Euklidesowy problem komiwojażera

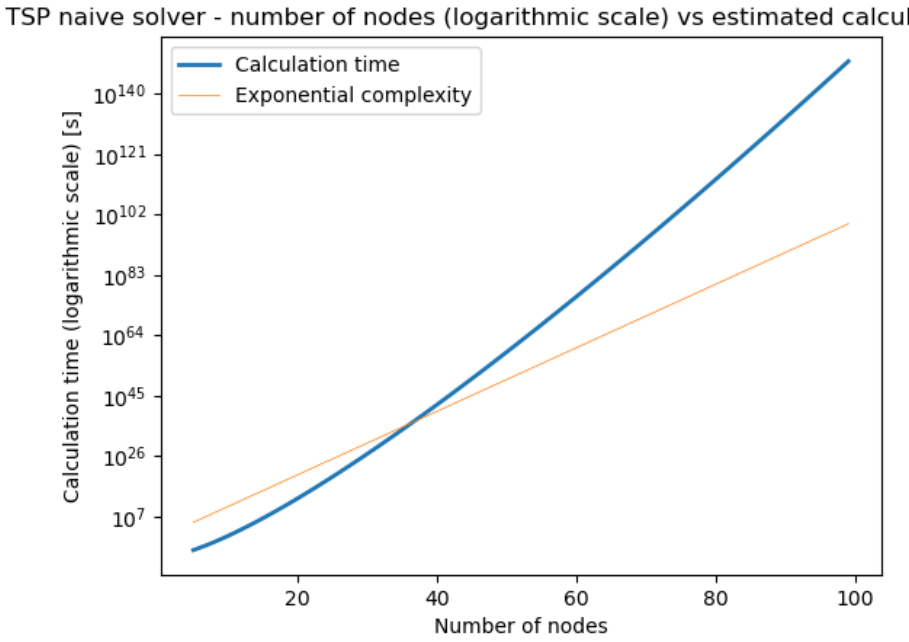
Techniki algorytmiczne – projekt zaliczeniowy

Jakub Popowski

WCY23IB2N2

Rozwiązanie dokładne

- Problem komiwojażera – *znalezienie optymalnej trasy pomiędzy **n** miastami.*
- Wersja **symetryczna** i niesymetryczna – *czy trasa z punktu A do punktu B ma taki sam „koszt” jak trasa z punktu B do punktu A.*
- Rozwiązanie dokładne polega na obliczeniu kosztu wszystkich możliwych tras, których jest **n!** – *liczba permutacji zbioru n-elementowego*.

