Lokalne przeszukiwanie

Autorzy: Dominik Maćkowiak 151915, Szymon Szymankiewicz 151821

Opis zadania

Zadanie polega na implementacji metod lokalnego przeszukiwania dla problemu optymalizacji tras. Należy zaimplementować dwie wersje algorytmu lokalnego przeszukiwania:

- **Steepest (strome)** – wybiera najlepszy możliwy ruch w każdej iteracji.
- **Greedy (zachłanne)** – wybiera pierwszy lepszy ruch poprawiający rozwiązanie, przeglądając sąsiedztwo w losowej kolejności.

Każda z tych metod powinna być zaimplementowana z dwoma rodzajami sąsiedztwa:

- **Sąsiedztwo oparte na wymianie wierzchołków między cyklami**
- **Sąsiedztwo oparte na zmianie kolejności wierzchołków w trasie (wymiana dwóch wierzchołków lub dwóch krawędzi)**

Każdy algorytm powinien być uruchamiany 100 razy dla każdej instancji, zaczynając od dwóch różnych typów rozwiązań początkowych:

- **Losowe rozwiązania** – generowane przypadkowo.
- **Rozwiązania uzyskane z heurystyki** – wykorzystujące najlepszą heurystykę opracowaną w poprzednim zadaniu.

Dodatkowo należy zaimplementować **algorytm losowego błądzenia**, który wykonuje losowe ruchy i działa przez taki sam czas jak najwolniejsza metoda lokalnego przeszukiwania.

Dane są dwie instancje (“kroA200” oraz kroB200”) z biblioteki TSPlib, które zawierają informacje o położeniu konkretnych wierzchołków. Celem zadania jest utworzenie z każdej instancji dwa rozłączne cykle, tak aby ich łączna długość była jak najmniejsza.

Algorytmy

Operacje wymiany

1. **Zamiana wierzchołków w obrębie cyklu**

Dla każdej pary wierzchołków (i,j) w cyklu:
Zamień miejscami wierzchołki i oraz j
Oblicz zmianę kosztu (delta) wynikającą z tej zamiany
Zwróć nowy cykl oraz deltę

2. **Zamiana wierzchołków między cyklami**

Dla każdej pary wierzchołków (i z pierwszego cyklu, j z drugiego cyklu):
Zamień miejscami wierzchołek i z pierwszego cyklu oraz j z drugiego cyklu
Oblicz zmianę kosztu (delta) wynikającą z tej zamiany
Zwróć nowe wersje obu cykli oraz deltę

3. **Zamiana krawędzi w obrębie cyklu**

Dla każdej pary krawędzi (i,j) w cyklu, gdzie j>i+1:
Odwróć fragment cyklu pomiędzy i a j
Oblicz zmianę kosztu (delta) wynikającą z tej zamiany
Zwróć nowy cykl oraz deltę

Metoda zachłannego przeszukiwania

Sposób randomizacji: Algorytm wykonuje operacje wymiany w ustalonej kolejności, ale kolejność sprawdzania możliwych ruchów wewnątrz każdej operacji jest losowa, poprzez randomizację listy wszystkich możliwych par dla danej operacji. Powoduje to, że za każdym razem pierwszy znaleziony poprawiający ruch może być inny, co prowadzi do różnych wyników w kolejnych uruchomieniach. Kolejność sprawdzania operacji jest stała, więc zaimplementowany sposób randomizacji nie jest całkowicie losowy.

Wygeneruj rozwiązanie startowe (losowe lub heurystyczne)
Oblicz wartość funkcji celu dla początkowego rozwiązania
Ustaw zmienną poprawy na **"prawda"**

Powtarzaj dopóki znaleziono poprawę:
Ustaw zmienną poprawy na **"fałsz"**

// **Sąsiedztwo 1: Ruchy w obrębie cyklu**
Dla każdego cyklu:
Sprawdź zamiany wierzchołków
Sprawdź zamiany krawędzi
Jeśli ruch poprawia rozwiązanie → **Wykonaj pierwszy znaleziony ruch poprawiający**, zapisz nowe rozwiązanie, ustaw poprawę na **"prawda"**, przerwij pętlę

// **Sąsiedztwo 2: Ruchy między cyklami**
Sprawdź zamiany wierzchołków między cyklami
Sprawdź zamiany wierzchołków wewnątrz każdego cyklu

Sprawdź zamiany krawędzi w każdym cyklu
Jeśli ruch poprawia rozwiązanie → **Wykonaj pierwszy znaleziony ruch poprawiający**, zapisz nowe rozwiązanie, ustaw poprawę na **"prawda"**, przerwij pętlę

Zwróć najlepsze znalezione rozwiązanie oraz jego długość

Metoda stromego przeszukiwania

Wygeneruj rozwiązanie startowe (losowe lub heurystyczne)
Oblicz długość początkowego rozwiązania
Ustaw zmienną poprawy na **"prawda"**

Powtarzaj dopóki znaleziono poprawę:
Ustaw zmienną poprawy na **"fałsz"**
Ustaw najlepszą znalezioną poprawę na **0**
Ustaw najlepszy ruch na **"brak"**

