Problem komiwojażera

Adrian Stępień i Wojciech Młyńczak

6 kwietnia 2020

1 Zadanie 2

1.1 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.

1.2 Opis zaimplementowanych algorytmów

1.2.1 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.
```

1.2.2 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:
```

```
        Wygeneruj drugą losową serię.
        Dla każdego punktu w drugiej losowej serii:
      Jeżeli wybrany punkt jest taki sam jak punkt z
          pierwszej serii, to przejdź do następnego punktu z
          drugiej serii.
            Oblicz deltę.
            Jeżeli delta jest lepsza od najlepszej delty:
                Zamień najlepszą deltę na obliczoną deltę.
                Zamień znalezione punkty z obecnymi
                    punktami do zamiany.
                Oznacz, że nowe rozwiązanie jest lepsze.
    Jeżeli najlepsza delta jest mniejsza od 0, to zamień
        znalezione dwa punkty ze sobą.
```

1.2.3 Algorytm lokalnego przeszukiwania w wersji zachłannej dla zamiany krawędzi

```
Wygeneruj losowe rozwiązanie.
Dopóki nowe rozwiązanie jest lepsze:
    Oznacz, że nowe rozwiązanie jest gorsze.
    Wyzeruj najlepszą deltę i najlepsze punkty do zamiany.
    Wylosuj czy następuje wymiana wierzchołków
        pozatrasowych czy krawędzi wewnątrz trasowych
    Jeżeli wylosowano wymianę wierzchołków pozatrasowych
      Wygeneruj losową serię.
      Dla każdego punktu w serii:
          Oblicz dystans aktualnego rozwiązania.
          Wygeneruj drugą losową serię bez punktów z
              aktualnego rozwiązania.
          Dla każdego punktu w drugiej losowej serii:
              Jeżeli wybrany punkt jest taki sam jak punkt
                  z pierwszej serii, to przejdź do
                  następnego punktu z drugiej serii.
              Oblicz deltę.
              Jeżeli delta jest mniejsza od zera
                  Zamień znalezione dwa punkty ze sobą.
                  Oznacz, że nowe rozwiązanie jest lepsze i
                      przejdź do pętli zewnętrznej.
  Jeżeli wylosowano wymianę krawędzi
      Wygeneruj losową serię.
      Dla każdej krawędzi w serii:
          Wygeneruj drugą losową serię.
          Dla każdej krawędzi w drugiej losowej serii:
              Jeżeli wybrana krawędź jest taka sama jak
                  krawędź z pierwszej serii lub sąsiadują za
                  sobą, to przejdź do następnej krawędzi z
                  drugiej serii.
              Oblicz deltę.
              Jeżeli delta jest mniejsza od zera
                  Zamień znalezione dwie krawędzie.
                  Oznacz, że nowe rozwiązanie jest lepsze i
                      przejdź do pętli zewnętrznej.
```