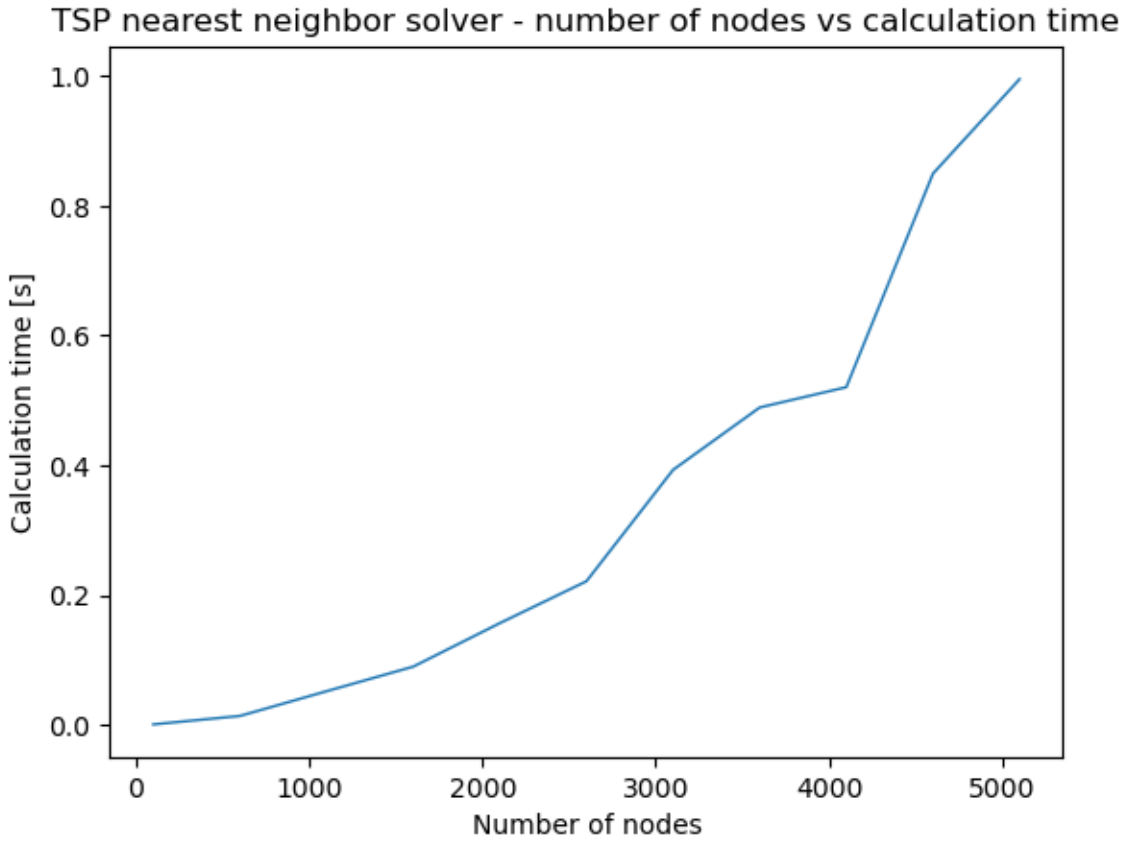


	Number of nodes	Estimated calculation time
0	5	0.01593425004810005 ms
1	6	0.09560550028860029 ms
2	7	0.669238502020202 ms
3	8	5.353908016161616 ms
4	9	48.18517214545455 ms
5	10	4.818517214545455 s
6	11	53.00368936 s
7	12	10.600737872 m
8	13	2.2968265389333333 hr
9	14	1.3398154810444445 days
10	15	20.097232215666665 days
11	16	321.55571545066664 days
12	17	15.184575451837038 years
13	18	273.32235813306664 years
14	19	5193.124804528267 years
15	20	103862.49609056534 years
16	21	2181112.417901872 years
17	22	47984473.19384118 years
18	23	1103642883.4583473 years
19	24	26487429203.00033 years

Rozwiązanie heurystyczne – algorytm najbliższego sąsiada

- Algorytm najbliższego sąsiada – *będąc w danym mieście podróżuj do najbliższego jeszcze nieodwiedzonego miasta.*
- Algorytm bardzo prosty w implementacji i interpretacji.
- Złożoność obliczeniowa $\boldsymbol{O(n^2)}$ - *będąc w i-tym mieście należy porównać* $n - i$ *miast*, tj.