// **Sąsiedztwo 1: Ruchy w obrębie cyklu**
Dla każdego cyklu:
Sprawdź wszystkie możliwe zamiany wierzchołków i wybierz najlepszą, jeżeli długość (delta) została zmniejszona
Sprawdź wszystkie możliwe zamiany krawędzi i wybierz najlepszą, jeżeli długość (delta) została zmniejszona

// **Sąsiedztwo 2: Ruchy między cyklami**
Sprawdź wszystkie możliwe zamiany wierzchołków między cyklami i wybierz najlepszą, jeżeli długość (delta) została zmniejszona
Sprawdź wszystkie możliwe zamiany wierzchołków wewnątrz każdego cyklu i wybierz najlepszą, jeżeli długość (delta) została zmniejszona
Sprawdź wszystkie możliwe zamiany krawędzi w każdym cyklu i wybierz najlepszą, jeżeli długość (delta) została zmniejszona

Jeśli znaleziono poprawiający ruch:
Wykonaj ruch
Zapisz nowe rozwiązanie
Ustaw poprawę na **"prawda"**

Zwróć najlepsze znalezione rozwiązanie oraz jego długość

Metoda losowego błądzenia

Wygeneruj losowo lub heurystyczne początkowe rozwiązanie (dwa cykle)
Ustaw najlepsze rozwiązanie jako początkowe
Rozpocznij pomiar czasu

Powtarzaj dopóki czas działania < max_time (średnio najwolniejsza z wersji lokalnego przeszukiwania):

Wybierz losowo typ ruchu:

Jeżeli ruch **wewnątrz cyklu**:

Wybierz losowy cykl

Wykonaj losową zamianę dwóch wierzchołków w cyklu

Jeżeli ruch **między cyklami**:

Wykonaj losową zamianę dwóch wierzchołków między cyklami

Jeżeli ruch **na krawędziach**:

Wybierz losowy cykl

Wykonaj losową zamianę krawędzi

Oblicz długość nowego rozwiązania

Jeżeli nowe rozwiązanie jest lepsze:

Zapisz jako najlepsze znalezione

Zwróć najlepsze rozwiązanie

Wyniki eksperymentu obliczeniowego

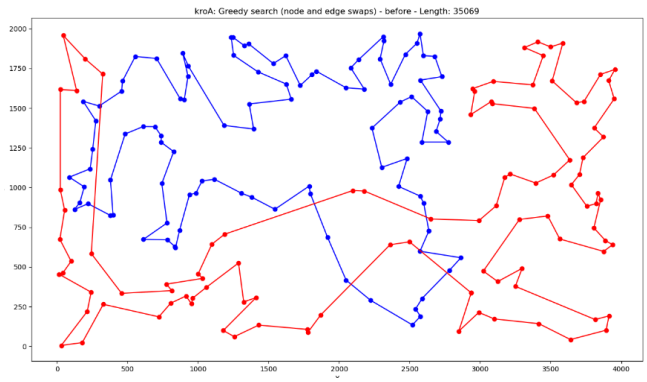
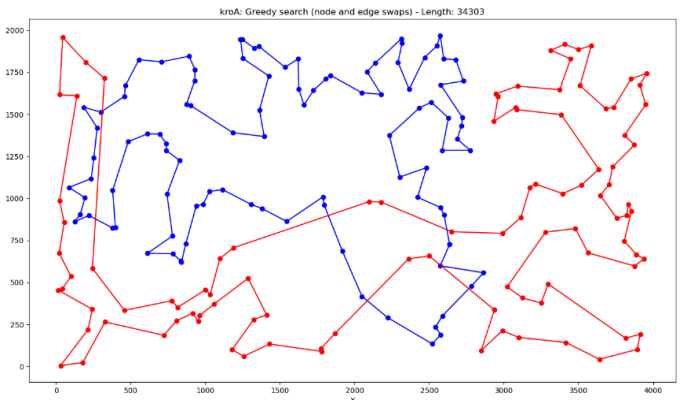
Przeprowadziliśmy eksperyment, który polegał na zmierzeniu długości dwóch utworzonych cykli. Dla każdej instancji przeprowadziliśmy 100 testów. Obliczyliśmy średnią długość z tych testów, wybraliśmy odcinek najdłuższy i najkrótszy oraz średni czas wykonania algorytmów.

