Problem komiwojażera

Adrian Stępień i Wojciech Młyńczak

10 maja 2020

1 Zadanie 4

1.1 Opis zadania

Zadanie polega na implementacji trzech metod - multiple start local search, iterated local search, iterated local search.

1.2 Opis zaimplementowanych algorytmów

1.2.1 Multiple Start Local Search

```
Wygeneruj losowe rozwiązanie.
Powtarzaj:
    Wygeneruj listę możliwych ruchów:
        Zainicjalizuj pustą listę zwracanych ruchów.
        Wygeneruj punkty poza rozwiązaniem.
        Dla każdego ruchu:
            Jeżeli typ ruchu to zamiana punktów:
                Dla każdego punktu w rozwiązaniu:
                    Dla każdego punktu poza rozwiązaniem:
                        Oblicz deltę.
                        Dodaj ruch do listy zwracanych
                            ruchów.
            W przeciwnym wypadku dla zamiany krawędzi:
                Dla każdego punktu w rozwiązaniu:
                    Dla każdego punktu w rozwiązaniu poza
                        punktem wybranym w otaczającej pętli:
                        Oblicz deltę.
                        Dodaj ruch do listy zwracanych
                            ruchów.
        Posortuj listę zwracanych ruchów według obliczonej
            delty.
    Wykonaj najlepszy ruch:
        Jeżeli lista ruchów nie jest pusta:
            Jeżeli delta pierwszego ruchu z listy jest
                mniejsza od zera:
                Jeżeli typ ruchu to zamiana punktów, to
                    zamień punkty ze sobą.
                W przeciwnym wypadku dla zamiany krawędzi
                    zamień ze sobą krawędzie.
    Jeżeli nie można wykonać najlepszego ruchu to przerwij
        pętlę.
```

```
    Jeżeli rozwiązanie jest puste albo jest lepsze od
        aktualnego to wybierz znalezione rozwiązanie z
        najlepszym ruchem.
```

1.2.2 Iterated Local Search 1

```
Wygeneruj losowe rozwiązanie.
Wykonaj lokalne przeszukiwanie.
Powtarzaj dopóki nie osiągnięto określonego czasu:
    Wykonaj perturbację:
        Wylosuj 10 punktów do zamiany.
        Zamień punkty z rozwiązania z wylosowanymi punktami.
    Powtarzaj:
        Wygeneruj listę możliwych ruchów:
            Zainicjalizuj pustą listę zwracanych ruchów.
            Wygeneruj punkty poza rozwiązaniem.
            Dla każdego ruchu:
                Jeżeli typ ruchu to zamiana punktów:
                    Dla każdego punktu w rozwiązaniu:
                        Dla każdego punktu poza
                            rozwiązaniem:
                            Oblicz deltę.
                            Dodaj ruch do listy zwracanych
                                ruchów.
                W przeciwnym wypadku dla zamiany krawędzi:
                    Dla każdego punktu w rozwiązaniu:
                        Dla każdego punktu w rozwiązaniu
                            poza punktem wybranym w
                            otaczającej pętli:
                            Oblicz deltę.
                            Dodaj ruch do listy zwracanych
                                ruchów.
            Posortuj listę zwracanych ruchów według
                obliczonej delty.
        Wykonaj najlepszy ruch:
            Jeżeli lista ruchów nie jest pusta:
                Jeżeli delta pierwszego ruchu z listy jest
                    mniejsza od zera:
                    Jeżeli typ ruchu to zamiana punktów, to
                        zamień punkty ze sobą.
                    W przeciwnym wypadku dla zamiany
                        krawędzi zamień ze sobą krawędzie.
        Jeżeli nie można wykonać najlepszego ruchu to
            przerwij pętlę.
    Jeżeli rozwiązanie po wykonaniu perturbacji jest lepsze
        od aktualnego rozwiązania, to zapisz znalezione
        rozwiązanie jako aktualne rozwiązanie.
```

1.2.3 Iterated Local Search 2

```
Wygeneruj losowe rozwiązanie.
```

```
Wykonaj lokalne przeszukiwanie.
Powtarzaj dopóki nie osiągnięto określonego czasu:
    Wykonaj perturbację:
        Usuń 20% punktów z rozwiązania.
        Zamień punkty z rozwiązania z wylosowanymi punktami.
        Napraw rozwiązanie za pomocą algorytmu Greedy Cycle.
    Powtarzaj:
        Wygeneruj listę możliwych ruchów:
            Zainicjalizuj pustą listę zwracanych ruchów.
            Wygeneruj punkty poza rozwiązaniem.
            Dla każdego ruchu:
                Jeżeli typ ruchu to zamiana punktów:
                    Dla każdego punktu w rozwiązaniu:
                        Dla każdego punktu poza
                            rozwiązaniem:
                            Oblicz deltę.
                            Dodaj ruch do listy zwracanych
                                ruchów.
                W przeciwnym wypadku dla zamiany krawędzi:
                    Dla każdego punktu w rozwiązaniu:
                        Dla każdego punktu w rozwiązaniu
                            poza punktem wybranym w
                            otaczającej pętli:
                            Oblicz deltę.
                            Dodaj ruch do listy zwracanych
                                ruchów.
            Posortuj listę zwracanych ruchów według
                obliczonej delty.
        Wykonaj najlepszy ruch:
            Jeżeli lista ruchów nie jest pusta:
                Jeżeli delta pierwszego ruchu z listy jest
                    mniejsza od zera:
                    Jeżeli typ ruchu to zamiana punktów, to
                        zamień punkty ze sobą.
                    W przeciwnym wypadku dla zamiany
                        krawędzi zamień ze sobą krawędzie.
        Jeżeli nie można wykonać najlepszego ruchu to
            przerwij pętlę.
    Jeżeli rozwiązanie po wykonaniu perturbacji jest lepsze
        od aktualnego rozwiązania, to zapisz znalezione
        rozwiązanie jako aktualne rozwiązanie.
```