$$\sum_{i=1}^{n}(n-i) = n * \frac{n-1}{2} = \frac{n^2-n}{2} = O(n^2)$$



Rozwiązanie heurystyczne – algorytm genetyczny

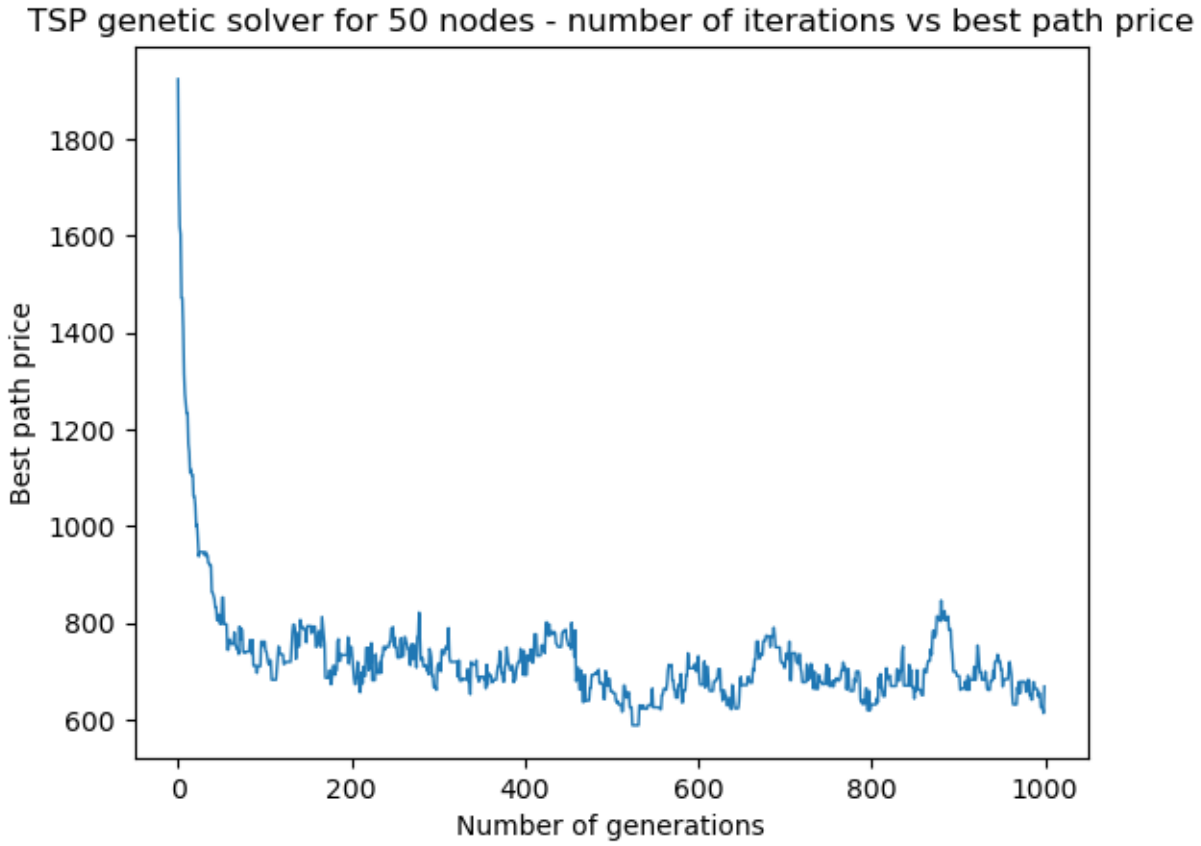
- Algorytm genetyczny – sposób wyszukiwania najlepszego rozwiązania problemu wzorowany na ewolucji biologicznej.
- Populacja – losowo wybrany zbiór rozwiązań problemu, zbiór tras.
- Osobnik – pojedynczy element populacji, trasa.
- Genotyp – informacja definiująca osobnika, kolejność miast na trasie (w przypadku TSP genotyp to także fenotyp).
- Gen – poszczególne miasto na trasie, składa się na genotyp (dla TSP unikalny w genotypie).
- Funkcja przystosowania – funkcja obliczająca „koszt” danej trasy.
- Generacja – populacja w kolejnej iteracji algorytmu.

Działanie algorytmu:

1. Zainicjuj populację,
2. Dokonaj selekcji osobników,
3. Dokonaj reprodukcji osobników,
   a) Krzyżowanie,
   b) Mutacja,
4. Nowa generacja,
5. Jeżeli STOP to zwróć najlepiej przystosowanego osobnika z aktualnej generacji, w przeciwnym przypadku wróć do 2.

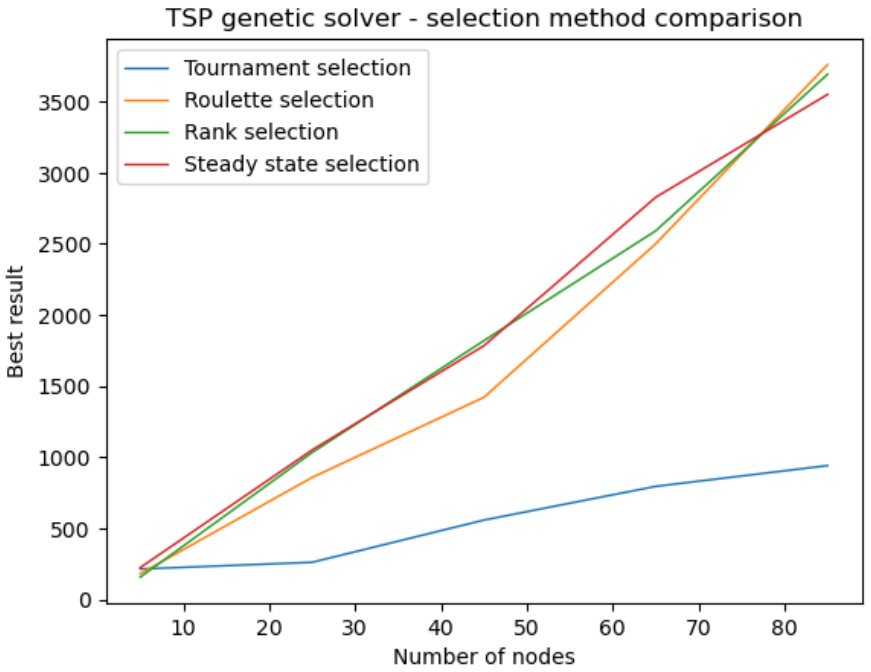
Rozwiązanie heurystyczne – algorytm genetyczny