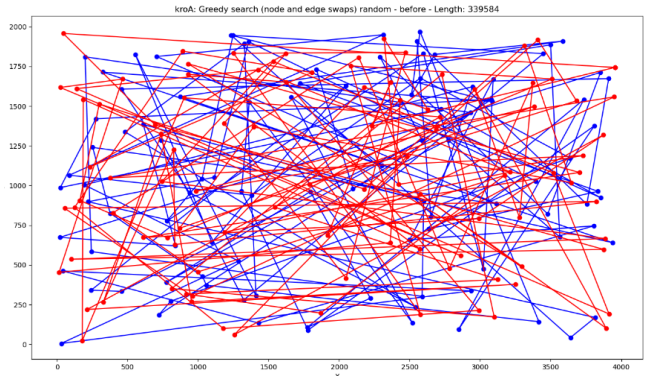
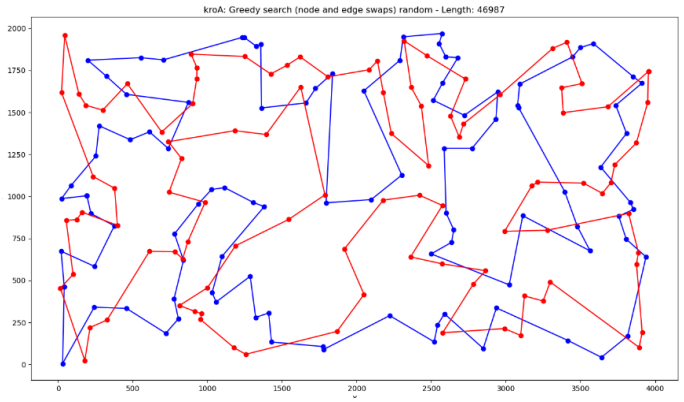
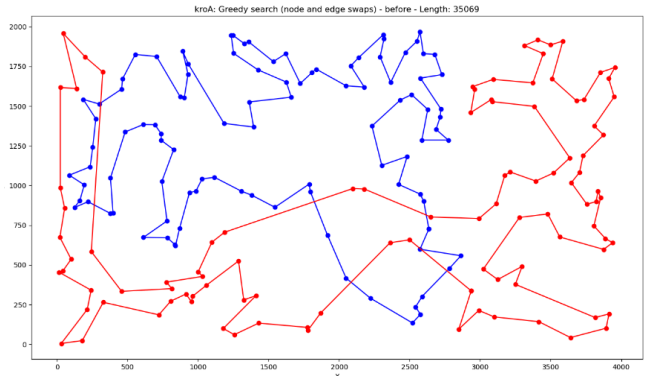
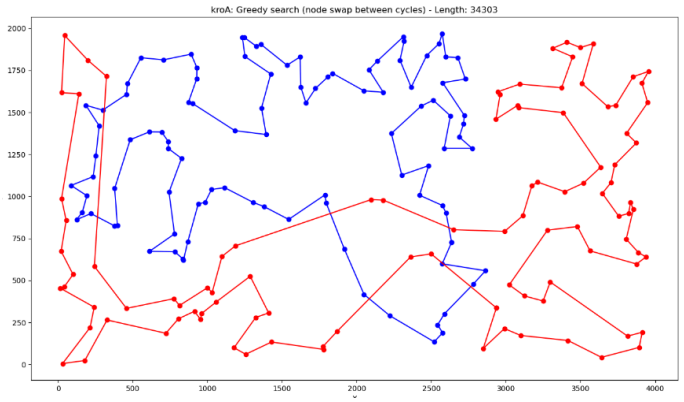
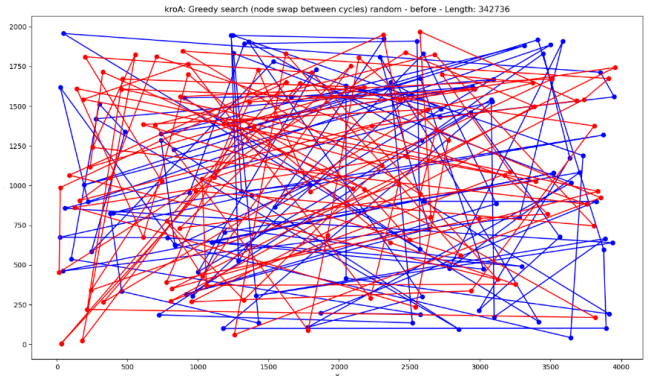
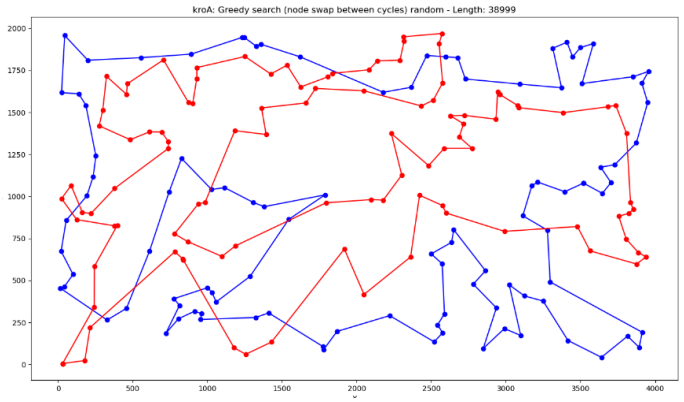
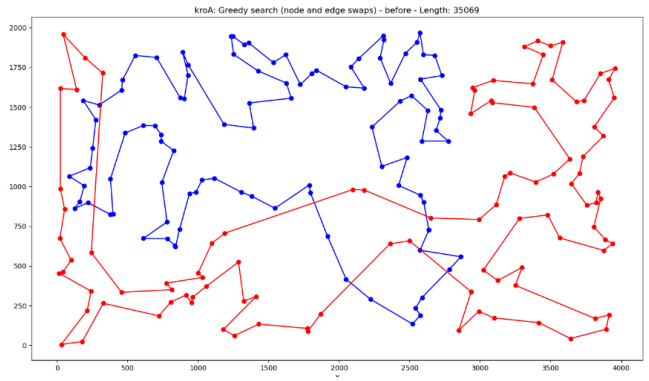
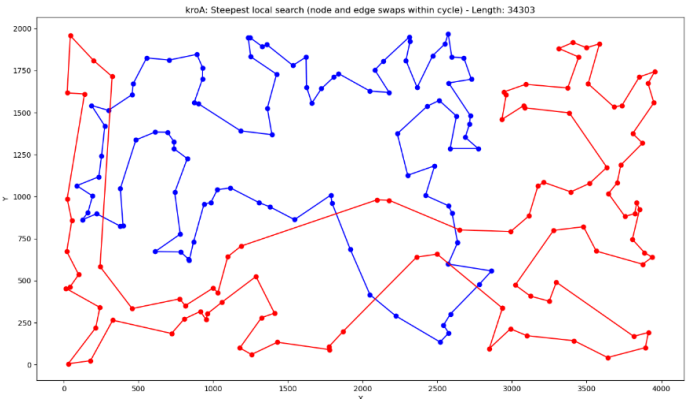
	kroA200				kroB200			
	start	średnia	min	max	start	średnia	min	max
Zachłanny wewnętrznotrasowy	35069	34303.0	34303	34303	35246	34351.0	34351	34351
Zachłanny wewnętrznotrasowy (losowy)	342736	45409.0	45409	45409	323589	47309.0	47309	47309
Zachłanny wymiana	35069	34310.0	34310	34310	35246	34132.0	34132	34132
Zachłanny wymiana (losowy)	342736	38999.0	38999	38999	323589	39742.0	39742	39742
Stromy wewnętrznotrasowy	35069	34303.0	34303	34303	35246	34351.0	34351	34351
Stromy wewnętrznotrasowy (losowy)	342736	43763.0	43763	43763	323589	48793.0	48793	48793
Stromy wymiana	35069	34303.0	34303	34303	35246	34028.0	34028	34028
Stromy wymiana (losowy)	342736	36144.0	36144	36144	323589	38963.0	38963	38963
Random search	35069	35069.0	35069	35069	35246	35246.0	35246	35246