1.2.4 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.
```

1.2.5 Sposób randomizacji kolejności przeglądania

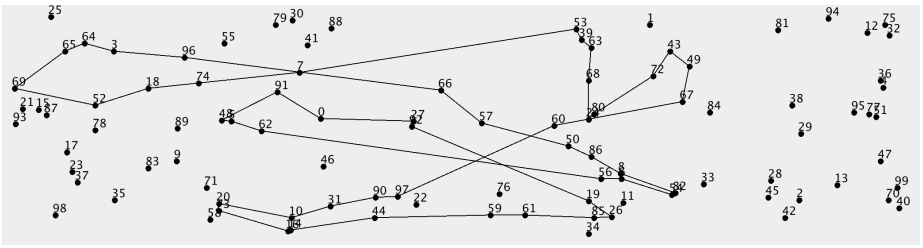
W algorytmie wykorzystano randomizację kolejności przeszukiwania. Są to wszystkie operacje opisane jako generowanie serii. Polegają one na wygenerowaniu indeksów krawędzi lub punktów (w zależności od tego co będzie wymieniane) w losowej kolejności. Następnie, gdy przeglądane są kolejne punkty lub krawędzie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

1.3 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

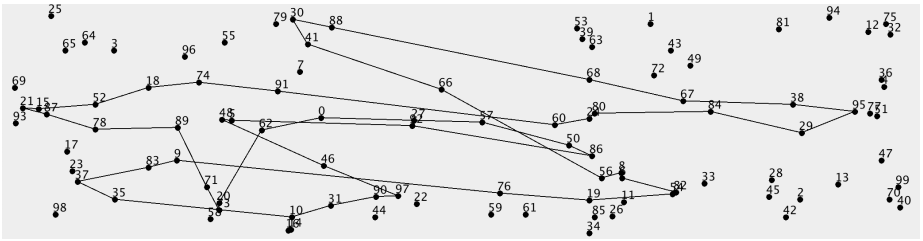
1.4 Wizualizacje najlepszych rozwiązań

1.4.1 Zamiana wierzchołków w wersji stromej dla kroA100



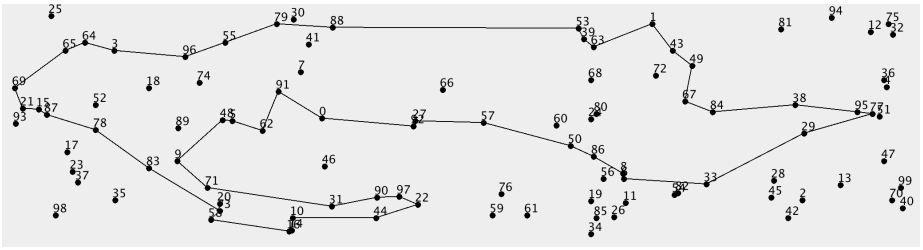
Rysunek 1: Zamiana wierzchołków w wersji stromej dla kroA100

1.4.2 Zamiana wierzchołków w wersji zachłannej dla kroA100



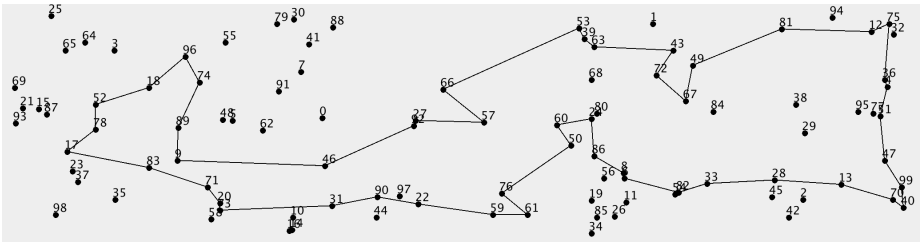
Rysunek 2: Zamiana wierzchołków w wersji zachłannej dla kroA100

1.4.3 Zamiana krawędzi w wersji stromej dla kroA100



