

# Techniki Optymalizacji

## Laboratorium nr 1

### Sprawozdanie

Paulina Sadowska, Rafał Araszkiewicz

10 października 2016

## 1. Wprowadzenie

Celem ćwiczenia było zaimplementowanie algorytmów rozwiązujących problem Komiwojażera dla zbioru 100 punktów. Algorytmy te znaleźć miały najbardziej optymalną ścieżkę łączącą 50 dowolnych punktów grafu gdzie punktem startowym miał być każdy z punktów znajdujących się w zbiorze. Dane testowe zawarte są na stronie Uniwersytetu Heidelberg.

## 2. Nearest Neighbour

W algorytmie tym trasa budowana jest w taki sposób że w każdym punkcie szukamy trasy o najmniejszym koszcie (odległości) i punkt do którego ta trasa prowadzi dodajemy do ścieżki i dla niego szukamy kolejnego najbliższego punktu w grafie.

### 2.1. Implementacja w pseudokodzie

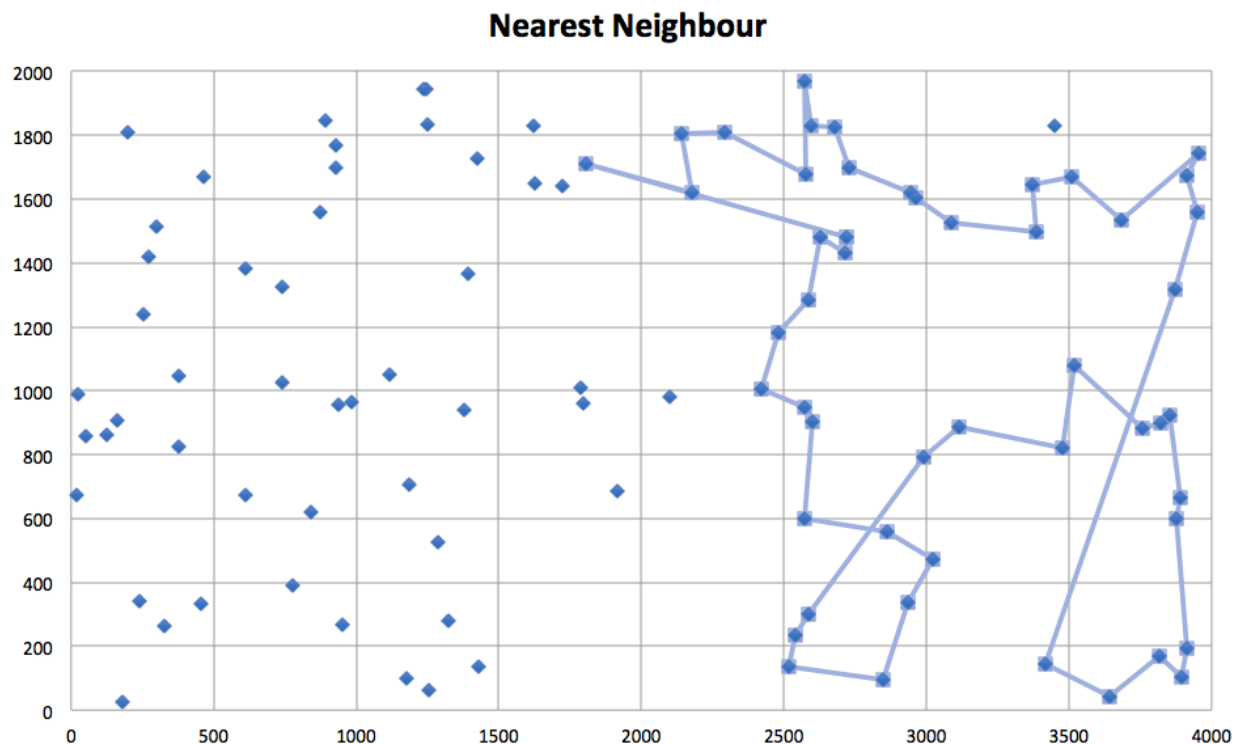
```
dla każdego z zdefiniowanych punktów początkowych
    dodaj do ścieżki punkt początkowy
    wykonaj 49 razy
        dla każdego punktów nie znajdujących się na ścieżce
            koszt = odległość pomiędzy tym punktem a ostatnim
                    ↪ dodanym do ścieżki
            jeżeli koszt < dotychczasowy najmniejszy koszt
                dotychczasowy najmniejszy koszt = koszt
        koniec
    dodaj do ścieżki punkt o najmniejszym koszcie (odległości)
koniec

dodaj na końcu ścieżki punkt początkowy
koniec
```