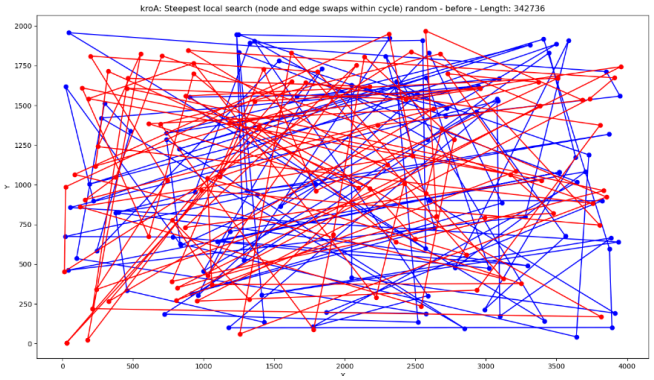
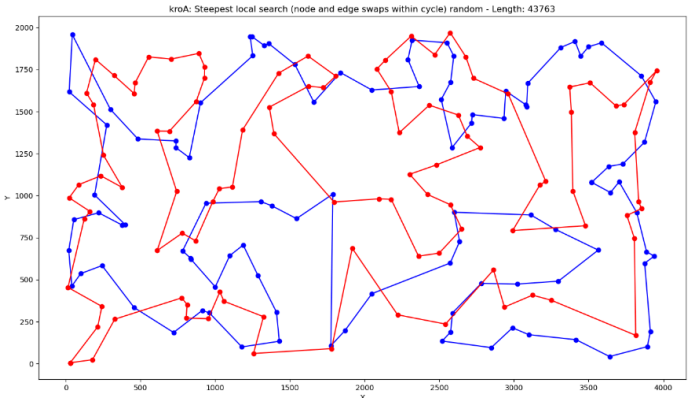
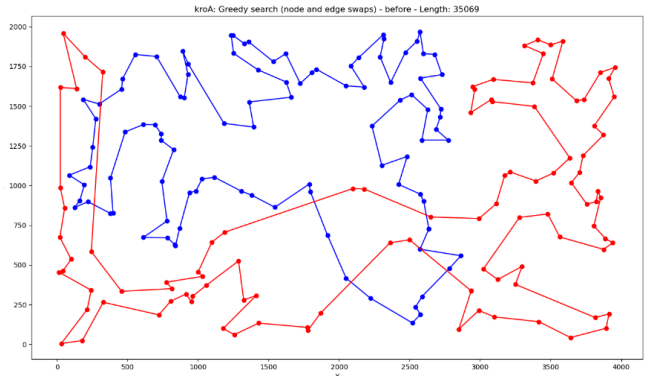
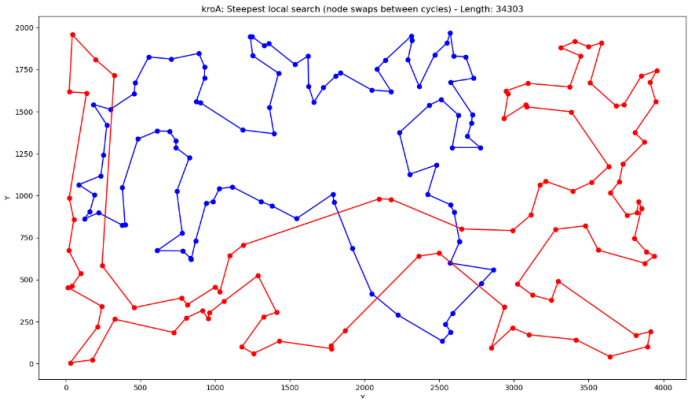
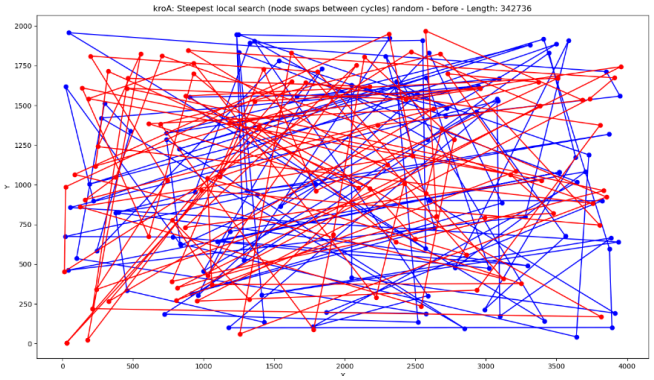
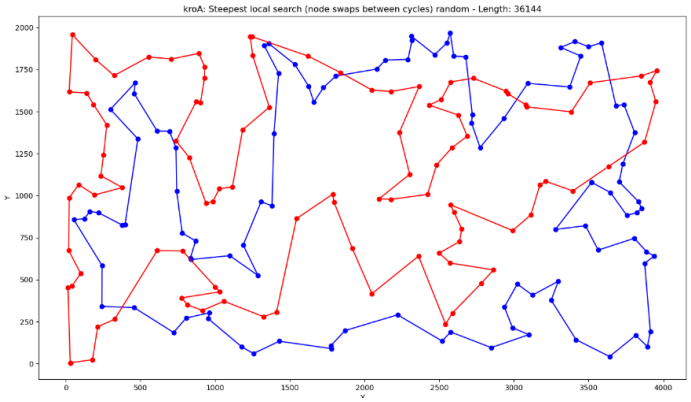
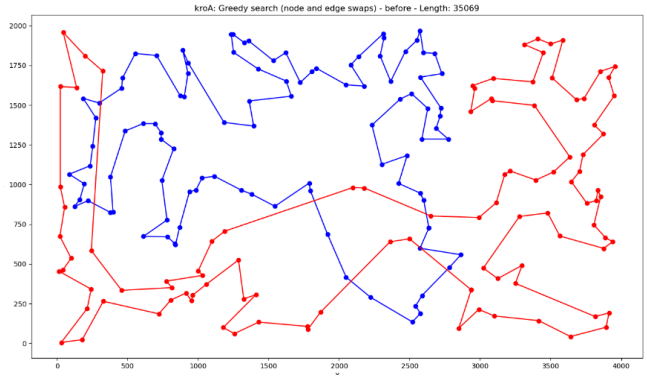
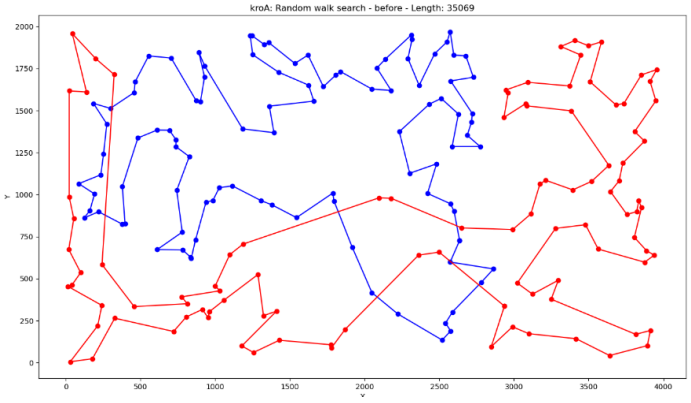
	kroA200			kroB200		
	średnia (s)	min (s)	max (s)	średnia (s)	min (s)	max (s)
Zachłanny wewnętrznotrasowy	0.5680	0.5680	0.5680	0.7227	0.7227	0.7227
Zachłanny wewnętrznotrasowy (losowy)	15.8595	15.8595	15.8595	14.1242	14.1242	14.1242
Zachłanny wymiana	1.1128	1.1128	1.1128	1.2746	1.2746	1.2746
Zachłanny wymiana (losowy)	27.2554	27.2554	27.2554	26.8271	26.8271	26.8271
Stromy wewnętrznotrasowy	0.4830	0.4830	0.4830	0.6127	0.6127	0.6127
Stromy wewnętrznotrasowy (losowy)	8.8425	8.8425	8.8425	9.5813	9.5813	9.5813
Stromy wymiana	0.8072	0.8072	0.8072	1.2800	1.2800	1.2800
Stromy wymiana (losowy)	15.7232	15.7232	15.7232	14.9875	14.9875	14.9875
Random search	0.4830	0.4830	0.4830	0.6127	0.6127	0.6127

