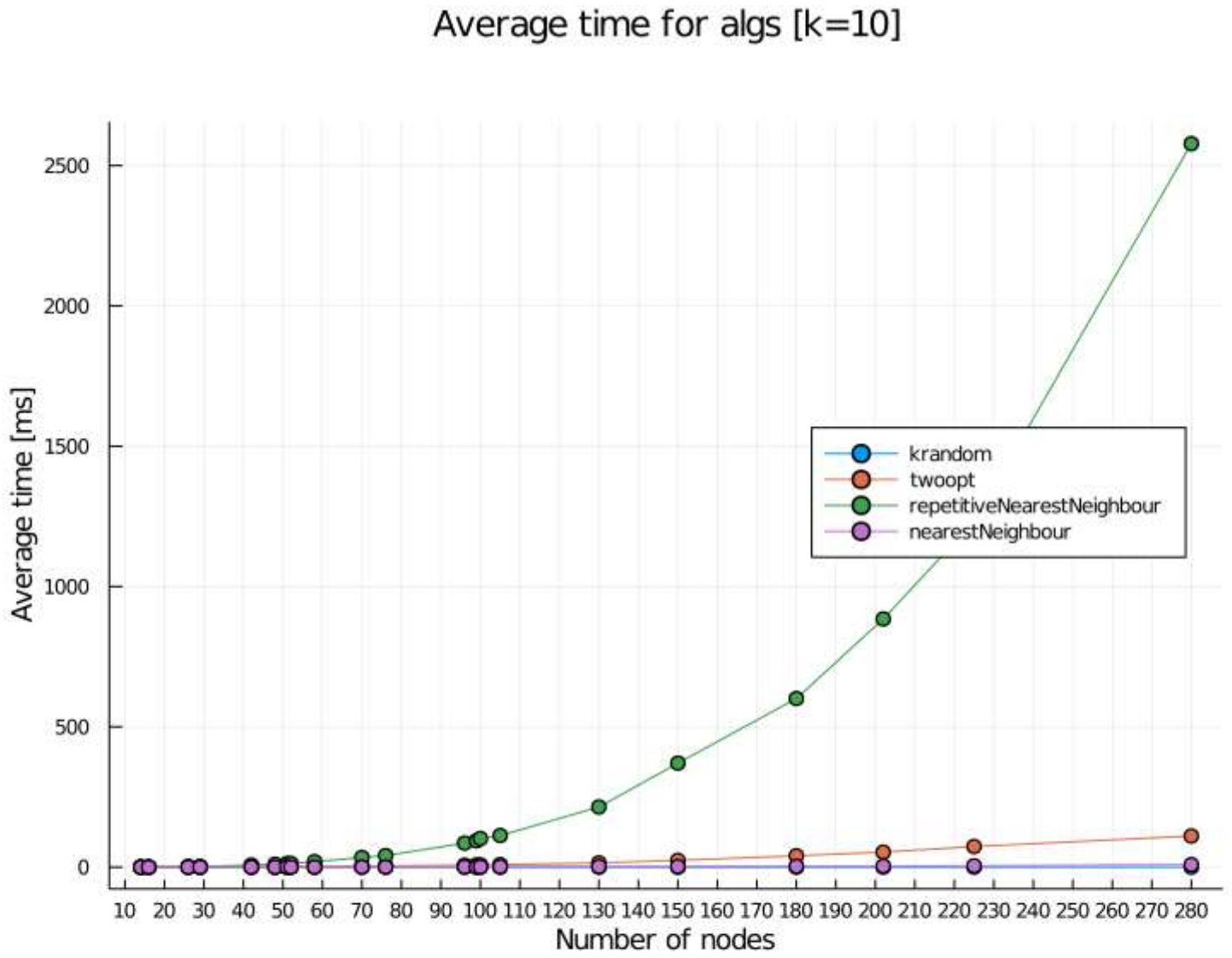
- przypadek asymetryczny:
  - wszystkie heurystyki:


Average time for algs [k=10]
Average time [ms]
3000
2000
1000
0
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300
Number of nodes
krandom
twoopt
repetitiveNearestNeighbour
nearestNeighbour