1.3 Wyniki pomiarów

Pomiar	Wartość średnia	Wartość minimalna	Wartość maksymalna	Średni czas [ms]	Minimalny czas [ms]	Maksymalny czas [ms]
Lokalne przeszukiwanie w wersji MSLS dla kroA200	15137.5	14608.0	15476.0	126088	123303	129858
Lokalne przeszukiwanie w wersji MSLS dla kroB200	15339.5	14850.0	15748.0	125760	123343	129725
Lokalne przeszukiwanie w wersji ISL1 dla kroA200	13339.9	12955.0	13567.0	126190	126095	126424
Lokalne przeszukiwanie w wersji ISL1 dla kroB200	13546.5	13288.0	14187.0	125837	125768	126027
Lokalne przeszukiwanie w wersji ISL2 dla kroA200	14329.2	13072.0	16294.0	126211	126126	126308
Lokalne przeszukiwanie w wersji ISL2 dla kroB200	14554.5	13828.0	15429.0	125938	125820	126058

ISL1:

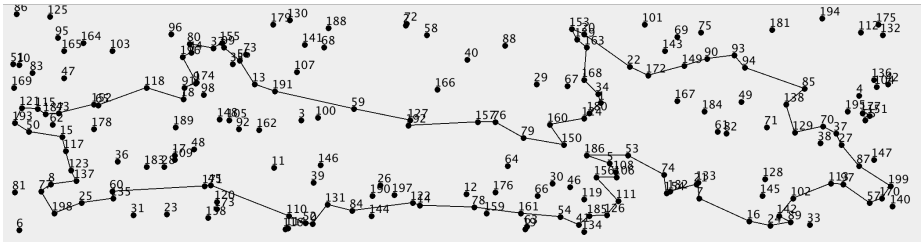
- Average number of LocalSearch Iterations 544.8 for :kroA200.tsp
- Minimum number of LocalSearch Iterations 536 for :kroA200.tsp
- Maximum number of LocalSearch Iterations 561 for :kroA200.tsp
- Average number of LocalSearch Iterations 546.2 for :kroB200.tsp
- Minimum number of LocalSearch Iterations 525 for :kroB200.tsp
- Maximum number of LocalSearch Iterations 566 for :kroB200.tsp

ISL2:

- Average number of LocalSearch Iterations 650.3 for :kroA200.tsp
- Minimum number of LocalSearch Iterations 462 for :kroA200.tsp
- Maximum number of LocalSearch Iterations 991 for :kroA200.tsp
- Average number of LocalSearch Iterations 495.8 for :kroB200.tsp
- Minimum number of LocalSearch Iterations 362 for :kroB200.tsp
- Maximum number of LocalSearch Iterations 621 for :kroB200.tsp

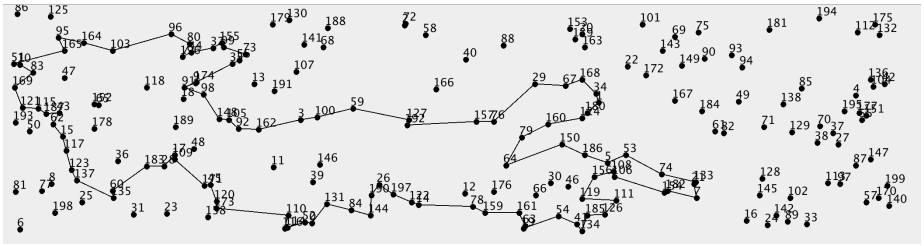
1.4 Wizualizacje najlepszych rozwiązań

1.4.1 Lokalne przeszukiwanie w wersji MSLS dla kroA200



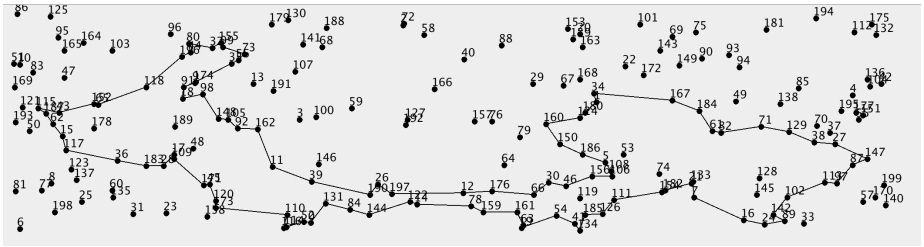
Rysunek 1: Lokalne przeszukiwanie w wersji MSLS dla kroA200

1.4.2 Lokalne przeszukiwanie w wersji ILS1 dla kroA200



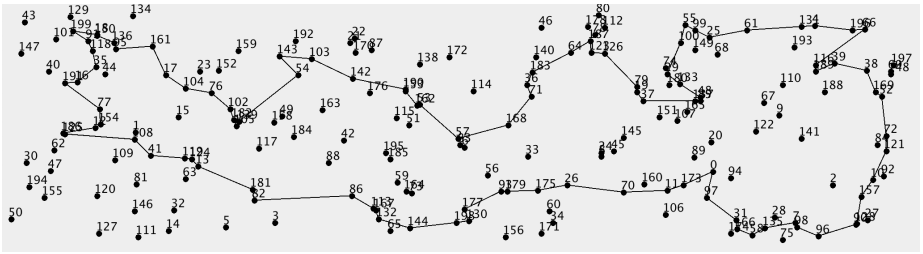
Rysunek 2: Lokalne przeszukiwanie w wersji ILS1 dla kroA200

1.4.3 Lokalne przeszukiwanie w wersji ILS2 dla kroA200



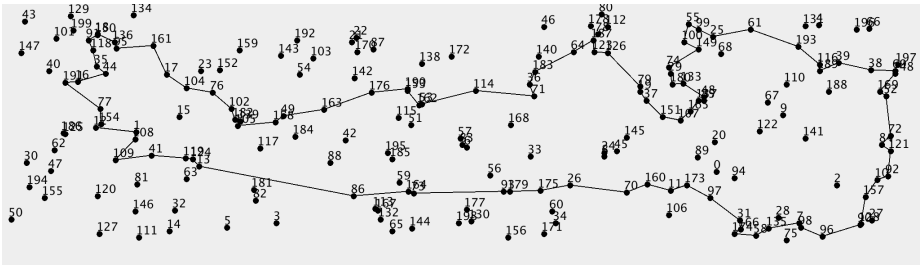
Rysunek 3: Lokalne przeszukiwanie w wersji ILS2 dla kroA200