## 2.2. Wyniki

Tablica 1: Nearest Neighbour - wyniki

min	10298
mean	12793
max	14877
best path	80, 24, 60, 50, 86, 8, 6, 56, 19, 11, 26, 85, 34, 61, 59, 76, 22, 97, 90, 44, 31, 10, 14, 16, 73, 20, 58, 71, 9, 83, 35, 37, 23, 17, 78, 52, 87, 15, 21, 93, 69, 65, 64, 3, 96, 55, 79, 30, 88, 41, 80



Rysunek 1: Najlepsza trasa - Nearest Neighbour

## 3. Greedy Cycle

W algorytmie tym trasa jest budowana w taki sposób, aby zawsze tworzyła cykl Hamiltona. W każdej iteracji dodawany jest jeden najkrótszy łuk z pozostałych dostępnych.

### 3.1. Implementacja w pseudokodzie

```
dla każdego z zdefiniowanych punktów początkowych
    dodaj do ścieżki punkt początkowy
dla każdego z punktów oprócz początkowego
    koszt = odległość pomiędzy tym punktem a początkowym
    jeżeli koszt < dotychczasowy najmniejszy koszt
        dotychczasowy najmniejszy koszt = koszt
koniec
```

```

dodaj do sciezki punkt o najmniejszym koszcie (odleglosci)
dodaj na koncu sciezki punkt poczatkowy

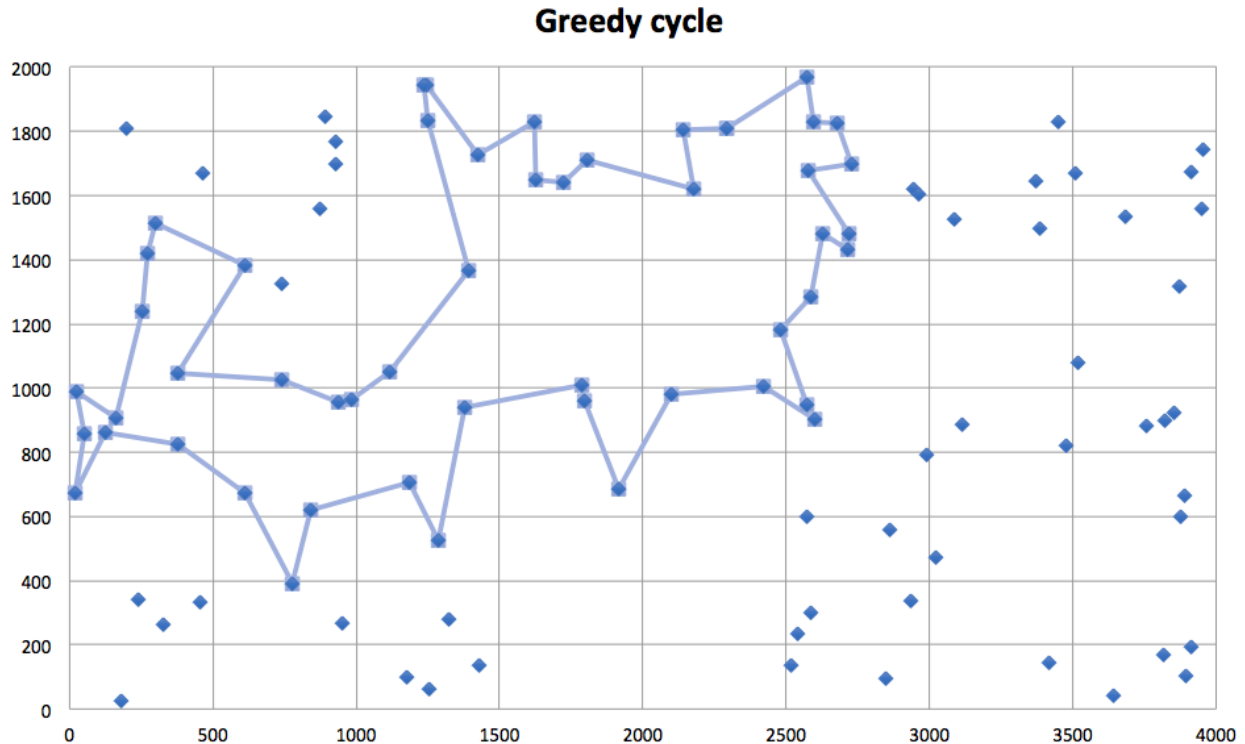
dla pozostalych 48 krokow
    dla wszystkich par punktow w sciezce
        dla wszystkich nieodwiedzonych punktow
            koszt = odleglosc miedzy trojka punktow –
                ↪ odleglosc miedzy para punktow
            jezeli koszt < dotychczasowy najmniejszy koszt
                dotychczasowy najmniejszy koszt =
                    ↪ koszt
        koniec
    koniec
    dodaj punkt powodujacy najmniejsza zmiane kosztu w odpowiednie
        ↪ miejsce w sciezce
koniec
koniec
koniec