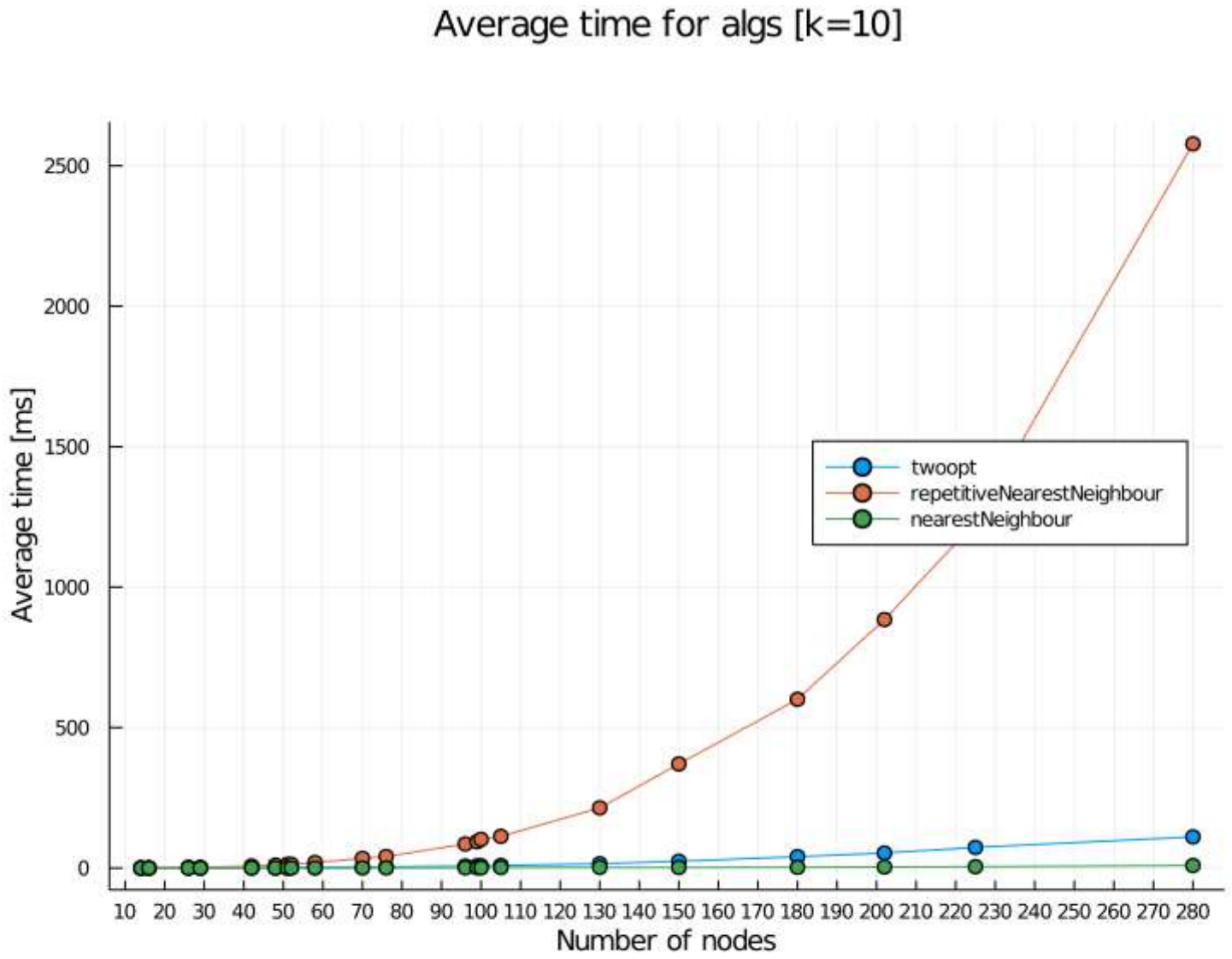
- bez k-random:



- problemy TSPLIB:
  - wszystkie heurystyki:


Average time for algs [k=10]
Average time [ms]
2500
2000
1500
1000
500
0
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280
Number of nodes
krandom
twoopt
repetitiveNearestNeighbour
nearestNeighbour

- bez k-random:



Dla badanych heurystyk oszacowaliśmy ich złożoności obliczeniowe:

- k-random O(kn), gdzie k to ilosc losowo generowanych permutacji, a n to wielkość danych wejściowych (węzłów).
- metoda najbliższego sąsiada O(n^2), gdyż dla każdego węzła musimy sprawdzić do n-1 innych węzłów.

- metoda rozszerzonego najbliższego sąsiada O(n^3), gdyż przechodzimy przez metodę najbliższego sąsiada tyle razy ile mamy węzłów, czyli n razy.
- algorytm 2-opt O(n^3), liczenie drogi (koszt O(n)) dla każdego możliwego przestawienia dwóch par węzłów z wejściowej drogi.