1.4.4 Lokalne przeszukiwanie w wersji MSLS dla kroB200



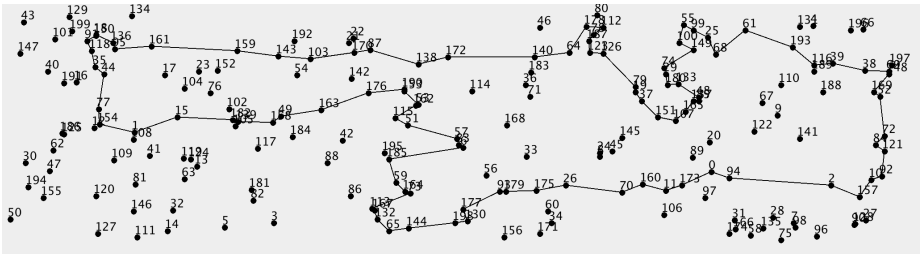
Rysunek 4: Lokalne przeszukiwanie w wersji MSLS dla kroB200

1.4.5 Lokalne przeszukiwanie w wersji ILS1 dla kroA200



Rysunek 5: Lokalne przeszukiwanie w wersji ILS1 dla kroB200

1.4.6 Lokalne przeszukiwanie w wersji ILS2 dla kroB200



Rysunek 6: Lokalne przeszukiwanie w wersji ILS2 dla kroB200

1.5 Wnioski

2 Zadanie 3

2.1 Opis zadania

Zadanie polega na poprawie efektywności czasowej lokalnego przeszukiwania w wersji stromej z ruchem wymiany krawędzi.

2.2 Opis zaimplementowanych algorytmów

2.2.1 Algorytm wykorzystania ocen ruchów z poprzednich iteracji z uporządkowaną listą ruchów

```
1  Wygeneruj losowe rozwiązanie.
2  Wygeneruj listę ruchów.
3  Powtarzaj:
4      Sprawdź listę ruchów:
5          Dla każdego ruchu z listy ruchów:
6              Jeżeli typ ruchu to zamiana punktów:
7                  Weź trzy punkty z rozwiązania.
8                  Jeżeli wszystkie trzy punkty istnieją oraz
                       punkt spoza rozwiązania rzeczywiście
                       jest poza rozwiązaniem:
9                      Jeżeli pierwszy i drugi punkt z object1
                           są sąsiadami oraz drugi i trzeci
                           punkt z object1 są sąsiadami:
10                         Oblicz dystans aktualnego
                               rozwiązania.
11                         Dodaj punkty z aktualnego
                               rozwiązania do listy.
12                         Zamień środkowy punkt z trójki
                               punktów z punktem spoza
                               rozwiązania.
13                         Oblicz dystans rozwiązania po
                               zamianie punktów.
14                         Jeżeli różnica dystansów nie jest
                               równa delcie to wyrzuć błąd.
15                         Zapisz ruch jako wykonany ruch.
16                         Dodaj ruch co listy ruchów do
                               usunięcia.
17                     W przeciwnym wypadku:
18                         Dodaj ruch co listy ruchów do
                               usunięcia.
19                 W przeciwnym wypadku:
20                     Dodaj ruch co listy ruchów do usunięcia.
21             W przeciwnym wypadku dla zamiany krawędzi:
22                 Weź punkty pierwszej krawędzi i punkty
                       drugiej krawędzi.
23                 Jeżeli indeks pierwszego punktu z pierwszej
                       krawędzi jest większy od indeksu
                       drugiego punktu i pierwszy punkt nie
                       jest ostatnim punktem, to zamień te
                       punkty ze sobą.
24                 Jeżeli indeks pierwszego punktu z drugiej
                       krawędzi jest większy od indeksu
                       drugiego punktu i pierwszy punkt nie
                       jest ostatnim punktem, to zamień te
                       punkty ze sobą.
25                 Jeżeli pierwszy ruch z pierwszej pary oraz
                       pierwszy ruch z drugiej pary istnieją:
26                     Jeżeli następny punkt względem
                           pierwszego punktu z pierwszej
```

```
                        krawędzi to drugi punkt z pierwszej
                        krawędzi oraz następny punkt
                        względem pierwszego punktu z drugiej
                        krawędzi to drugi punkt z drugiej
                        krawędzi:
27                      Oblicz dystans aktualnego
                            rozwiązania.
28                      Dodaj punkty z aktualnego
                            rozwiązania do listy.
29                      Zamień krawędzie ze sobą.
30                      Oblicz dystans rozwiązania po
                            zamianie krawędzi.
31                      Jeżeli różnica dystansów nie jest
                            równa delcie to wyrzuć błąd.
32                      Zapisz ruch jako wykonany ruch.
33                      Dodaj ruch co listy ruchów do
                            usunięcia.
34                  W przeciwnym wypadku:
35                      Dodaj ruch co listy ruchów do
                            usunięcia.
36              W przeciwnym wypadku:
37                  Dodaj ruch co listy ruchów do usunięcia.
38      Usuń z listy ruchów te ruchy, które zostały dodane
            do listy ruchów do usunięcia.
39      Jeżeli wykonano ruch, to dodaj wykonany ruch do
            listy
40  Zaktualizuj rozwiązanie.
41  Zaktualizuj listę ruchów.
42  Dodaj ruch do listy poprzednio wykonanych ruchów.
43  Przerwij zewnętrzną pętlę jeżeli lista ruchów jest
        pusta.
```

2.2.2 Algorytm ruchów kandydackich

```
1  Wygeneruj losowe rozwiązanie.
2  Powtarzaj:
3      Wygeneruj ruchy kandydackie:
4          Znajdź punkty znajdujące się poza rozwiązaniem.
5          Dla każdego typu ruchu:
6              Jeżeli typ ruchu to zamiana punktów:
7                  Dla każdego punktu w rozwiązaniu:
8                      Dla każdego punktu poza rozwiązaniem:
9                          Oblicz deltę dla zamianych tych
                               dwóch punktów (punktu w
                               rozwiązaniu i punktu poza
                               rozwiązaniem).
10                         Jeżeli delta jest mniejsza od zera,
                               to dodaj ten ruch do listy
                               ruchów kandydackich.
11             W przeciwnym wypadku dla zamiany krawędzi:
12                 Dla każdego punktu w rozwiązaniu:
13                     Dla najbliższych punktów tego punktu:
```

