Inteligentne metody optymalizacji

Laboratorium nr 4

24 stycznia 2024

Prowadzący: prof. dr hab. inż. Andrzej Jaszkiewicz

Autorzy: **Krzysztof Martin** 141275
Bartosz Paliński 145224

Zajęcia poniedziałkowe, 11:45.

Oświadczam/y, że niniejsze sprawozdanie zostało przygotowane wyłącznie przez powyższych autora/ów, a wszystkie elementy pochodzące z innych źródeł zostały odpowiednio zaznaczone i są cytowane w bibliografii.

1 Opis zadania

Celem niniejszego sprawozdania jest przedstawienie wyników badań trzech metod usprawniających lokalne przeszukiwanie:

- Multiple start local search (MSLS),
- Iterated local search z niewielką perturbacją (ILS1),
- Iterated local search z perturbacją typu Destroy-Repair (ILS2).

W dalszej części sprawozdania przedstawione zostaną wyniki badań, wraz z analizą skuteczności poszczególnych metod oraz omówieniem wpływu zastosowanych mechanizmów na jakość wyników i czas przetwarzania.

Efektywność algorytmów była oceniana na podstawie instancji kroaA200 i kroB200 z biblioteki TSPLib. Do uruchomienia algorytmów wykorzystano losowe rozwiązania startowe.

2 Algorytmy

2.1 MSLS

Multiple start local search (MSLS) to metaheurystyczna metoda optymalizacji kombinatorycznej, która polega na wielokrotnym uruchomieniu algorytmu lokalnego przeszukiwania z różnych losowych lub zrandomizowanych punktów startowych.

W każdym uruchomieniu algorytmu losowane jest nowe rozwiązanie startowe, a następnie stosowana jest procedura lokalnego przeszukiwania, która polega na iteracyjnym poprawianiu rozwiązania poprzez wykonywanie kroków, które prowadzą do lepszego rozwiązania. Procedura ta jest powtarzana wielokrotnie, aż do momentu znalezienia najlepszego rozwiązania lub wyczerpania określonej ilości czasu obliczeń.

Jako lokalne przeszukiwanie wykorzystano najlepszą metodę z poprzednich zajęć - **przeszukiwanie lokalne z pamięcią we wersji stromej z wymianą krawędzi.**

1: *bestSolution* = None
2: **Powtarzaj 100 razy**:
3: Wygeneruj losowe rozwiązanie początkowe - *currentSolution*
4: Wykonaj na tym rozwiązaniu algorytm lokalnego przeszukiwania z pamięcią we wersji stromej ze zamianą krawędzi.
5: **Jeżeli** *currentSolution.length() <bestSolution.length() lub bestSolution == None* :
6: bestSolution = currentSolution

2.2 ILS1

W algorytmie ILS1, po znalezieniu lokalnego minimum przez algorytm lokalnego przeszukiwania, dokonywana jest niewielka perturbacja (zmiana) rozwiązania poprzez losową zamianę wierzchołków między cyklami lub losową zamianę krawędzi w cyklu. Następnie algorytm lokalnego przeszukiwania jest ponownie uruchamiany z punktu startowego otrzymanego po perturbacji.

1: Wygeneruj losowe rozwiązanie początkowe - *Solution*
2: Uruchom algorytm lokalnego przeszukiwania na rozwiązaniu Solution .
3: **Powtarzaj**:

4: Dokonaj losowej perturbacji rozwiązania *Solution* tj. losowo zamień wierzchołki między cyklami lub losowo zamień krawędzie w cyklu i zapisz jako *permSolution*.
5: Uruchom algorytm lokalnego przeszukiwania na perturbowanym rozwiązaniu *permSolution*
6: **Jeżeli** *permSolution.length() <Solution.length()* :
7: Solution = permSolution
8: **Jeżeli** upłynął maksymalny czas działania algorytmu:
9: Zwróć rozwiązanie Solution jako najlepsze rozwiązanie znalezione przez algorytm.