Wizualizacja dla najlepszych rozwiązań

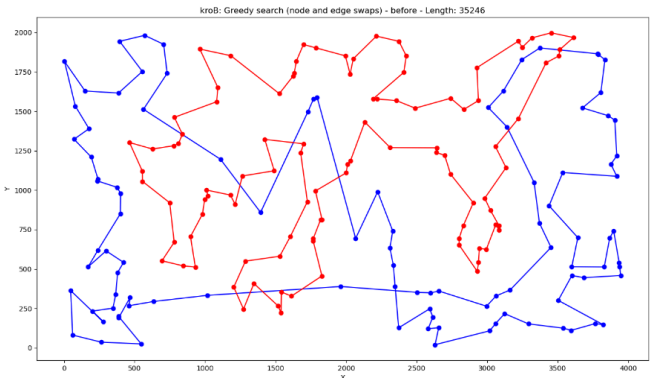
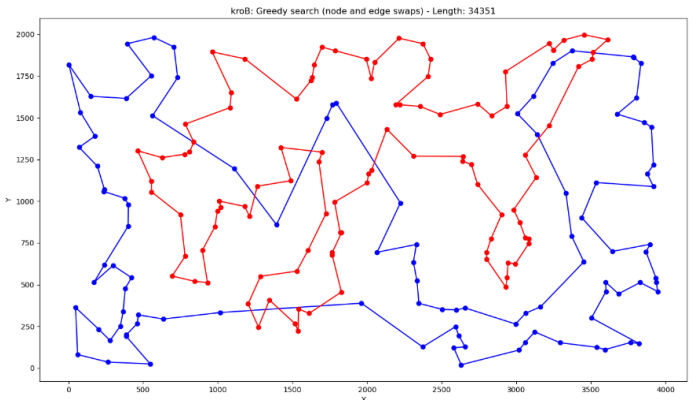
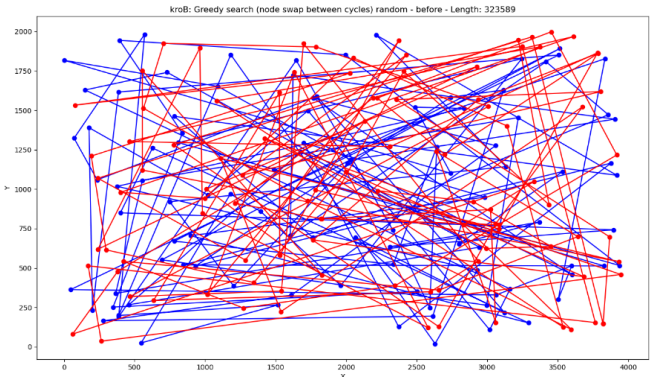
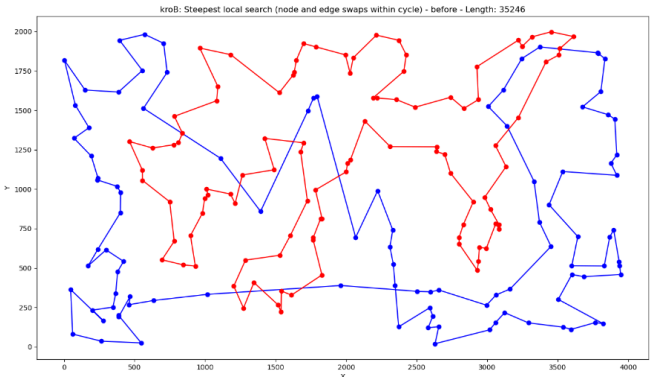
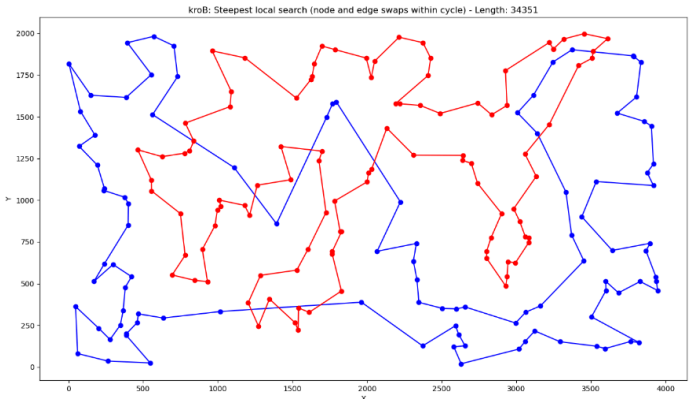
Zestaw kroA200