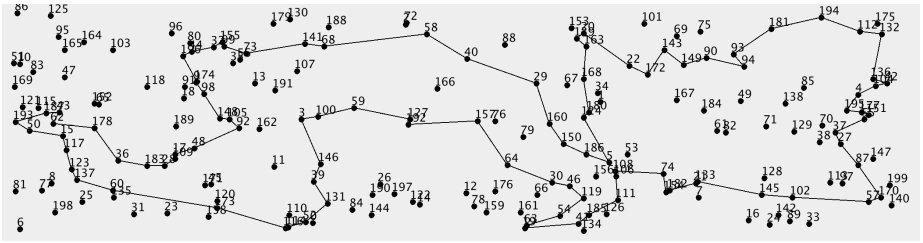
```
14                          Przerwij pętlę, jeżeli nie jest to
                                jeden z pięciu najbliższych
                                punktów.
15                          Przejdź do następnego najbliższego
                                punktu, jeżeli punkt nie leży w
                                aktualnym rozwiązaniu.
16                          Oblicz deltę dla zamiany krawędzi
                                pomiędzy tymi dwoma punktami.
17                          Jeżeli delta jest mniejsza od zera,
                                to dodaj ten ruch do listy
                                ruchów kandydackich.
18          Posortuj wygenerowane ruchy kandydackie od
                najlepszego do najgorszego.
19      Jeżeli nie wygenerowano ruchów kandydackich, to
            przerwij zewnętrzną pętlę, w przeciwnym wypadku:
20          Jeżeli delta najlepszego punktu nie jest mniejsza
                od zera, to przerwij zewnętrzną pętlę, w
                przeciwnym wypadku:
21              Jeżeli typ najlepszego ruchu to
                    pointOuterInnerChange:
22                  W rozwiązaniu zamień punkty wskazane przez
                        najlepszy kandydacki ruch.
23              W przeciwnym wypadku:
24                  W rozwiązaniu zamień krawędzie wskazane
                        przez najlepszy kandydacki ruch.
```

2.3 Wyniki pomiarów

Pomiar	**Wartość średnia**	**Wartość minimalna**	**Wartość maksymalna**	**Średni czas [ms]**	**Minimalny czas [ms]**	**Maksymalny czas [ms]**
Lokalne przeszukiwanie w wersji stromej dla kroA200	17054.24	14433.00	19243.00	19400	16502	23334
Wykorzystanie ocen ruchów z poprzednich iteracji dla kroA200	16958.04	14813.00	19128.00	385	276	862
Ruchy kandydackie dla kroA200	17149.35	15565.00	18977.00	489	396	746
Lokalne przeszukiwanie w wersji stromej dla kroB200	17114.17	14682.00	20310.00	18859	15488	22255
Wykorzystanie ocen ruchów z poprzednich iteracji dla kroB200	17077.72	15493.00	19104.00	374	302	467
Ruchy kandydackie dla kroB200	17211.44	15896.00	18945.00	477	373	645

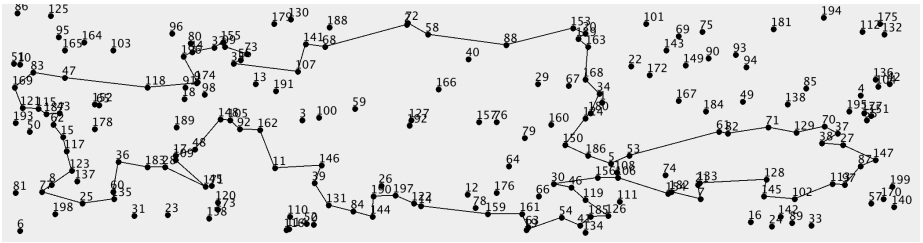
2.4 Wizualizacje najlepszych rozwiązań

2.4.1 Lokalne przeszukiwanie w wersji stromej dla kroA200



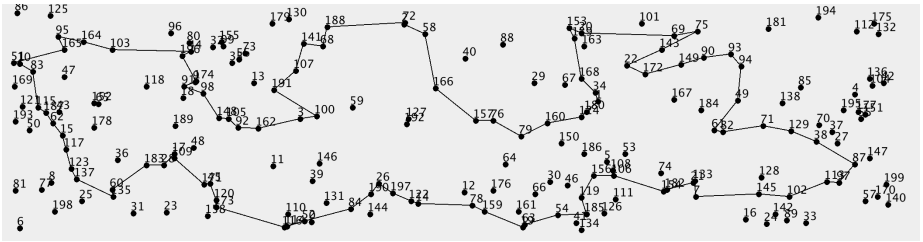
Rysunek 7: Lokalne przeszukiwanie w wersji stromej dla kroA200

2.4.2 Wykorzystanie ocen ruchów z poprzednich iteracji dla kroA200



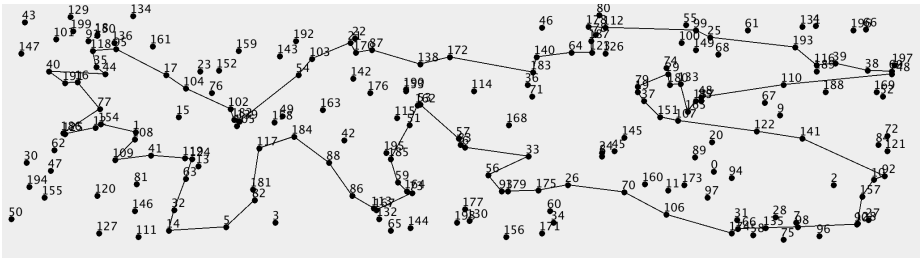
Rysunek 8: Wykorzystanie ocen ruchów z poprzednich iteracji dla kroA200

2.4.3 Ruchy kandydackie dla kroA200



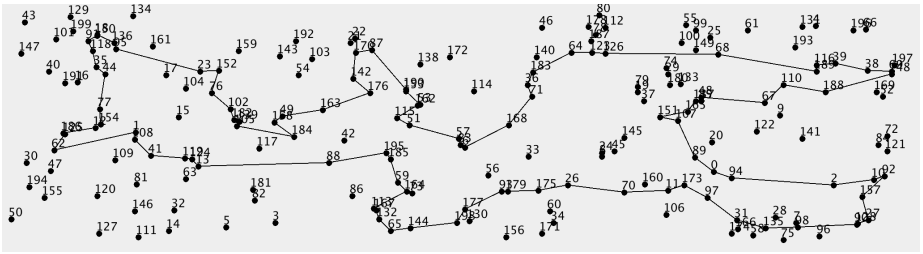
Rysunek 9: Ruchy kandydackie dla kroA200

