Politechnika Poznańska

Wydział Informatyki i Telekomunikacji

Instytut Informatyki

Zakład Inteligentnych Systemów Wspomagania Decyzji

Heurystyki konstrukcyjne

Inteligentne metody optymalizacji

Paweł Chumski, 144392
pawel.chumski@student.put.poznan.pl

Michał Ciesielski, 145325
michal.p.ciesielski@student.put.poznan.pl

Prowadzący:
dr hab. inż. Andrzej Jaszkiewicz, prof. PP
andrzej.jaszkiewicz@cs.put.poznan.pl

18-03-2023

Spis treści

Wstęp

Celem zadania była implementacja i porównanie trzech heurystyk w celu rozwiązania zmodyfikowanego problemu komiwojażera. Na podstawie danych współrzędnych wierzchołków wejściowych należało utworzyć, a następnie minimalizować dwie rozłączne zamknięte ścieżki (cykle), każdą zawierającą 50% wierzchołków.

Badane heurystyki to:
- Heurystyka najbliższego sąsiada (nearest neighbour)
- Metoda rozbudowy cyklu (greedy cycle)
- Heurystyka oparta na 2-żalu (regret heuristic)

Eksperymenty zostały przeprowadzone na instancjach kroA100.tsp oraz kroB100.tsp z biblioteki TSPLib.

Heurystyka najbliższego sąsiada (nearest neighbour)

1.1 Opis algorytmu

Algorytm najbliższego sąsiada polega na wybraniu losowego punktu startowego, a następnie iteracyjnym wybieraniu najbliższego nieodwiedzonego punktu i dodawaniu go do trasy. Proces jest powtarzany aż do odwiedzenia wszystkich punktów. Algorytm ten nie zawsze daje optymalne rozwiązanie problemu komiwojażera, ale jest stosunkowo prosty i szybki w implementacji. Wadą algorytmu najbliższego sąsiada jest to, że wynik zależy od punktu startowego, co oznacza, że istnieją przypadki, w których algorytm może dać bardzo złe wyniki.

1.2 Pseudokod

Znajdź najbardziej oddalony wierzchołek od bieżącego, od którego zacznie się budowa drugiego cyklu

Inicjalizuj listę *remaining_nodes* zawierająca potencjalnych kandydatów do wydłużenia ścieżki

Inicjalizuj cykle, pierwszy zawierający wierzchołek wybrany, drugi najbardziej od niego oddalony

Powtarzaj

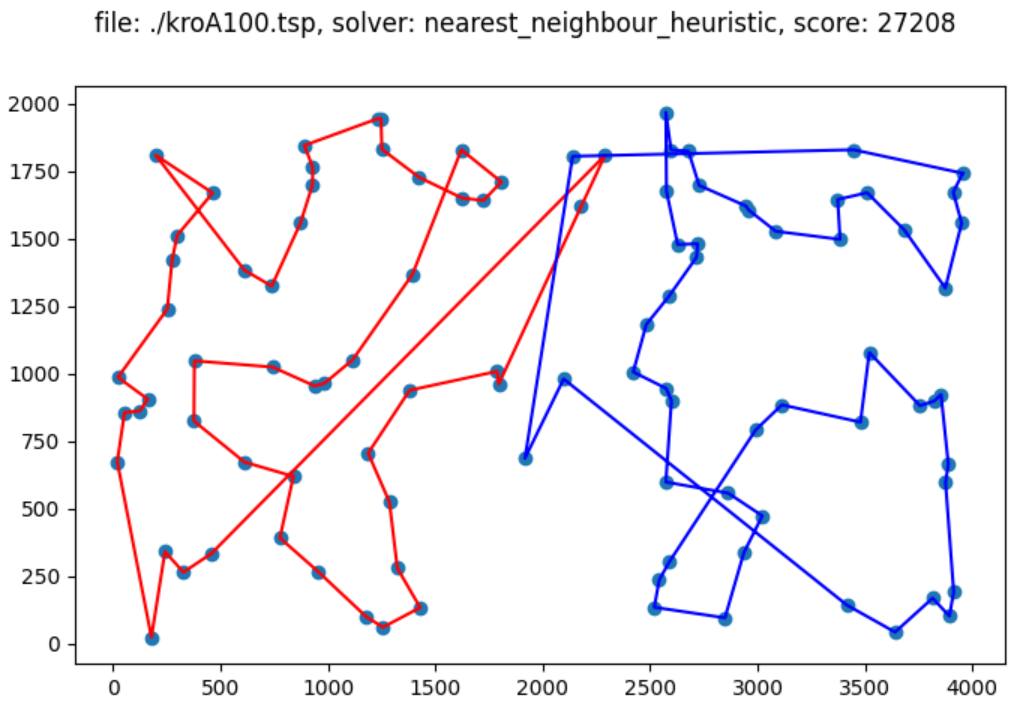
> Wydłuż każdy cykl o najbliższego sąsiada dla ostatnio dodanego wierzchołka
>
> Usuń dodany wierzchołek z listy *remaining_nodes*