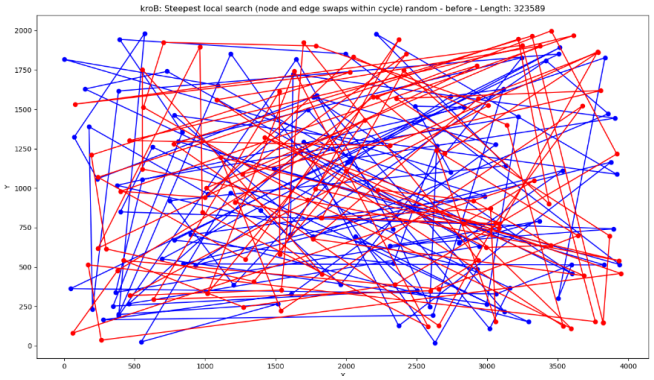
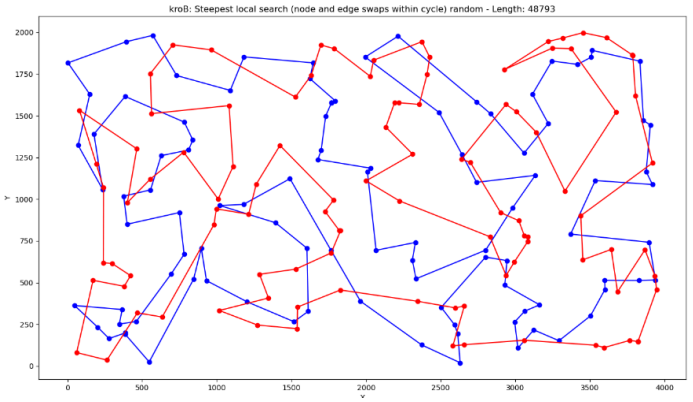
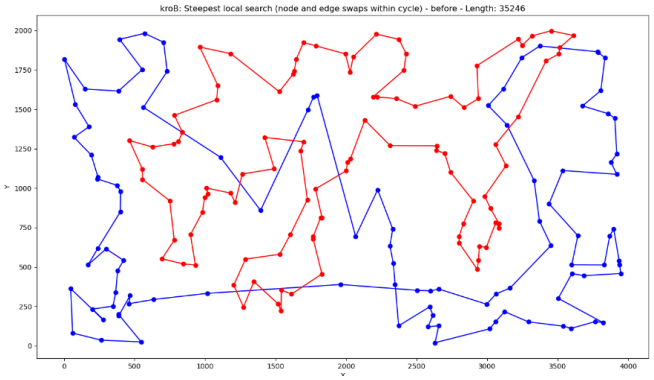
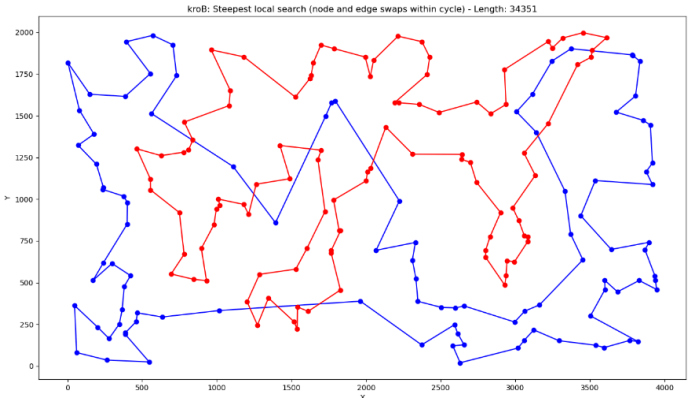
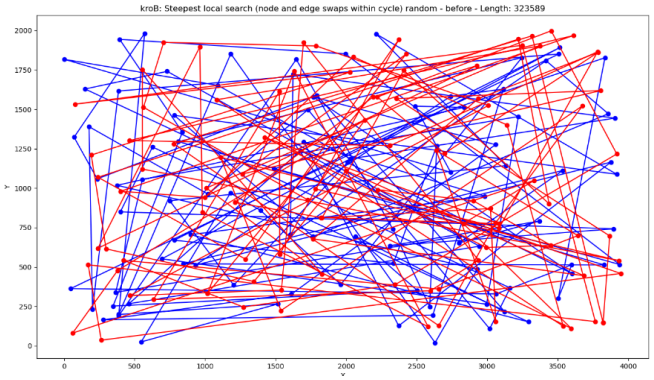
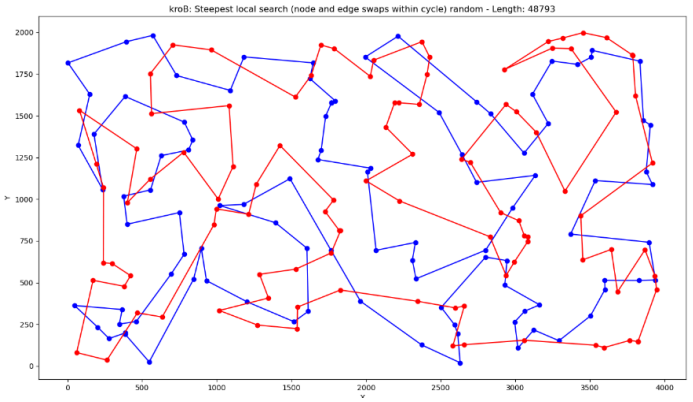
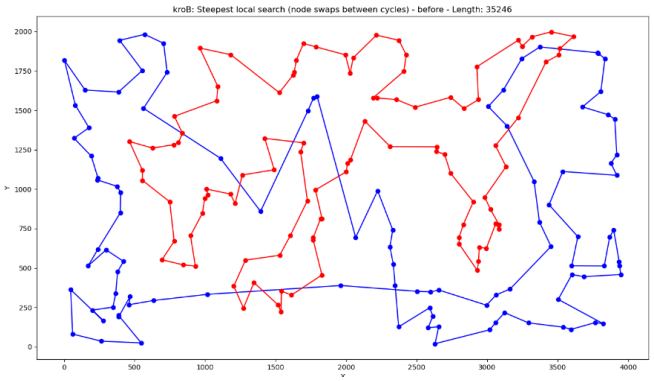
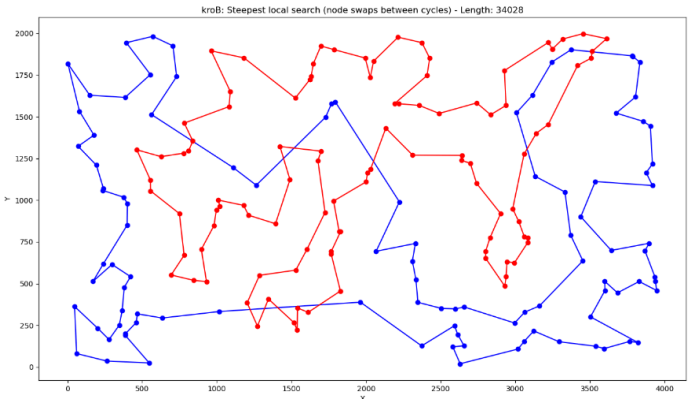
	Przed	Po
Zachłanny wewnętrzno trasowy		