2.4.4 Lokalne przeszukiwanie w wersji stromej dla kroB200



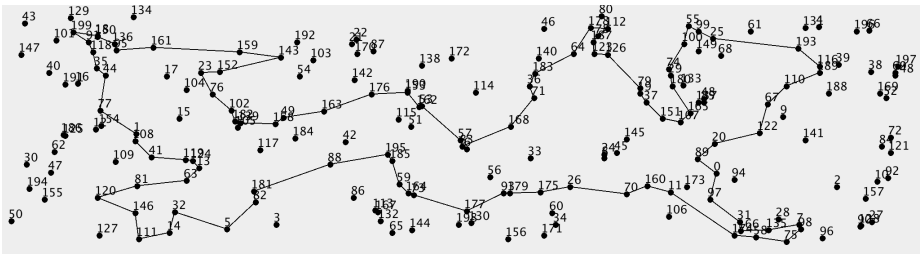
Rysunek 10: Lokalne przeszukiwanie w wersji stromej dla kroB200

2.4.5 Wykorzystanie ocen ruchów z poprzednich iteracji dla kroB200



Rysunek 11: Wykorzystanie ocen ruchów z poprzednich iteracji dla kroB200

2.4.6 Ruchy kandydackie dla kroB200



Rysunek 12: Ruchy kandydackie dla kroB200

2.5 Wnioski

Z wymienionych wyżej pomiarów można wywnioskować, że wykorzystanie oceny ruchów z poprzednich iteracji albo ruchów kandydackich znacząco przyspieszania działanie algorytmu - czas jednej iteracji jest zredukowany o dwa rzędy wielkości. Dystans osiągany przez zmodyfikowane algorytmy jest zbliżony do tych osiąganych przez algorytm wzorcowy, przy czym wykorzystanie oceny ruchów z poprzednich iteracji uzyskuje średni wynik lepszy zarówno od ruchów kandydackich jak i algorytmu wzorcowego.

3 Zadanie 2

3.1 Opis zadania

Zadanie polega na implementacji lokalnego przeszukiwania dla zmodyfikowane problemu komiwojażera. Lokalne przeszukiwanie my być zaimplementowane w wersjach stromej i zachłannej z dwoma różnym rodzajami sąsiedztwa.

3.2 Opis zaimplementowanych algorytmów

3.2.1 Algorytm lokalnego przeszukiwania w wersji zachłannej dla zamiany wierzchołków

```
1  Wygeneruj losowe rozwiązanie.
2  Dopóki nowe rozwiązanie jest lepsze:
3      Oznacz, że nowe rozwiązanie jest gorsze.
4      Wygeneruj losową serię.
5      Dla każdego punktu w serii:
6          Wygeneruj drugą losową serię.
7          Dla każdego punktu w drugiej losowej serii:
8              Jeżeli wybrany punkt jest taki sam jak punkt z
                   pierwszej serii, to przejdź do następnego
                   punktu z drugiej serii.
9              Oblicz deltę.
10             Jeżeli delta jest mniejsza od 0:
11                 Zamień punkt z rozwiązania wskazany, przez
                       pierwszą losową serię, na punkt z
                       drugiej serii.
12                 Oznacz, że nowe rozwiązanie jest lepsze i
                       przejdź do pętli zewnętrznej.
```

3.2.2 Algorytm lokalnego przeszukiwania w wersji stromej dla zamiany wierzchołków

```
1  Wygeneruj losowe rozwiązanie.
2  Dopóki nowe rozwiązanie jest lepsze:
3      Oznacz, że nowe rozwiązanie jest gorsze.
4      Wyzeruj najlepszą deltę i najlepsze punkty do zamiany.
5      Wygeneruj losową serię.
6      Dla każdego punktu w serii:
7          Wygeneruj drugą losową serię.
8          Dla każdego punktu w drugiej losowej serii:
9        Jeżeli wybrany punkt jest taki sam jak punkt z
             pierwszej serii, to przejdź do następnego punktu z
             drugiej serii.
10             Oblicz deltę.
11             Jeżeli delta jest lepsza od najlepszej delty:
12                 Zamień najlepszą deltę na obliczoną deltę.
13                 Zamień znalezione punkty z obecnymi
                       punktami do zamiany.
14                 Oznacz, że nowe rozwiązanie jest lepsze.
```

```
15      Jeżeli najlepsza delta jest mniejsza od 0, to zamień
            znalezione dwa punkty ze sobą.
```

3.2.3 Algorytm lokalnego przeszukiwania w wersji zachłannej dla zamiany krawędzi

```
1  Wygeneruj losowe rozwiązanie.
2  Dopóki nowe rozwiązanie jest lepsze:
3      Oznacz, że nowe rozwiązanie jest gorsze.
4      Wyzeruj najlepszą deltę i najlepsze punkty do zamiany.
5      Wylosuj czy następuje wymiana wierzchołków
           pozatrasowych czy krawędzi wewnątrz trasowych
6      Jeżeli wylosowano wymianę wierzchołków pozatrasowych
7        Wygeneruj losową serię.
8        Dla każdego punktu w serii:
9            Oblicz dystans aktualnego rozwiązania.
10           Wygeneruj drugą losową serię bez punktów z
                 aktualnego rozwiązania.
11           Dla każdego punktu w drugiej losowej serii:
12               Jeżeli wybrany punkt jest taki sam jak punkt
                     z pierwszej serii, to przejdź do
                     następnego punktu z drugiej serii.
13               Oblicz deltę.
14               Jeżeli delta jest mniejsza od zera
15                   Zamień znalezione dwa punkty ze sobą.
16                   Oznacz, że nowe rozwiązanie jest lepsze i
                         przejdź do pętli zewnętrznej.
17     Jeżeli wylosowano wymianę krawędzi
18         Wygeneruj losową serię.
19         Dla każdej krawędzi w serii:
20             Wygeneruj drugą losową serię.
21             Dla każdej krawędzi w drugiej losowej serii:
22                 Jeżeli wybrana krawędź jest taka sama jak
                       krawędź z pierwszej serii lub sąsiadują za
                       sobą, to przejdź do następnej krawędzi z
                       drugiej serii.
23                 Oblicz deltę.
24                 Jeżeli delta jest mniejsza od zera
25                     Zamień znalezione dwie krawędzie.
26                     Oznacz, że nowe rozwiązanie jest lepsze i
                           przejdź do pętli zewnętrznej.
```

