# Algorytmy Metaheurystyczne
# Komiwojażer Heurystycznie

Gabriel Budziński(254609)
Franciszek Stepek (256310)

## Przedmowa

Na samym początku omówimy po krótce użyte algorytmy, oraz zastanowimy się nad ich złożonością obliczeniową, natomiast dalej dopiero przejdziemy do opisu eksperymentów.

## 1 Podsumowanie złożoności obliczeniowych implementacji

## 2 Opis eksperymentów

### 2.1 Implementacja

Algorytmy implementujemy w języku `C/C++`, odległości między wierzchołkami są przechowywane jako pełne tablice dwuwymiarowe typu `int`, a trasy są w kontenerach `vector`, co ułatwia operacje odwracania i mieszania.
Korzystaliśmy z kompilatora g++ wraz z użyciem flag -lSDL2 (używanej przy wizualizacji, wraz z odpowiednim dla danego systemu operacyjnego podlinkowania do folderu zawierającego) oraz -lpthread (przy korzystaniu z wielowątkowości)

### 2.2 Sprzęt

Programy były testowane na dwóch maszynach, laptopie *Lenovo* i komputerze stacjonarnym. Obie jednostki są wyposażone w procesor architektury `x86` marki `intel` oraz 16GB pamięci RAM.

#### 2.2.1 Pececik

Komputer stacjonarny posiada procesor sześciordzeniowy i5-10600K 4,1 GHz (o obniżonym napięciu operacyjnym).

#### 2.2.2 Lapek

Laptop posiada procesor czterordzeniowy i7-6700HQ 2,6 GHz

### 2.3 Instancje

#### 2.3.1 Przykłady TSPLIB

W części eksperymentów użyto instancji euklidejskiego problemu komiwojażera.

#### 2.3.2 Instancje losowe

W celu zwiększenia liczności i dokładności testów spreparowano losowo generowane instancje eukidejskiego problemu komiwojażera.

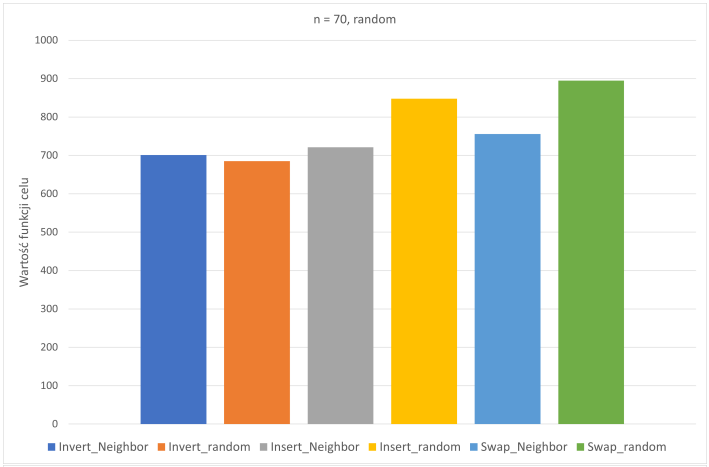
### 2.4 Metodologia/cel

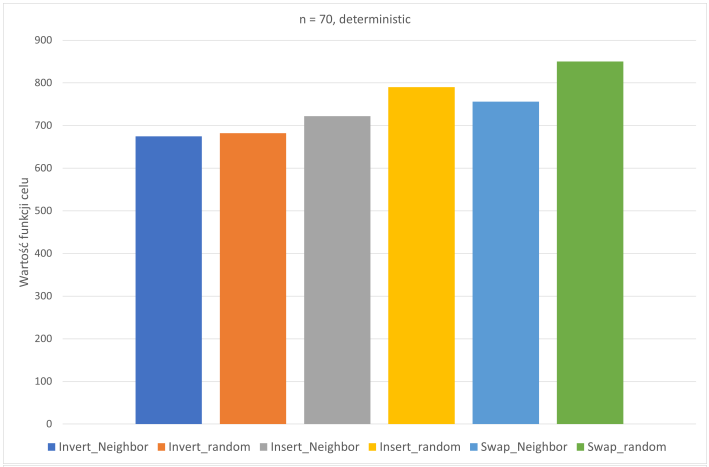
Testy przeprowadzono za pomocą zaimplementowanych w tym celu funkcji ku jak największej automatyzacji. Dane o przeprowadzonych testach zapisywano do plików tekstowych w formacie CSV, a następnie poddane analizie. Testowanie miało na celu wskazanie mocnych i słabych stron zaimplementowanych heurystyk, jak i ich porównanie.

