

Optymalizacja efektywności lokalnego przeszukiwania

Autorzy: Dominik Maćkowiak 151915, Szymon Szymankiewicz 151821

Opis zadania

Celem zadania jest przyspieszenie lokalnego przeszukiwania w wersji steepest descent z użyciem losowych rozwiązań startowych. Aby poprawić efektywność, implementujemy dwie osobne wersje algorytmu:

1. **Z uporządkowaną listą ruchów (LM)** – wykorzystuje oceny ruchów z poprzednich iteracji, uwzględniając zarówno ruchy wewnątrz-, jak i międzytrasowe.
2. **Z ruchem kandydackim** – ogranicza liczbę generowanych ruchów do najbardziej obiecujących.

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

Algorytm lokalnego przeszukiwania typu steepest z wymianą krawędzi z listą ruchów

Wygeneruj losowe rozwiązanie początkowe (dwa cykle).

Zainicjuj listę ruchów poprawiających (LM).

Dopóki możliwe są poprawy:

- a. Wygeneruj wszystkie możliwe nowe ruchy.
- b. Wybierz tylko ruchy poprawiające ($\Delta < 0$).
- c. Posortuj je rosnąco według Δ .
- d. Dla każdego ruchu m z listy:
 - i. Pobierz usuwane krawędzie z bieżącego rozwiązania (`get_removed_edges`).
 - ii. Sprawdź ich status (`is_edge_in_cycles`):
 - Jeśli którakolwiek krawędź nie istnieje — usuń m z LM.
 - Jeśli wszystkie są, ale w odwróconym kierunku — pomiń m , ale pozostaw w LM.

- Jeśli wszystkie są w tym samym kierunku — zastosuj **m**, dodaj usunięte krawędzie i ich odwrotności do LM, przejdź do nowej iteracji.

Zwróć najlepsze znalezione rozwiązanie.

Algorytm lokalnego przeszukiwania typu steepest z wymianą krawędzi z ruchami kandydadyckimi

Oblicz początkową długość rozwiązania:

Wyznacz krawędzie kandydackie:

Ustaw flagę poprawy: $\text{improved} \leftarrow \text{True}$

Dopóki improved jest prawdziwe:

a. Ustaw $\text{improved} \leftarrow \text{False}$

b. Ustaw $\text{best_delta} \leftarrow 0$ oraz $\text{best_move} \leftarrow \text{None}$

c. Sprawdź poprawę przez zamiany między cyklami i w cyklach:

- Jeśli $\text{delta} < \text{best_delta}$:

- $\text{best_delta} \leftarrow \text{delta}$

- $\text{best_move} \leftarrow \text{move}$

f. Jeśli znaleziono jakikolwiek ruch poprawiający ($\text{best_move} \neq \text{None}$):

i. Wykonaj ruch zgodnie z typem best_move :

- $\text{first_cycle} \leftarrow \text{new_first}$

- $\text{second_cycle} \leftarrow \text{new_second}$

ii. Zaktualizuj best_length :

- $\text{best_length} \leftarrow \text{best_length} + \text{best_delta}$

iii. Ustaw $\text{improved} \leftarrow \text{True}$

Zwróć najlepsze rozwiązanie

Funkcje pomocnicze:

Pobranie usuwanych krawędzi (get_removed_edges)

Dla podanego ruchu:

– Zidentyfikuj krawędzie, które zostaną usunięte w wyniku ruchu

Zwróć listę tych krawędzi (z kierunkiem)

Sprawdzenie obecności krawędzi (is_edge_in_cycles)

Dla danej krawędzi i aktualnych cykli:

– Jeśli krawędź występuje w cyklu w tym samym kierunku: zwróć 1

– Jeśli w przeciwnym kierunku: zwróć -1

– Jeśli brak: zwróć 0

Wyznaczenie krawędzi kandydackich (get_candidate_edges)

Inicjalizuj pusty słownik candidate_edges.

Dla każdego wierzchołka i:

- a. Posortuj indeksy wierzchołków wg odległości od i:
 - sorted_neighbors ← posortowane indeksy wg DistanceMatrix[i]
- b. Wybierz k najbliższych sąsiadów:
 - candidate_edges[i] ← pierwsze k elementów z sorted_neighbors

Zwróć candidate_edges.