Dopóki lista *remaining_nodes* nie jest pusta

Zwróć słownik zawierający listy wierzchołków dla obu cykli

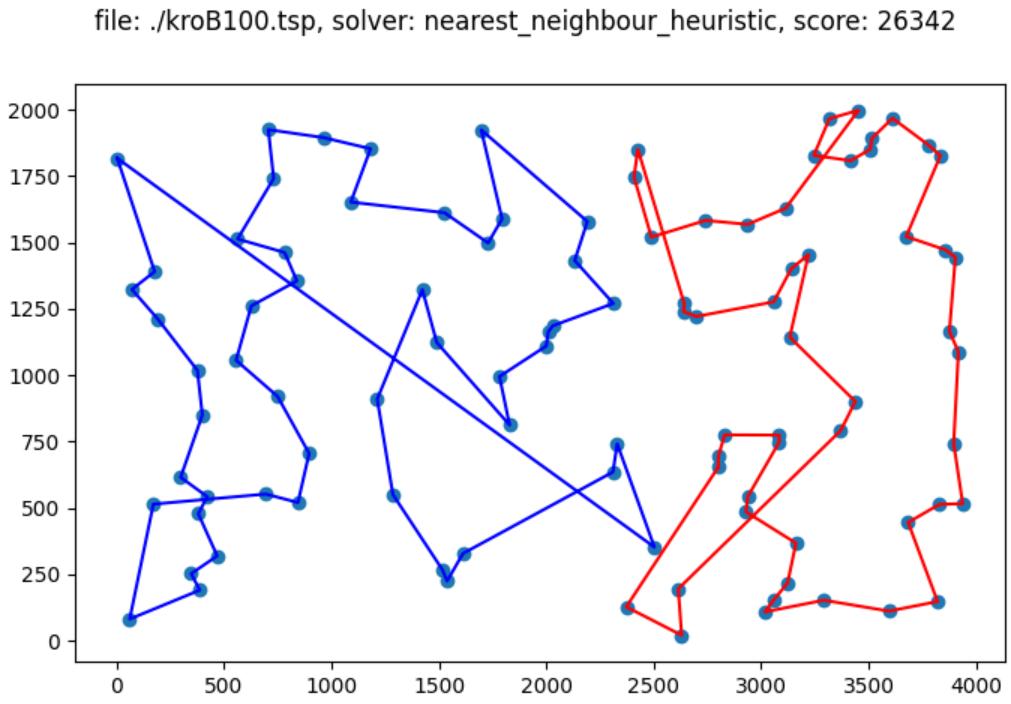
1.3 Wizualizacje

- Wynik dla kroA100:



Rys. 1. Obraz najlepszej ścieżki uzyskany dla heurystyki najbliższego sąsiada dla danych kroA100.

- Wynik dla kroB100:



Rys. 2. Obraz najlepszej ścieżki uzyskany dla heurystyki najbliższego sąsiada dla danych kroB100.