3.2.4 Algorytm lokalnego przeszukiwania w wersji stromej dla zamiany krawędzi

```
1  Wygeneruj losowe rozwiązanie.
2  Dopóki nowe rozwiązanie jest lepsze:
3      Oznacz, że nowe rozwiązanie jest gorsze.
4      Wyzeruj najlepszą deltę i najlepsze punkty do zamiany.
5      Wygeneruj losową serię.
6      Dla każdego punktu w serii:
```

```
7           Oblicz dystans aktualnego rozwiązania.
8           Wygeneruj drugą losową serię bez punktów z
                aktualnego rozwiązania.
9           Dla każdego punktu w drugiej losowej serii:
10              Jeżeli wybrany punkt jest taki sam jak punkt z
                    pierwszej serii, to przejdź do następnego
                    punktu z drugiej serii.
11              Oblicz deltę.
12              Jeżeli delta jest lepsza od najlepszej delty:
13                  Zamień najlepszą deltę na obliczoną deltę.
14                  Zamień znalezione punkty z obecnymi
                        punktami do zamiany.
15                  Oznacz, że nowe rozwiązanie jest lepsze.
16      Wygeneruj losową serię.
17      Dla każdego punktu w serii:
18          Wygeneruj drugą losową serię.
19          Dla każdej krawędzi w drugiej losowej serii:
20              Jeżeli wybrana krawędź jest taka sama jak
                    krawędź z pierwszej serii lub sąsiadują za
                    sobą, to przejdź do następnej krawędzi z
                    drugiej serii.
21              Oblicz deltę.
22              Jeżeli delta jest lepsza od najlepszej delty:
23                  Zamień najlepszą deltę na obliczoną deltę.
24                  Zamień znalezione punkty z obecnymi
                        punktami do zamiany.
25                  Oznacz, że nowe rozwiązanie jest lepsze.
26      Jeżeli najlepsza delta jest mniejsza od 0, to zamień
            znalezione dwa punkty ze sobą.
```

3.2.5 Sposób randomizacji kolejności przeglądania

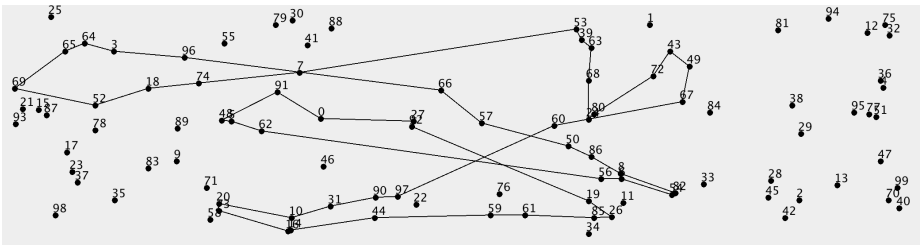
W algorytmie wykorzystano randomizację kolejności przeszukiwania. Są to wszystkie operacje opisane jako generowanie serii. Polegają one na wygenerowaniu indeksów krawędzi lub punktów (w zależności od tego co będzie wymieniane) w losowej kolejności. Następnie, gdy przeglądane są kolejne punkty lub krawędzie są one pobierane z wygenerowanej serii, dzięki czemu kolejność przeglądania za każdą iteracją pętli głównej jest inna.

3.3 Wyniki pomiarów

Pomiar	Wartość średnia	Wartość minimalna	Wartość maksymalna	Średni czas [ms]	Minimalny czas [ms]	Maksymalny czas [ms]
Zamiana wierzchołków w wersji stromej dla kroA100	18116.54	14316.00	16646.00	701	436	925
Zamiana wierzchołków w wersji zachłannej dla kroA100	16747.61	13694.00	16207.00	116	71	224
Zamiana krawędzi w wersji stromej dla kroA100	12545.69	10769.00	11889.00	685	517	858
Zamiana krawędzi w wersji zachłannej dla kroA100	13469.44	11091.00	14262.00	56	26	92
Zamiana wierzchołków w wersji stromej dla kroB100	17721.50	14033.00	15635.00	717	504	1108
Zamiana wierzchołków w wersji zachłannej dla kroB100	16873.84	12300.00	15330.00	106	72	158
Zamiana krawędzi w wersji stromej dla kroB100	12541.73	11199.00	13471.00	665	502	835
Zamiana krawędzi w wersji zachłannej dla kroB100	13261.47	11477.00	13227.00	60	25	115

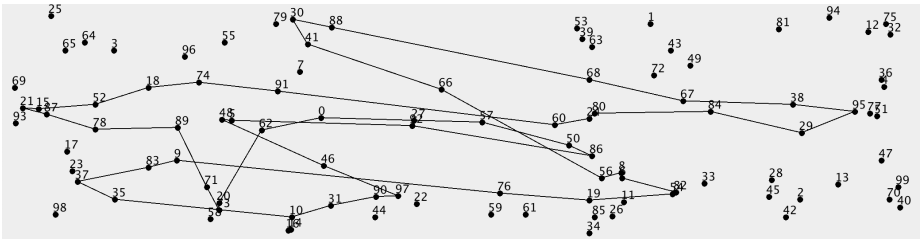
3.4 Wizualizacje najlepszych rozwiązań

3.4.1 Zamiana wierzchołków w wersji stromej dla kroA100



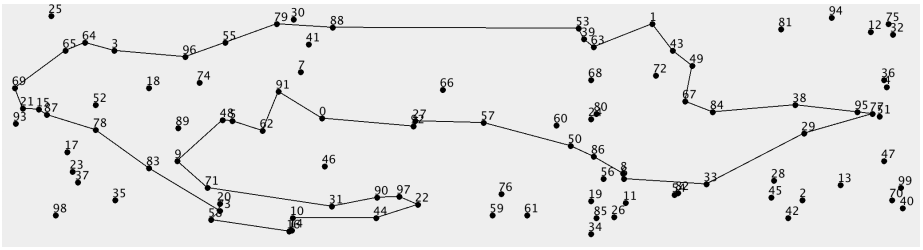
Rysunek 13: Zamiana wierzchołków w wersji stromej dla kroA100

3.4.2 Zamiana wierzchołków w wersji zachłannej dla kroA100



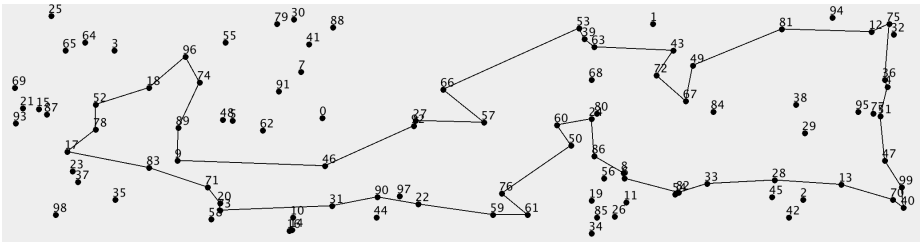
Rysunek 14: Zamiana wierzchołków w wersji zachłannej dla kroA100