2.3 ILS2

W algorytmie ILS2, jako rozwiązanie startowe zostaje przyjete rozwiązanie heurystyki 2-żalu ważonego. Nastepnie dokonywana jest perturbacja rozwiązania poprzez losowe usunięcie 20% wierzchołków, a następnie naprawieniu go za pomocą heurystyki 2-żalu ważonego. Algorytm kontynuuje iteracyjne przeszukiwanie z perturbacjami aż do osiągnięcia maksymalnego czasu działania.

1: Wygeneruj rozwiązanie początkowe oparte na heurystyce 2-żalu ważonego, - *Solution*
2: $toDelete = 0.2 \cdot numberOfVertices$
3: **Powtarzaj**:
4: *permSolution = Solution*
5: **Dopóki** *toDelete*>0:
6: Wylosuj wierzchołek do usunięcia (V),
7: Usuń wierzchołek V z rozwiązania *permSolution*,
8: **Powtarzaj** z prawdopodobienstwem 0.8 i **Dopóki** *toDelete*>0:
9: Usuń wierzchołek V' z rozwiązania *permSolution* natępujący po poprzednio usuniętym wiezchołku.
10: $toDelete = toDelete - 1$,
11: Uruchom algorytm 2-żalu ważonego, aby uzupełnić rozwiązanie *permSolution* o brakujące wierzchołki.
12: **Jeżeli** *permSolution.length() <Solution.length()* :
13: Solution == permSolution
14: **Jeżeli** upłynął maksymalny czas działania algorytmu:
15: Zwróć rozwiązanie Solution jako najlepsze rozwiązanie znalezione przez algorytm.

3 Wyniki eksperymentu obliczeniowego

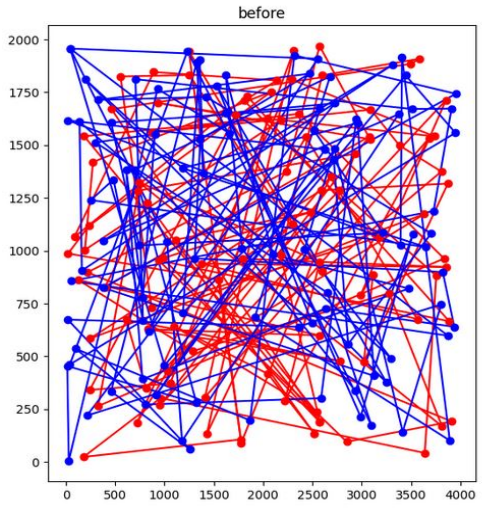
Każdy algorytm został przetestowany 10 razy dla każdej instancji. Średnie, najlepsze oraz najgorsze wyniki są przedstawione w poniższej tabeli wraz z czasami wykonywania.

Tabela 1: Losowe rozwiązanie początkowe (LS - Local search)

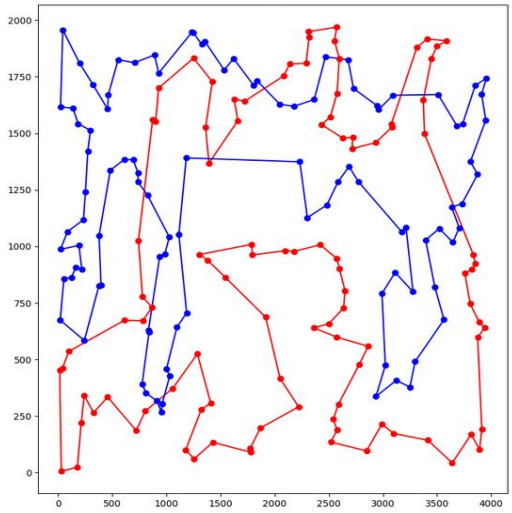
Lokalne przeszukanie	Instancja kroA200	Instancja kroB200
Random	338767 (311756 - 366842)	346899 (297563 - 386124)
LS z pamięcią	38564 (35166 - 40916) - 260,2 ms	38576 (35636 - 41038) - 260,2 ms
MSLS z pamięcią	35845 (35459 - 36266) - 23982,2 ms	35573 (35160 - 35837) - 23748 ms
ILS1	35842 (35151 - 36559) - 24199 ms	36282 (35812 - 36663) - 23858 ms
Ważony 2-żal	35986 (34061 - 39211) - 208,5ms	36470 (32409 - 39274) - 220,1ms
ILS2	32915 (31510 - 34054) - 24199 ms	33091 (31390 - 37462) - 23858ms