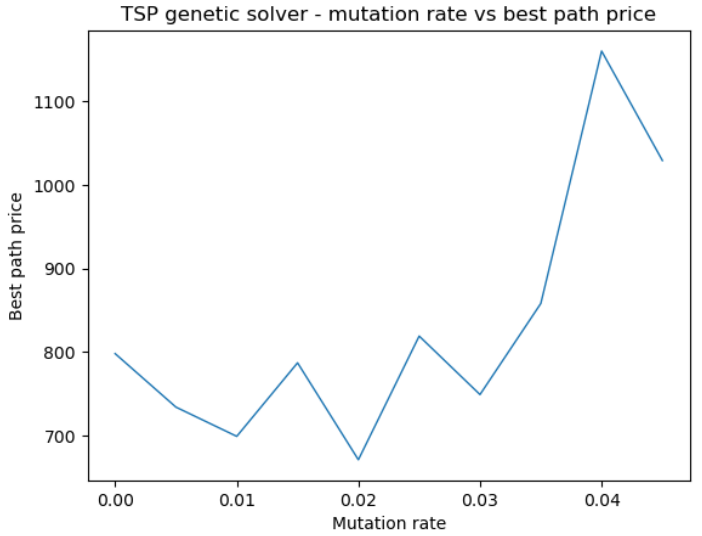
- Inicjalizacja populacji losowa.
- Metody selekcji osobników:
  - *tournament selection,*
  - *roulette wheel selection,*
  - *rank selection,*
  - *steady state selection,*
  - i inne.
- Krzyżowanie – tworzenie genotypu dziecka przez losowy dobór genów rodzica (dla TSP konieczne zachowanie unikalności genów).
- Mutacja – losowe zmiany w genotypie nowo powstałego osobnika.
- Warunek STOP – liczba generacji (iteracji) lub osiągnięcie danego wyniku (potencjalna nieskończona pętla).

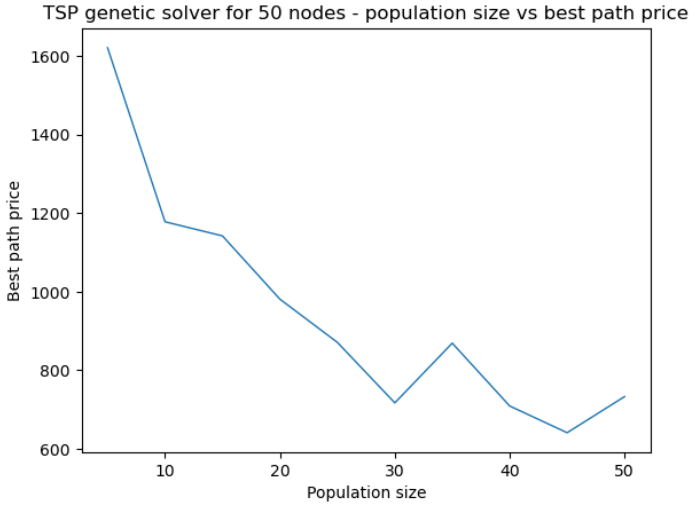


Rozwiązanie heurystyczne – algorytm genetyczny

- Parametryzacja algorytmu m.in przez:
  - rozmiar populacji,
  - metodę selekcji,
    - ilość osobników w turnieju dla selekcji turniejowej,
  - współczynnik mutacji,
  - liczbę generacji.