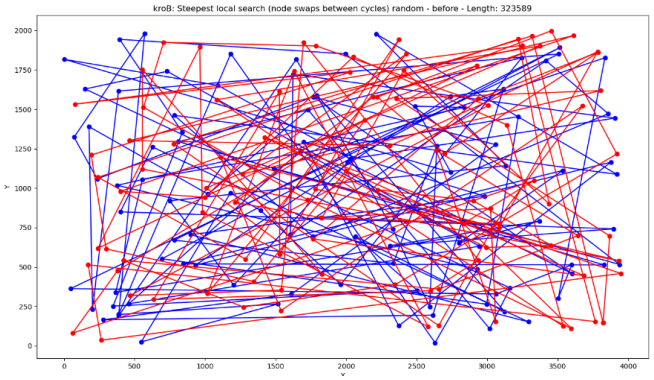
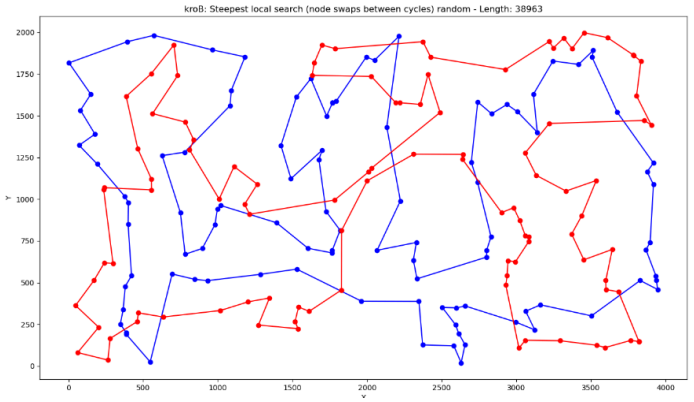
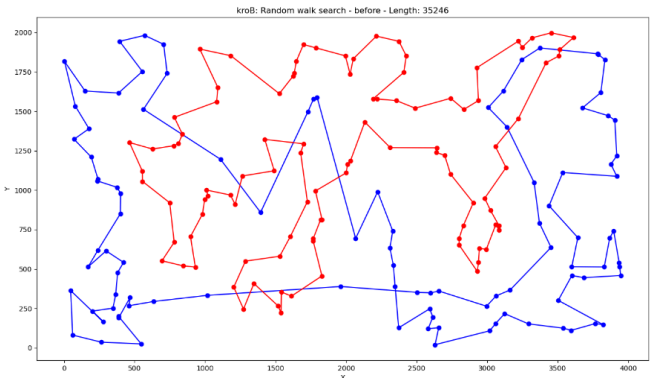
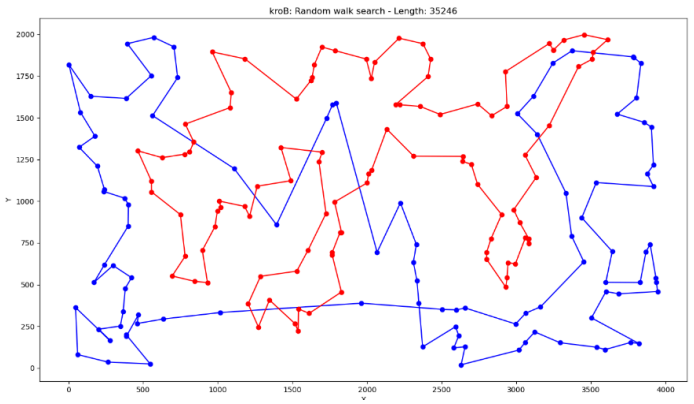
Zachłanny wewnętrzno trasowy (losowy)		
Zachłanny wymiana		
Zachłanny wymiana (losowy)		
Stromy wewnętrzno trasowy		

Stromy wewnętrzno trasowy (losowy)		
Stromy wymiana		
Stromy wymiana (losowy)		
Random search		

Zestaw kroB200

	Przed	Po
Zachłanny wewnętrzno trasowy		
Zachłanny wewnętrzno trasowy (losowy)		
Zachłanny wymiana		

Zachłanny wymiana (losowy)		
Stromy wewnętrzno trasowy		
Stromy wewnętrzno trasowy (losowy)		
Stromy wymiana		

Stromy wymiana (losowy)		
Random search		

Wnioski

Kod programu: https://github.com/szymon240/lab2_imo