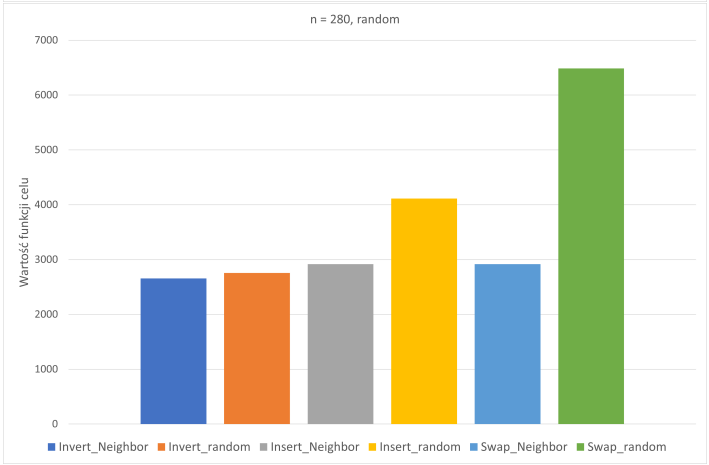
## 2.5 Opis wyników

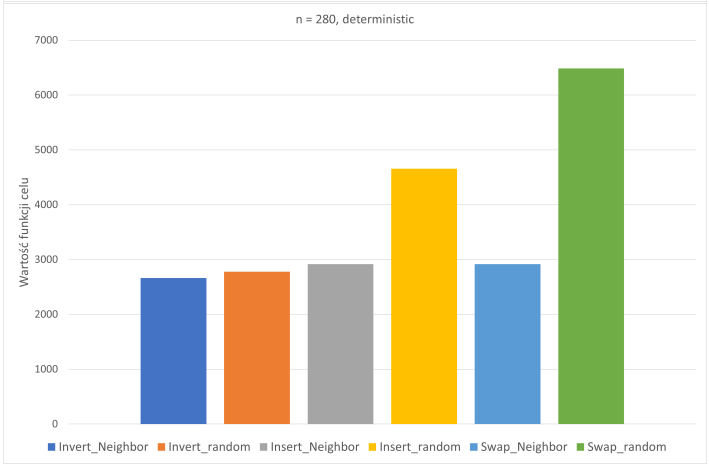
## 2.6 Porównanie wariantów dla optymalnych parametrów

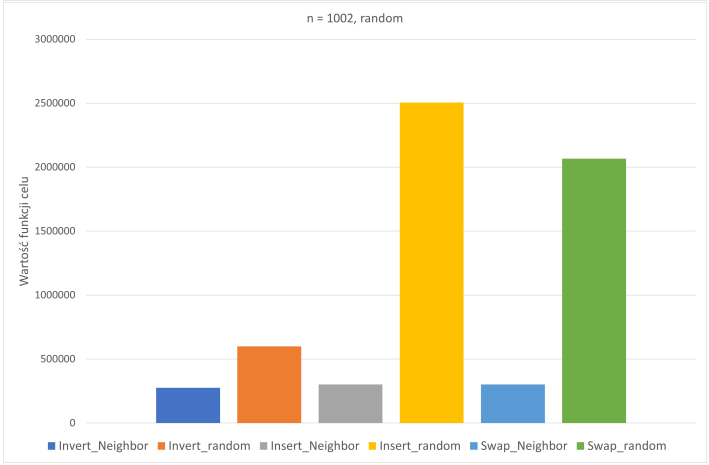
Po przetestowaniu optymalnych parametrów przeprowadzono testy dla ustalonych instancji TSPLIB. Wykresy podzielono na dwie grupy, po lewej operacja `kick` była losowa w zadanym przedziale, po prawej deterministyczna.