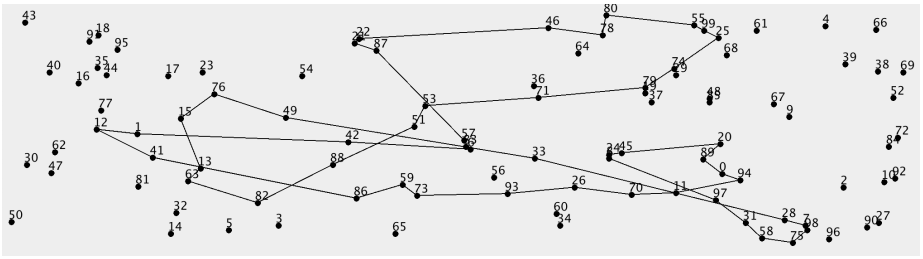
Rysunek 3: Zamiana krawędzi w wersji stromej dla kroA100

1.4.4 Zamiana krawędzi w wersji zachłannej dla kroA100



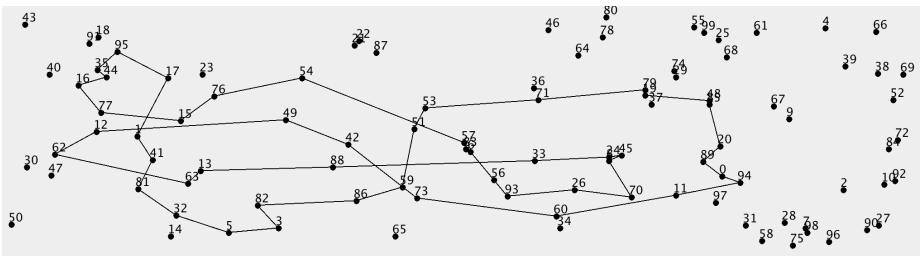
Rysunek 4: Zamiana krawędzi w wersji zachłannej dla kroA100

1.4.5 Zamiana wierzchołków w wersji stromej dla kroB100



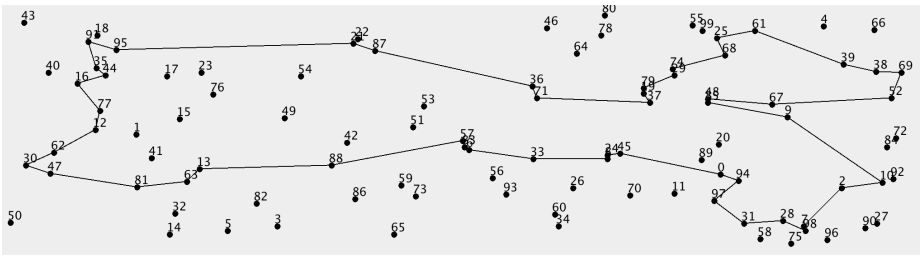
Rysunek 5: Zamiana wierzchołków w wersji stromej dla kroB100

1.4.6 Zamiana wierzchołków w wersji zachłannej dla kroB100



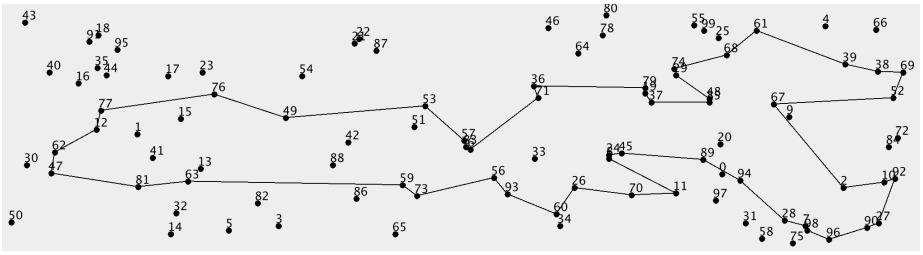
Rysunek 6: Zamiana wierzchołków w wersji zachłannej dla kroB100

1.4.7 Zamiana krawędzi w wersji stromej dla kroB100



Rysunek 7: Zamiana krawędzi w wersji stromej dla kroB100

1.4.8 Zamiana krawędzi w wersji zachłannej dla kroB100



Rysunek 8: Zamiana krawędzi w wersji zachłannej dla kroB100

1.5 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.

2 Zadanie 1

2.1 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).

2.2 Opis zaimplementowanych algorytmów

2.2.1 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.
```