3.4.3 Zamiana krawędzi w wersji stromej dla kroA100



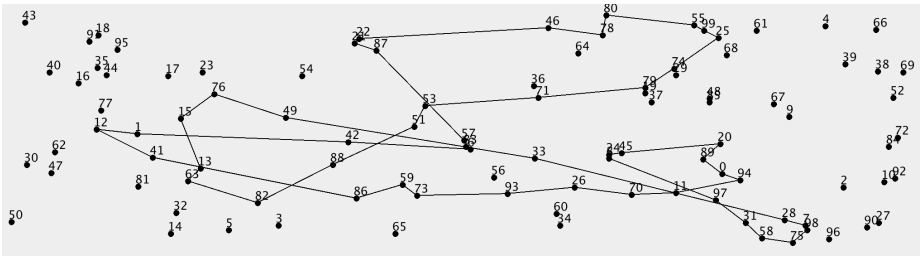
Rysunek 15: Zamiana krawędzi w wersji stromej dla kroA100

3.4.4 Zamiana krawędzi w wersji zachłannej dla kroA100



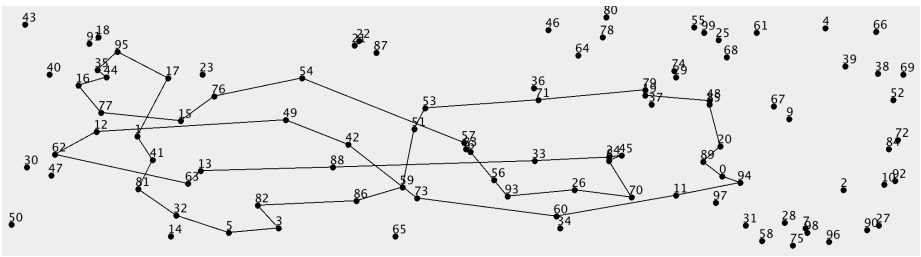
Rysunek 16: Zamiana krawędzi w wersji zachłannej dla kroA100

3.4.5 Zamiana wierzchołków w wersji stromej dla kroB100



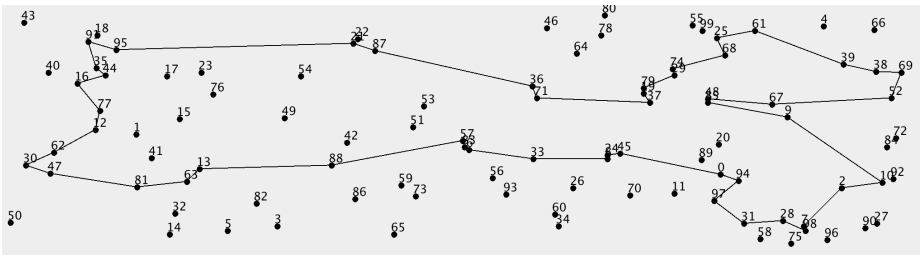
Rysunek 17: Zamiana wierzchołków w wersji stromej dla kroB100

3.4.6 Zamiana wierzchołków w wersji zachłannej dla kroB100



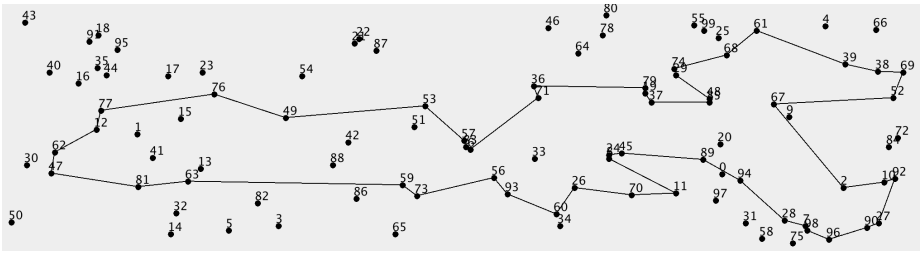
Rysunek 18: Zamiana wierzchołków w wersji zachłannej dla kroB100

3.4.7 Zamiana krawędzi w wersji stromej dla kroB100



Rysunek 19: Zamiana krawędzi w wersji stromej dla kroB100

3.4.8 Zamiana krawędzi w wersji zachłannej dla kroB100



Rysunek 20: Zamiana krawędzi w wersji zachłannej dla kroB100

3.5 Wnioski

Z wymienionych wyżej pomiarów można wywnioskować, że dla podanych warunków problemu zamiana kolejności krawędzi dla obu typów algorytmu daje dużo lepsze wyniki niż wymiana wierzchołków. Moża również zaobserwować, że wersja zachłanna dla zamiany wierzchołków daje średnio lepsze wyniki od wersji stromej. Dla zamiany kolejności krawędzi rozwiązania wyniki są odwrotne - wersja stroma daje średnio lepsze rezultaty od wersji zachłannej. Wersja stroma ma o wiele większe czasy wykonywania z powodu, że musi zawsze przeszukiwać całe rozwiązanie w poszukiwaniu najlepszego ruchu, a wersja zachłanna wybiera pierwszy poprawiający rozwiązanie ruch.

4 Zadanie 1

4.1 Opis zadania

Rozważany problem to zmodyfikowany problem komiwojażera. Dany jest zbiór wierzchołków i macierz odległości pomiędzy każdą parą wierzchołków. Celem zadania jest znalezienie najkrótszej ścieżki zamkniętej przechodzącą przez 50% wszystkich wierzchołków (w przypadku nieparzystej liczby wierzchołków liczba jest zaokrąglana w górę).