4 Wizualizacje najlepszych rozwiązań

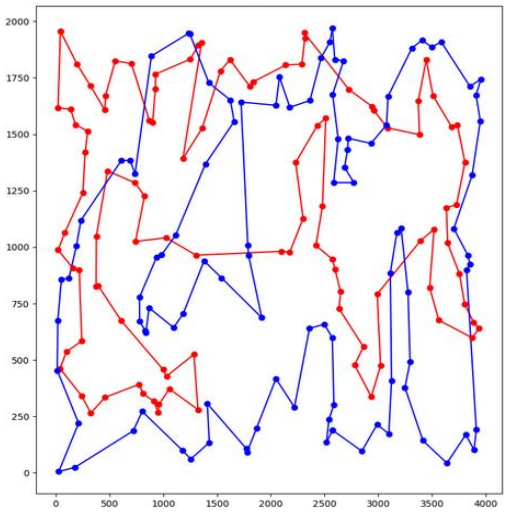
4.1 Instancja kroaA200



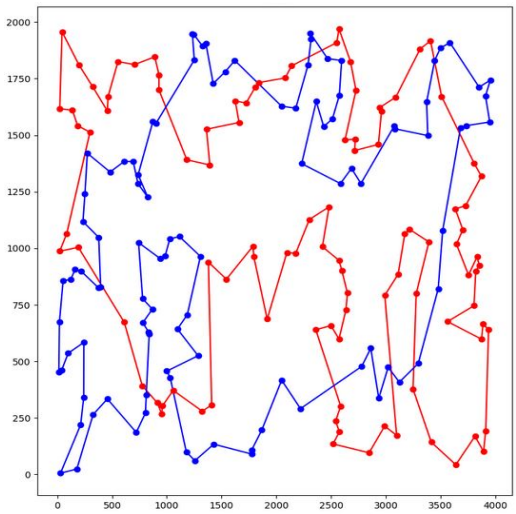
Rysunek 1: Losowe rozwiązanie



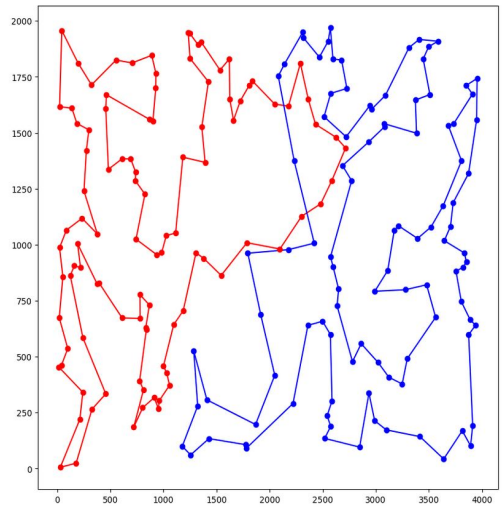
Rysunek 2: ILS1



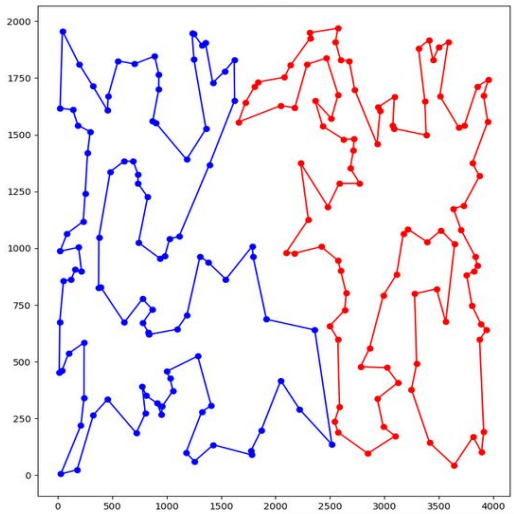
Rysunek 3: Lokalne przeszukiwanie z pamięcią ruchów



Rysunek 4: Wielokrotne lokalne przeszukiwanie z pamięcią ruchów (MLSL)