```

### 3.2. Wyniki

Tablica 2: Greedy Cycle - wyniki

min	11095
mean	12653
max	13393
best path	61, 34, 85, 26, 11, 19, 6, 8, 56, 86, 50, 24, 80, 60, 57, 66, 27, 92, 0, 7, 91, 74, 96, 18, 52, 15, 69, 21, 93, 87, 17, 23, 37, 83, 78, 89, 48, 5, 62, 46, 10, 16, 14, 31, 44, 90, 97, 22, 76, 59, 61



Rysunek 2: Najlepsza trasa - Greedy Cycle

## 4. Nearest Neighbour Grasp

Modyfikacja algorytmu Nearest Neighbour opisanego w punkcie 2 polegająca na tym, że w każdym kroku szukamy 3 najlepszych rozwiązań i z nich losujemy te, które będzie dodane do ścieżki.

### 4.1. Implementacja w pseudokodzie

Analogiczna do zawartej w podpunkcie 2.1 z tą różnicą że zamiast szukać jednego punktu grafu znajdującego się najbliżej aktualnego punktu ścieżki należy znaleźć 3 najlepsze i wybrać losowy z nich.

dla każdego z zdefiniowanych punktów początkowych

    dodaj do ścieżki punkt początkowy

    wykonaj 49 razy

        dla każdego punktu nie znajdującego się na ścieżce

            koszt = odległość pomiędzy tym punktem a ostatnim

            ↪ dodanym do ścieżki

            jeżeli koszt < największy z listy 3 najmniejszych

            ↪ odległości

            dodaj koszt do listy 3 najmniejszych

            ↪ odległości zamiast największego z listy

        koniec

    dodaj do ścieżki losowy z 3 punktów o najmniejszym koszcie (

        ↪ odległości)