2.2.2 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.
```

2.3 Wyniki pomiarów

2.3.1 Algorytm zachłanny dla problemu kroA100

Pomiar	Wynik
Wartość średnia	12898.45
Wartość minimalna	11325.00
Wartość maksymalna	14067.00

2.3.2 Algorytm oparty o żal dla problemu kroA100

Pomiar	Wynik
Wartość średnia	16879.11
Wartość minimalna	14456.00
Wartość maksymalna	17899.00

2.3.3 Algorytm zachłanny dla problemu kroB100

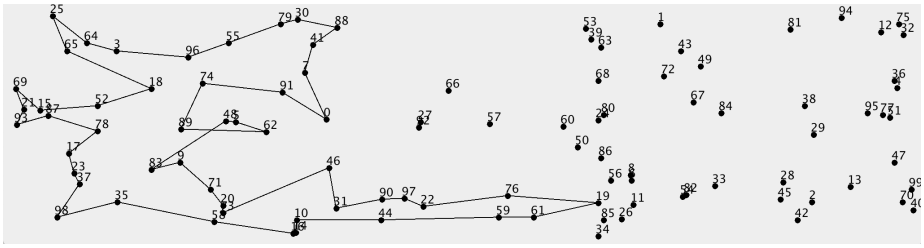
Pomiar	Wynik
Wartość średnia	12710.59
Wartość minimalna	10240.00
Wartość maksymalna	11320.00

2.3.4 Algorytm oparty o żal dla problemu kroB100

Pomiar	Wynik
Wartość średnia	17245.51
Wartość minimalna	15547.00
Wartość maksymalna	16965.00

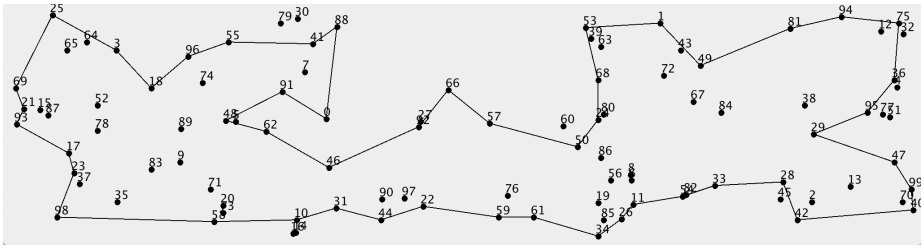
2.4 Wizualizacje najlepszych rozwiązań

2.4.1 Algorytm zachłanny dla problemu kroA100



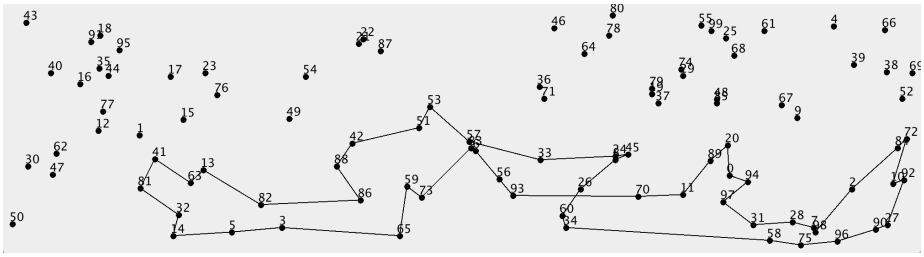
Rysunek 9: Algorytm zachłanny dla problemu kroA100

2.4.2 Algorytm oparty o żal dla problemu kroA100



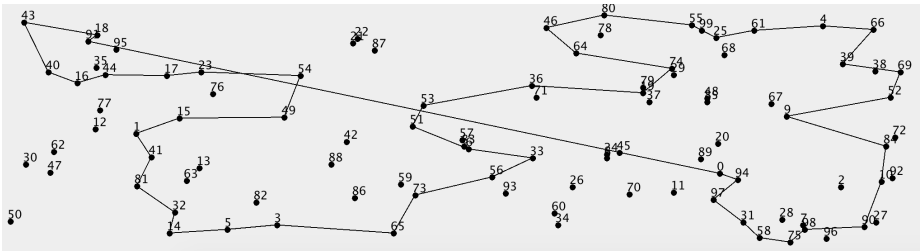
Rysunek 10: Algorytm oparty o żal dla problemu kroA100

2.4.3 Algorytm zachłanny dla problemu kroB100



Rysunek 11: Algorytm zachłanny dla problemu kroB100

2.4.4 Algorytm oparty o żal dla problemu kroB100



Rysunek 12: Algorytm oparty o żal dla problemu kroB100

2.5 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.

3 Kod programu

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