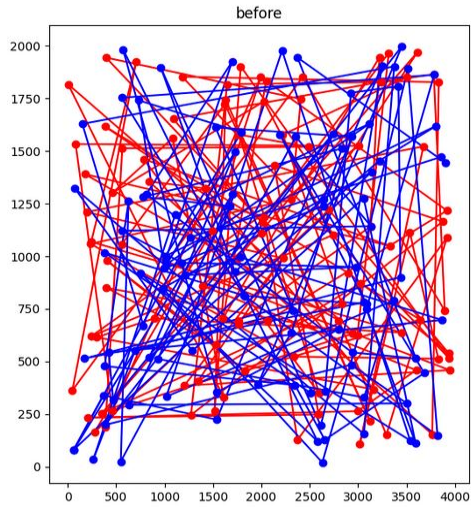


Rysunek 5: 2-żal ważony

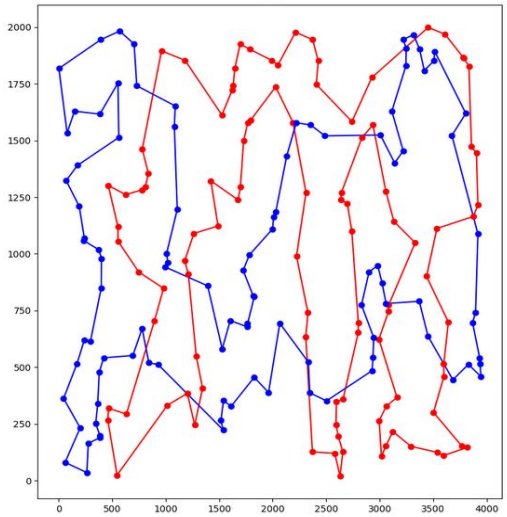


Rysunek 6: ILS2 wykorzystujący 2-żal

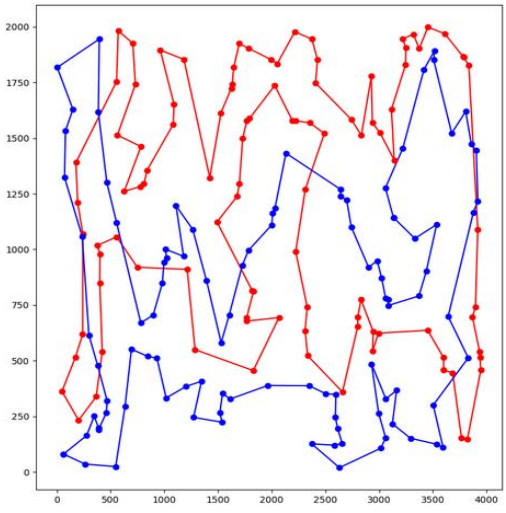
4.2 Instancja kroaB200



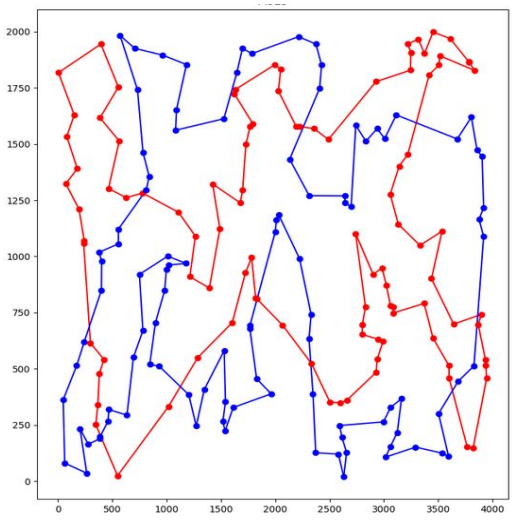
Rysunek 7: Losowe rozwiązanie



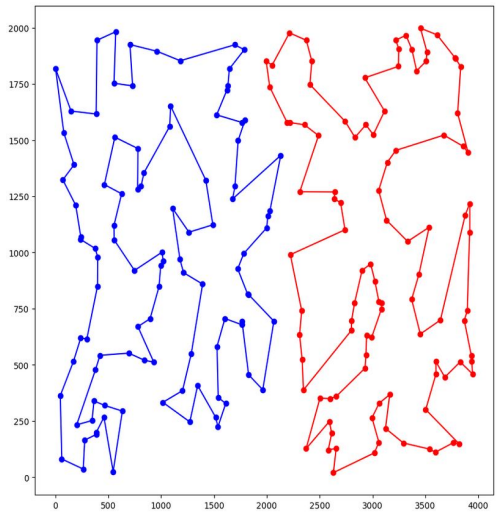
Rysunek 8: ILS1



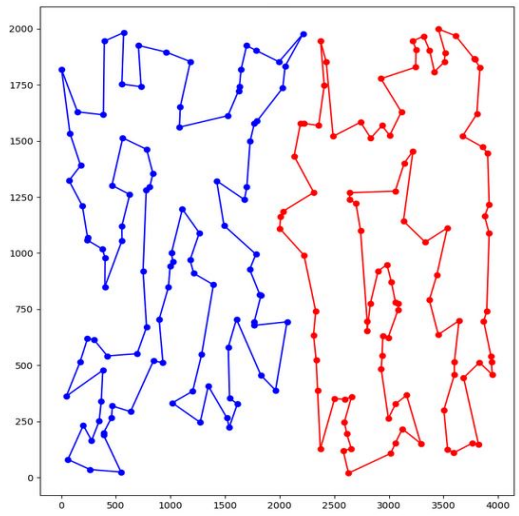
Rysunek 9: Lokalne przeszukiwanie z pamięcią ruchów



Rysunek 10: Wielokrotne lokalne przeszukiwanie z pamięcią ruchów (MLSL)



Rysunek 11: 2-żal ważony



Rysunek 12: ILS2 wykorzystujący 2-żal

5 Wnioski

-
- Algorytm Multiple start local search (MSLS) osiąga średnio wyniki lepsze niż standardowa wersja lokalnego przeszukiwania. Dzięki temu, że MSLS uruchamia loklane przeszukiwanie wielokrotnie, obserwuje się mnieszy ochył między wartościamy nalepszymi, średnimi oraz najgorszymi. Czas wykonywania MSLS zgodnie z oczekiwaniami jest około 100 razy dłuższy niż loklane przeszukiwanie.
- Algorytm ILS1 osiąga porównywalne wyniki z algorytmem MLSL.
- Metody przeszukiwania sąsiedztwa startujące z rozwiązania losowego osiągnęły wyniki gorsze niż heurystyka konstrukcyjna (ważony 2-żal).
- Algorytm ILS2 bazyjący na heurtsyce 2-żalu osiąga średnio 9% lepsze wyniki niż sama heurystyka 2-żalu.

6 Kod programu

Kod źródłowy dostępny on-line: https://github.com/barosz0/IMO/tree/main/lab4