Algorytm lokalnego przeszukiwania typu steepest z wymianą krawędzi z ruchami kandydackimi

Wyniki eksperymentu obliczeniowego

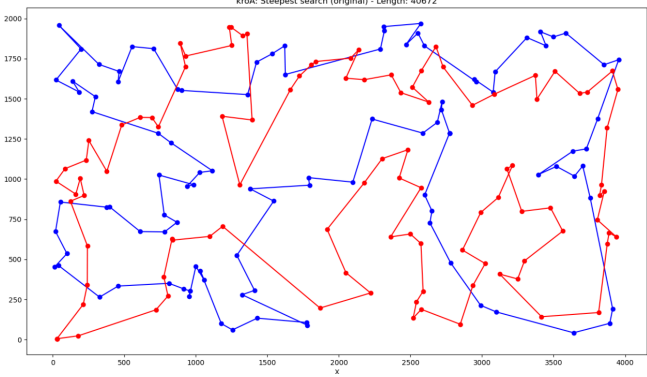
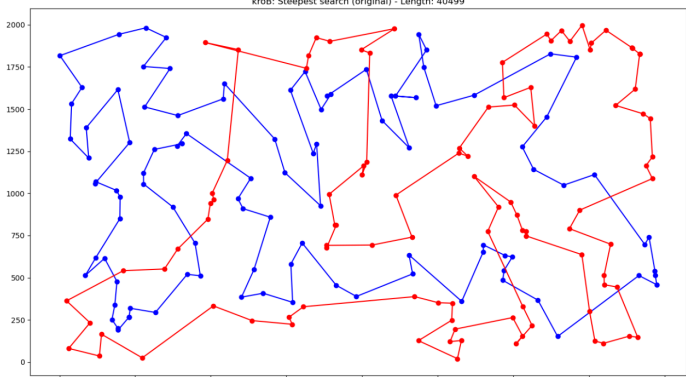
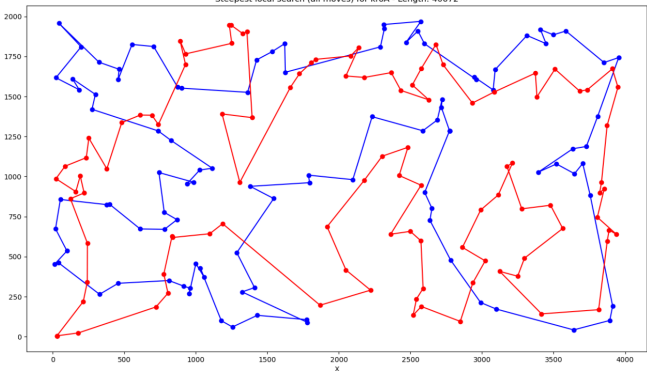
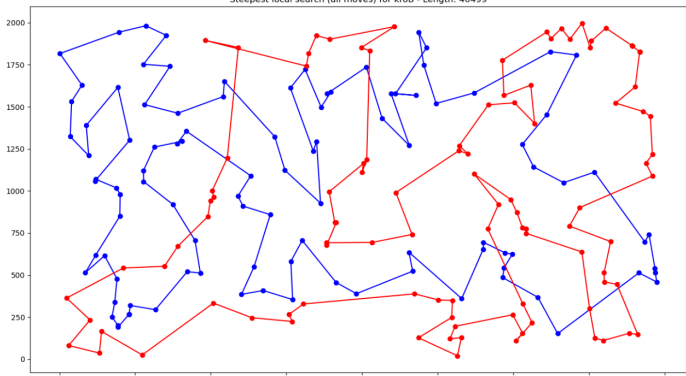
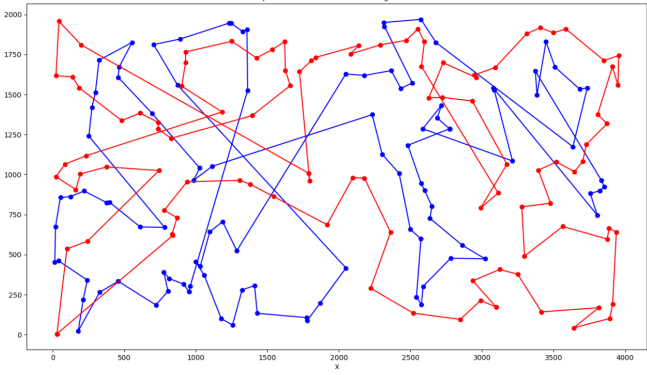
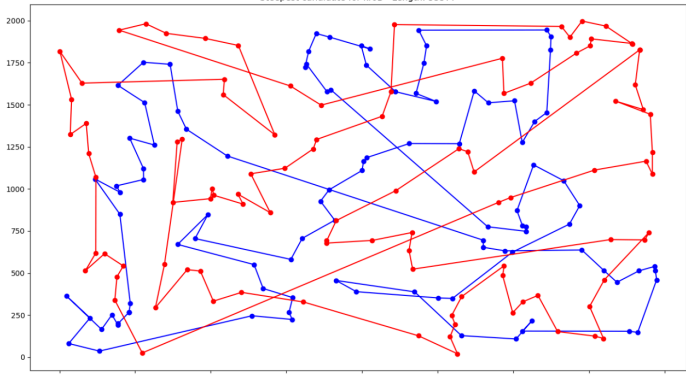
Przeprowadziliśmy eksperyment, który polegał na zmierzeniu długości dwóch utworzonych cykli. Dla każdej instancji przeprowadziliśmy 100 testów. Obliczyliśmy średnią długość z tych testów, wybraliśmy odcinek najdłuższy i najkrótszy oraz średni czas wykonania algorytmów.