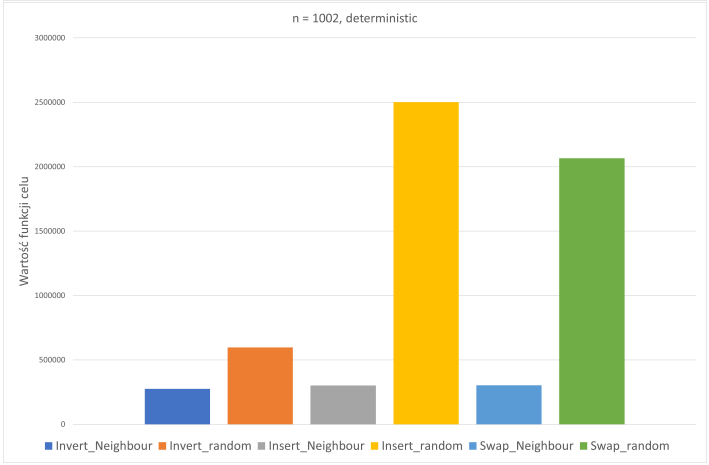


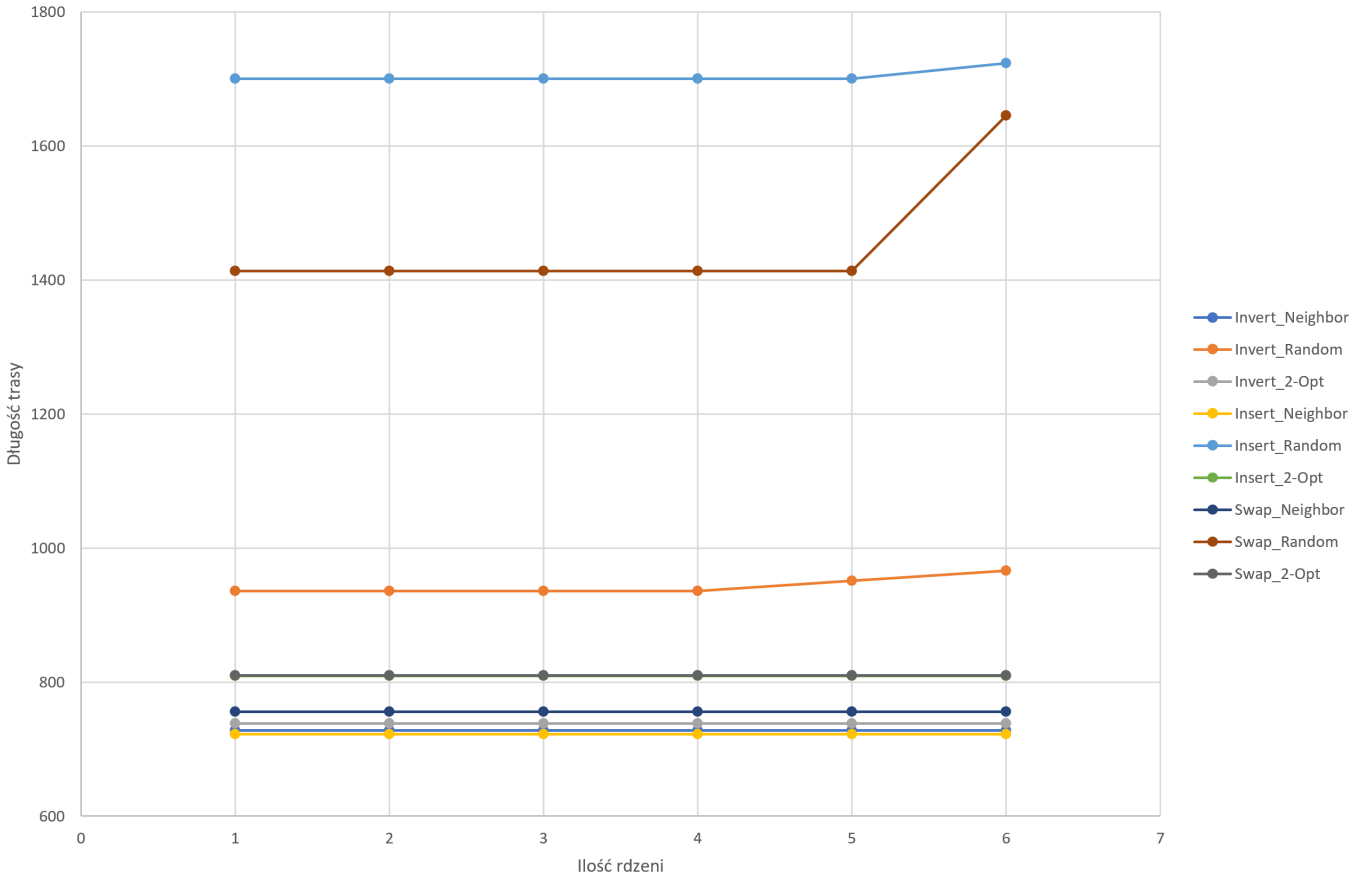

W obydwu wariantach operacji `kick` można zauważyć zbliżone wyniki przy wszystkich wielkościach problemu. Co więcej, wraz ze zwiększaniem się problemu bardziej klarownie widać różnice między wersjami.

- Początkowa trasa na podstawie Nearest Neighbour daje lepsze wyniki niż k-random
- Wszystkie otoczenia są porównywalne, najlepszy Invert, następnie Insert, najgorszy Swap

### 2.6.1 Algorytmy uwspółbieżnione

Uwspółbieżnienie zaimplementowano na zasadzie równoległego uruchamiania kilku instancji algorytmu `TABU-Search` jednocześnie. Z oczywistych względów nie testowano zachowania deterministycznej wersji algorytmu. Niestety, ta metoda

uwspółbieżnienia nie dała dobrych rezultatów:




Dla każdego otoczenia i każdego rodzaju trasy startowej nie uzyskano w ten sposób polepszenia trasy.

### 2.7 Wnioski

## Drobne uwagi