4.2 Opis zaimplementowanych algorytmów

4.2.1 Algorytm zachłanny greedy cycle

```
1 Wybierz pierwszy punkt.
2 Wybierz drugi punkt leżący najbliżej pierwszego.
3 Jeżeli nie dodałeś wszystkich punktów:
4   Dla pozostałych wolnych punktów:
5     Dla każdej krawędzi w aktualnym rozwiązaniu:
6       Oblicz koszt dodania punktu do rozwiązania w danej
            krawędzi.
7       Sprawdź czy to jest najlepsze rozwiązanie w danym
            momencie.
8   Dodaj znaleziony najlepszy punkt w wybranej krawędzi do
        cyklu.
```

4.2.2 Algorytm z żalem oparty o 1-żal

```
1  Wybierz pierwszy punkt.
2  Wybierz drugi punkt leżący najbliżej pierwszego.
3  Jeżeli nie dodałeś wszystkich punktów:
4    Dla pozostałych wolnych punktów:
5      Dla każdej krawędzi w aktualnym rozwiązaniu:
6        Oblicz koszt dodania punktu do rozwiązania w danej
             krawędzi.
7        Dodaj punkt do listy potencjalnych rozwiązań wraz z
             kosztem dodania.
8      Oblicz żal dla danego punktu
9    W liście potencjalnych rozwiązań znajdź rozwiązanie z
       największym żalem.
10   Dodaj znalezione rozwiązanie z największym żalem do cyklu.
```

4.3 Wyniki pomiarów

4.3.1 Algorytm zachłanny dla problemu kroA100

Pomiar	Wynik
Wartość średnia	12898.45
Wartość minimalna	11325.00
Wartość maksymalna	14067.00

4.3.2 Algorytm oparty o żal dla problemu kroA100

Pomiar	Wynik
Wartość średnia	16879.11
Wartość minimalna	14456.00
Wartość maksymalna	17899.00

4.3.3 Algorytm zachłanny dla problemu kroB100

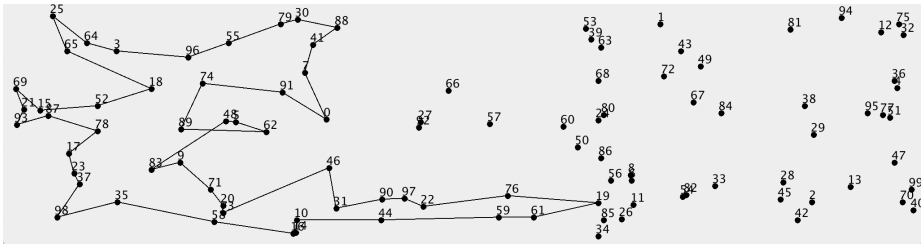
Pomiar	Wynik
Wartość średnia	12710.59
Wartość minimalna	10240.00
Wartość maksymalna	11320.00

4.3.4 Algorytm oparty o żal dla problemu kroB100

Pomiar	Wynik
Wartość średnia	17245.51
Wartość minimalna	15547.00
Wartość maksymalna	16965.00

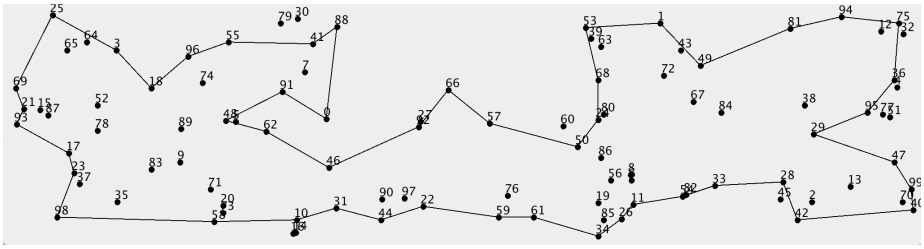
4.4 Wizualizacje najlepszych rozwiązań

4.4.1 Algorytm zachłanny dla problemu kroA100



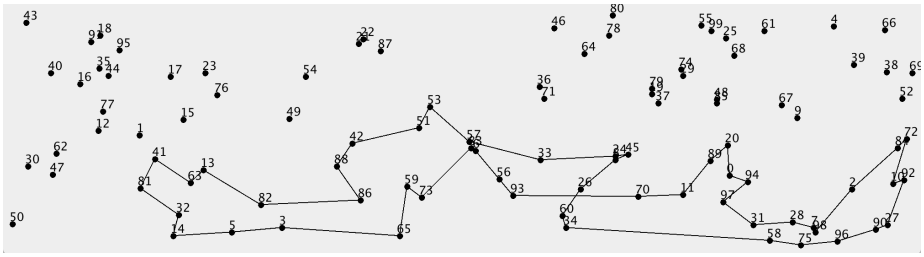
Rysunek 21: Algorytm zachłanny dla problemu kroA100

4.4.2 Algorytm oparty o żal dla problemu kroA100



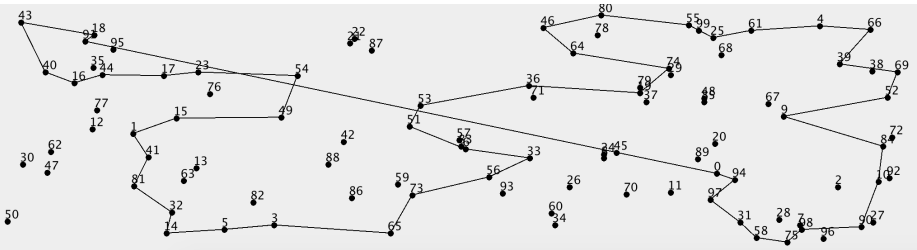
Rysunek 22: Algorytm oparty o żal dla problemu kroA100

4.4.3 Algorytm zachłanny dla problemu kroB100



Rysunek 23: Algorytm zachłanny dla problemu kroB100

4.4.4 Algorytm oparty o żal dla problemu kroB100



Rysunek 24: Algorytm oparty o żal dla problemu kroB100

4.5 Wnioski

Z wymienionych wyżej pomiarów można wywnioskować, że dla podanych warunków problemu (odwiedzanie połowy punktów), algorytm zachłanny radzi sobie lepiej od algorytmu opartego o żal (cykl, który generuje ma mniejszą długość). Przeprowadzono również testy dla przypadku, gdy oba te algorytmy uruchomione zostaną dla wszystkich punktów. Wtedy wyniki są odmienne, algorytm z żalem okazuje się lepszy od algorytmu zachłannego. Jest to spowodowane tym, że dla warunków zadania z odwiedzeniem połowy punktów algorytm z żalem czasami dodaje punkty, które mają duży żal, a w ogóle nie powinny zostać dodane do cyklu z powodu dużego kosztu ich dodania. Gdy odwiedzone mają być wszystkie punkty, koszt dodania punktu nie ma takiego znaczenia, ponieważ prędzej lub później i tak każdy punkt będzie musiał zostać dodany.

5 Kod programu

Repozytorium z kodem programu dostępne jest pod adresem: `https://github.com/adrianstepienfsw/AEM1`