koniec

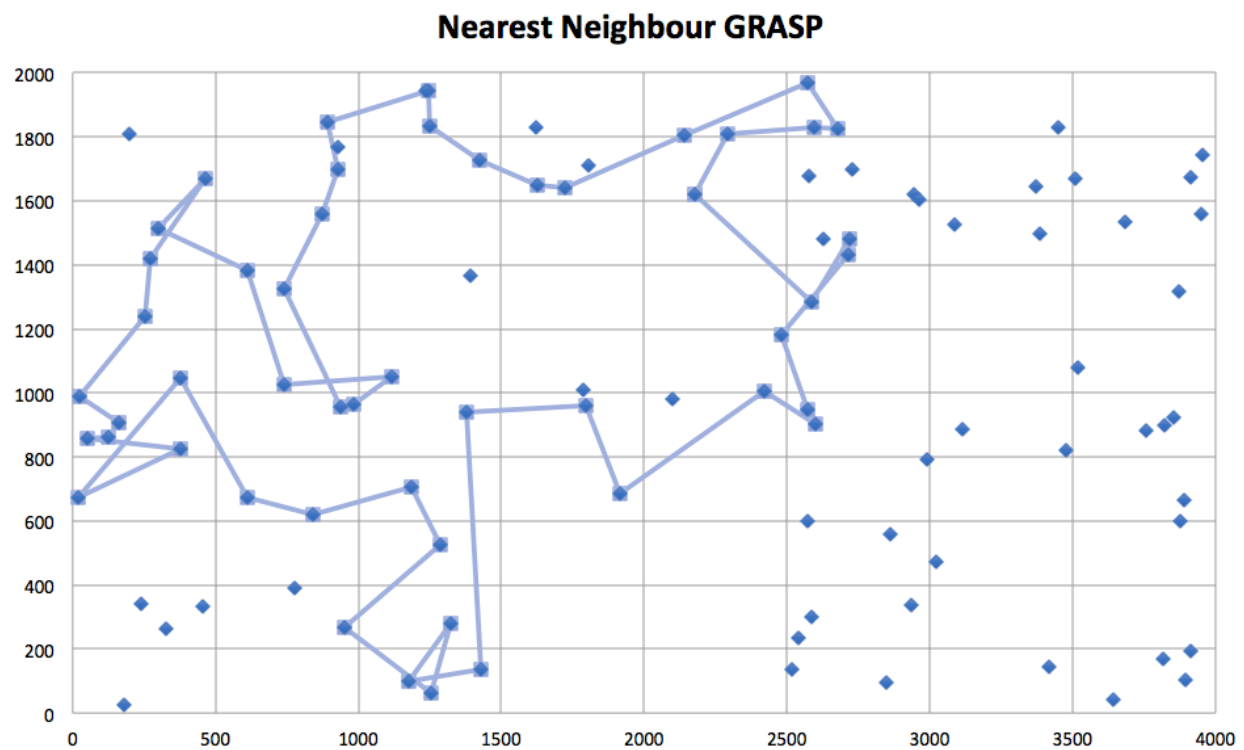
dodaj na koncu sciezki punkt poczatkowy

koniec

## 4.2. Wyniki

Tablica 3: Nearest Neighbour Grasp - wyniki

min	14103
mean	14146
max	21413
best path	76 61 85 26 34 59 97 90 31 10 14 58 20 71 9 48 5 62 89 83 37 35 23 17 93 87 15 21 52 69 78 18 74 91 7 55 30 41 79 88 0 27 66 60 80 24 50 8 6 86 76



Rysunek 3: Najlepsza trasa - Nearest Neighbour Grasp

## 5. Greedy Cycle Grasp

Modyfikacja algorytmu Greedy Cycle opisanego w punkcie 3 polegająca na tym, że w każdym kroku szukamy 3 najlepszych rozwiązań i z nich losujemy te, które będzie dodane do ścieżki.