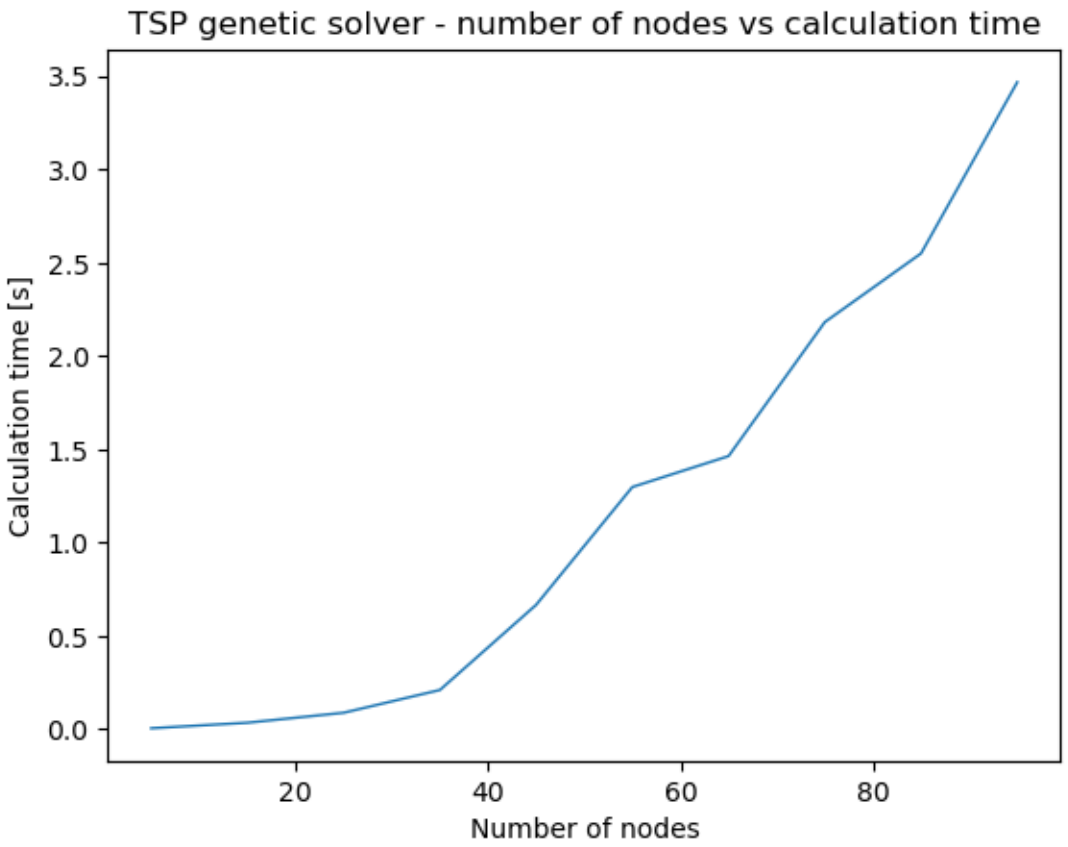






Rozwiązanie heurystyczne – algorytm genetyczny

- Trudna do wyznaczenia złożoność obliczeniowa, zależy od:
  - długości genotypu (liczby miast) n,
  - liczby generacji g,
  - rozmiaru populacji p.
- Złożoność obliczeniową można zapisać jako $g * p * [O(F) + O(C) + O(M)]$, gdzie:
  - F – złożoność obliczeniowa funkcji przystosowania,
  - C – złożoność obliczeniowa funkcji krzyżującej,
  - M – złożoność obliczeniowa funkcji mutującej,
  - W omawianej implementacji wszystkie powyższe funkcję mają złożoność nie większą niż $O(n)$.
- Dla stałego g i p złożoność obliczeniowa może uprościć się do liniowej.
- W omawianej implementacji domyślnie $\mathrm{p} = 0.8{*}\mathrm{n}$ oraz $\mathrm{g} = const$ więc złożoność obliczeniowa algorytmu wynosi $\boldsymbol{O(n^2)}$.



Porównanie algorytmów

- dla małych n – od 5 do 11,
- dane randomizowane,
- wykorzystywany algorytm dokładny.

	Num of nodes	Naive solver score	Nearest-n score	Genetic alg score	Naive solver time	Nearest-n time	Genetic alg time
0	5	223	228	228	0 ns	0 ns	0 ns
1	6	187	202	200	0.10042 ms	0 ns	0 ns
2	7	204	333	333	0.65903 ms	0 ns	0 ns
3	8	153	227	195	5.3575 ms	0 ns	0 ns
4	9	94	94	108	50.05089 ms	0 ns	0 ns
5	10	165	223	191	5.4979791 s	0 ns	0 ns
6	11	179	211	179	1.0957079216666668 m	0 ns	0 ns