	kroA200				kroB200			
	start	średnia	min	max	start	średnia	min	max
Steepest	327477	40672	40672	40672	343869	40499	40499	40499
Steepest + lista ruchów	327477	42974	40672	44012	343869	43845	40499	45593
Steepest + ruchy kandydackie	327477	50722	50722	50722	343869	53577	53577	53577
Ważony 2-żal	BRAK	35815.9	33597	37041	BRAK	36361.5	34581	37976

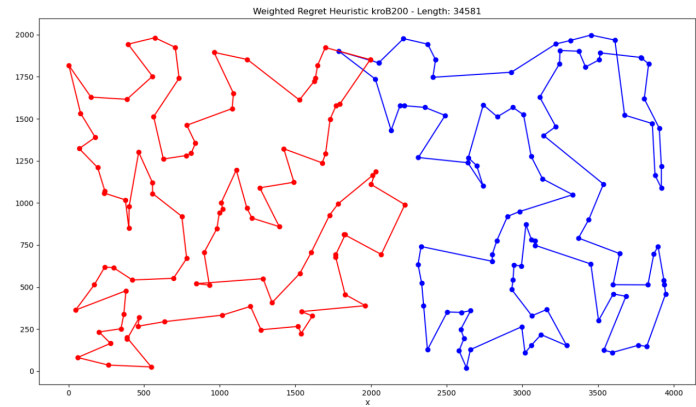
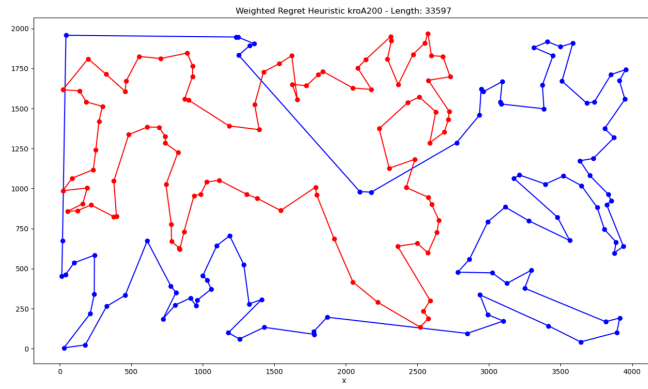
	kroA200			kroB200		
	średnia (s)	min (s)	max (s)	średnia (s)	min (s)	max (s)
Steepest	10.3065	9.8992	10.6925	10.0089	9.4367	10.4371
Steepest + lista ruchów	9.8907	8.6757	10.5930	9.4882	8.8687	10.2906
Steepest + ruchy kandydackie	5.5193	5.3663	5.8416	5.8528	5.6217	6.1246
Ważony 2-żal	1.2871	1.2234	1.4302	1.2765	1.2256	1.3480

Wizualizacja dla najlepszych rozwiązań

Zestaw kroA200

	kroA200	kroB200
Steepest	 <p>Steepest search (original) - Length: 40672</p>	 <p>Steepest search (original) - Length: 40499</p>
Steepest + lista ruchów	 <p>Steepest local search (all moves) for kroA - Length: 40672</p>	 <p>Steepest local search (all moves) for kroB - Length: 40499</p>
Steepest + ruchy kandydacki e	 <p>Steepest candidate for kroA - Length: 50722</p>	 <p>Steepest candidate for kroB - Length: 53577</p>

Ważony 2-żal



Wnioski

Kod programu: <https://github.com/szymon240/imo-lab3>