## 5.1. Implementacja w pseudokodzie

Analogiczna do zawartej w podpunkcie 3.1 z tą różnicą że zamiast szukać jednej ścieżki która da najmniejszą różnicę kosztów należy znaleźć 3 najlepsze trasy i wybrać losową z nich.

```
dla kazdego z zdefiniowanych punktow poczatkowych
    dodaj do sciezki punkt poczatkowy
    dla kazdego z punktow oprocz poczatkowego
        koszt = odleglosc pomiedzy tym punktem a ostatnim dodanym do
            ↪ sciezki
        jezeli koszt < najwiekszy z listy 3 najmniejszych
            ↪ odleglosci
            dodaj koszt do listy 3 najmniejszych
            ↪ odleglosci zamiast najwiekszego z listy

    koniec

dodaj do sciezki losowy z 3 punktow o najmniejszym koszcie (odleglosci
    ↪ )
dodaj na koncu sciezki punkt poczatkowy

dla pozostalych 48 krokow
    dla wszystkich par punktow w sciezce
        dla wszystkich nieodwiedzonych punktow
            koszt = odleglosc miedzy trojka punktow –
                ↪ odleglosc miedzy para punktow
            jezeli koszt < najwiekszy z listy 3
                ↪ najmniejszych odleglosci
                dodaj koszt do listy 3 najmniejszych
                ↪ odleglosci zamiast najwiekszego
                ↪ z listy

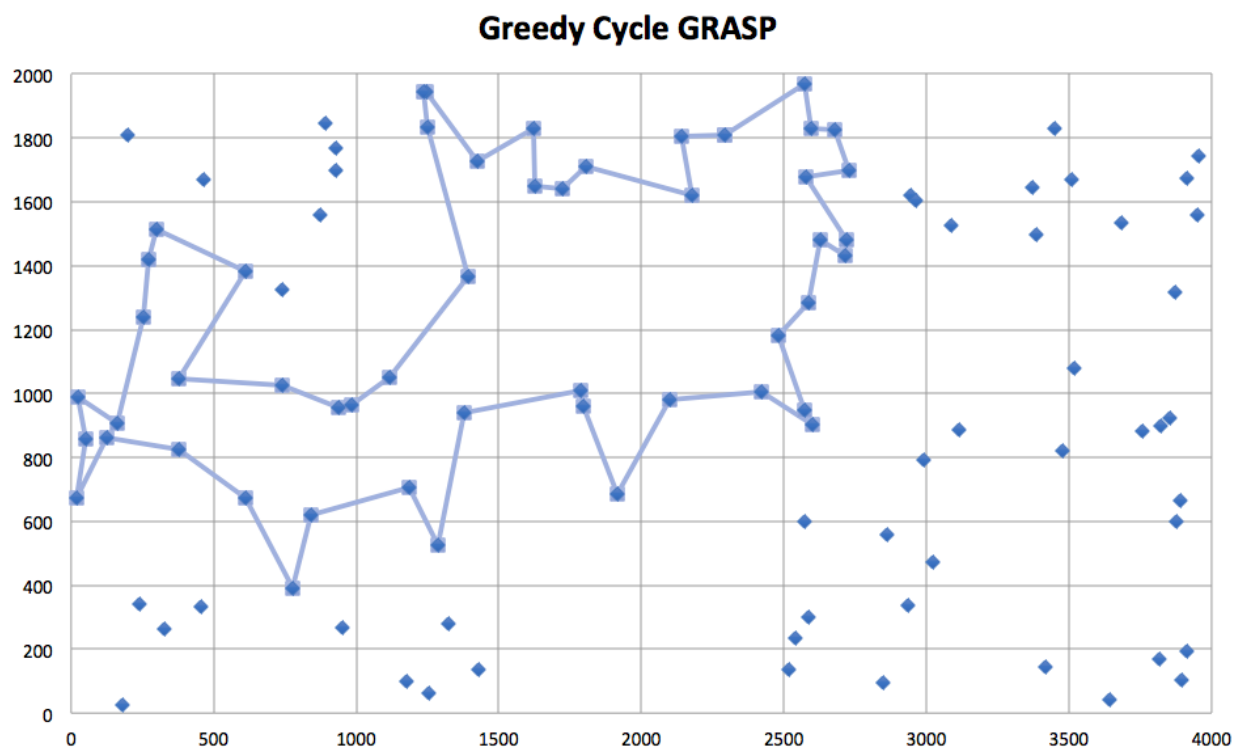
        koniec
    koniec
dodaj losowy punkt z 3 wybranych powodujacych najmniejsza
    ↪ zmiane kosztu w odpowiednie miejsce w sciezce

koniec
koniec
```

## 5.2. Wyniki

Tablica 4: Greedy Cycle Grasp - wyniki

min	11095
mean	13757
max	13504
best path	61 34 85 26 11 19 6 8 56 86 50 24 80 60 57 66 27 92 0 7 91 74 96 18 52 15 69 21 93 87 17 23 37 83 78 89 48 5 62 46 10 16 14 31 44 90 97 22 76 59 61



Rysunek 4: Najlepsza trasa - Greedy Cycle Grasp