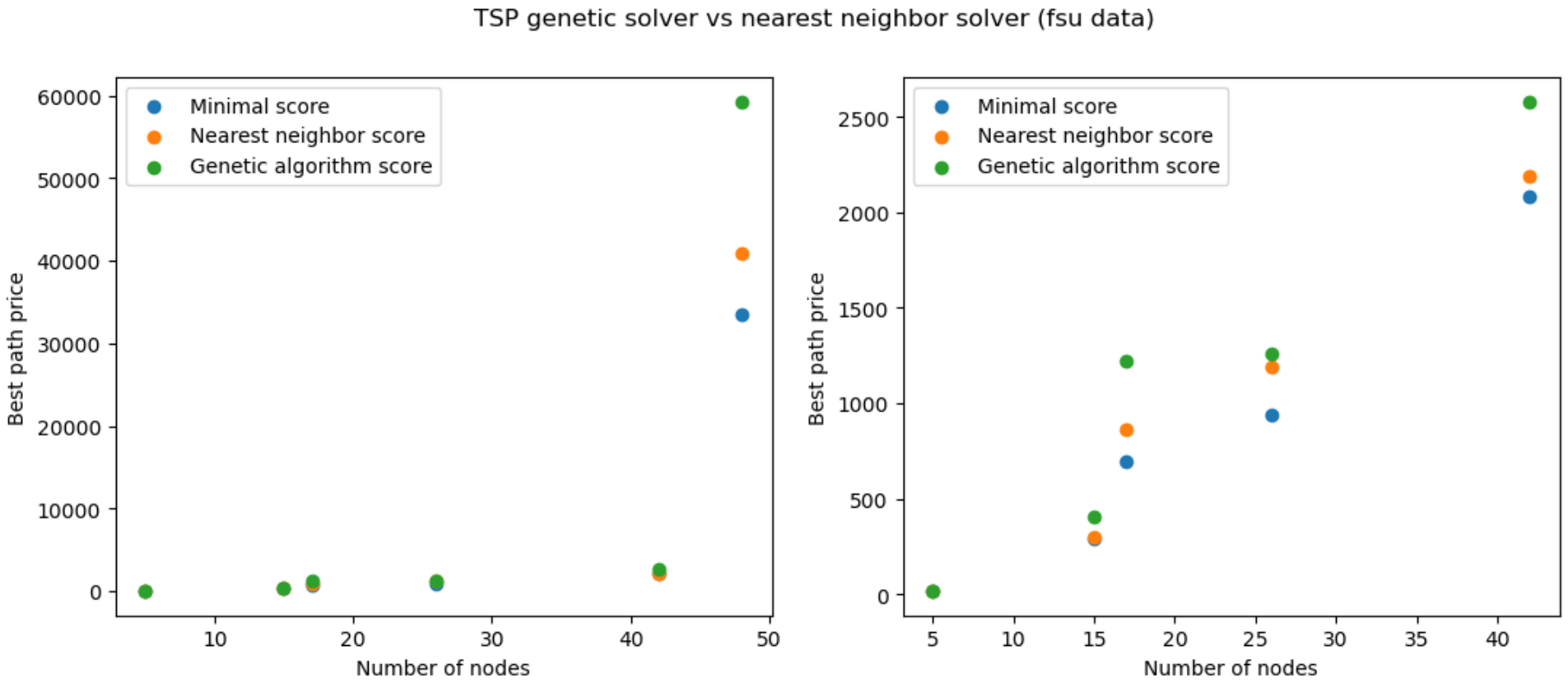
Porównanie algorytmów

- dla małych i średnich n – od 5 do 48,
- dane pobrane ze źródeł *Florida State University,*
- minimalna trasa już wyznaczona – algorytm dokładny niewykorzystywany.

	Dataset name	Num of nodes	Min score	Nearest-n score	Genetic score	Nearest-n time	Genetic time
0	five_d	5	19	21.0	19.0	0 ns	0.75416 ms
1	att48_d	48	33523	40924.0	59224.0	0 ns	76.65235 ms
2	dantzig42_d	42	699	864.0	1260.0	0 ns	33.48603 ms
3	fri26_d	26	937	1188.0	1218.0	0 ns	9.77268 ms
4	gr17_d	17	2085	2187.0	2578.0	0 ns	3.05038 ms
5	p01_d	15	291	299.0	404.0	0 ns	3.30981 ms

Porównanie algorytmów

- dla małych i średnich n – od 5 do 48,
- dane pobrane ze źródeł *Florida State University,*
- minimalna trasa już wyznaczona – algorytm dokładny niewykorzystywany.



Porównanie algorytmów

- dla małych i średnich n – od 5 do 100,
- dane randomizowane,
- algorytm dokładny niewykorzystywany,
- minimalna trasa nie wyznaczana,
- porównanie wyników algorytmu najbliższego sąsiada oraz algorytmu genetycznego względem siebie.