Metoda rozbudowy cyklu (greedy cycle)

2.1 Opis algorytmu

Algorytm Greedy Cycle Heuristics to metoda znajdowania rozwiązania problemu komiwojażera poprzez iteracyjne łączenie cykli o najmniejszej wadze. W każdej iteracji wybierany jest losowy nieodwiedzony punkt startowy, a następnie budowany jest cykl poprzez wybieranie najbliższego nieodwiedzonego punktu. Po zakończeniu budowania cyklu, łączy się go z dotychczasowymi cyklami poprzez wybór cyklu o najmniejszej wadze i połączenie z nim nowo utworzonego cyklu. Proces jest powtarzany aż do odwiedzenia wszystkich punktów i utworzenia jednego cyklu.

2.2 Pseudokod

Znajdź najbardziej oddalony wierzchołek od bieżącego, od którego zacznie się budowa drugiego cyklu

Inicjalizuj listę *remaining_nodes* zawierająca potencjalnych kandydatów do wydłużenia ścieżki

Inicjalizuj cykle, pierwszy zawierający wierzchołek wybrany, drugi najbardziej od niego oddalony

Powtarzaj dla każdego cyklu w słowniku cycles

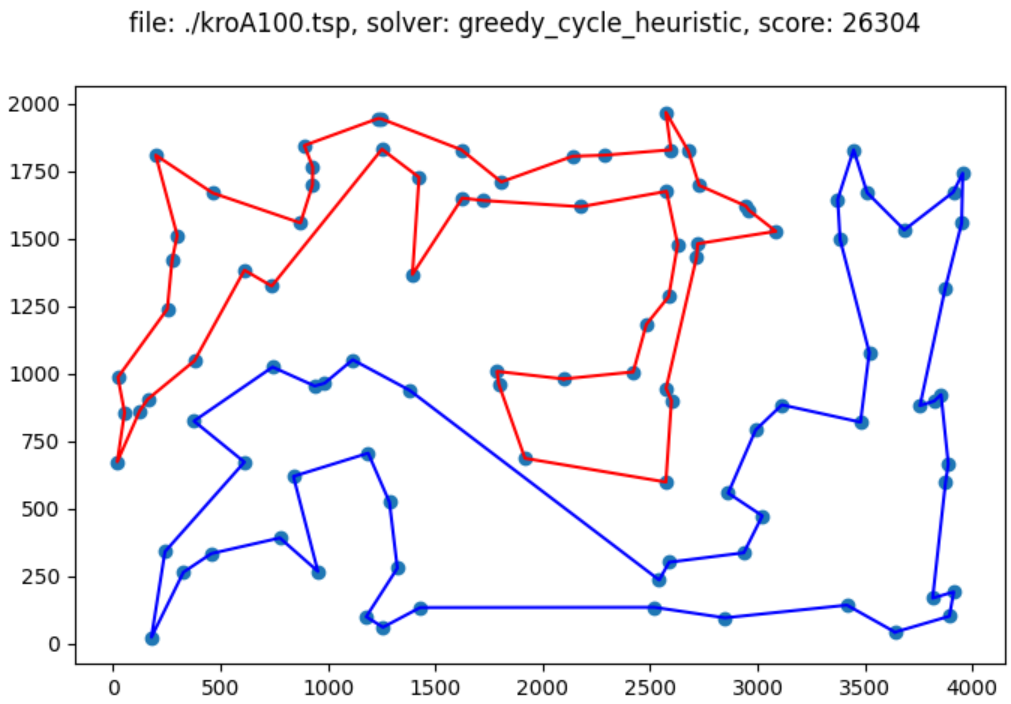
> Znajdź wierzchołek z *remaining_nodes*, który minimalizuje różnicę wynikającą z wstawienia go w cykl na różne pozycje
>
> Wstaw ten wierzchołek w cykl na najlepszą pozycję
>
> Usuń ten wierzchołek z *remaining_nodes*

Dopóki lista *remaining_nodes* nie jest pusta

Zwróć słownik zawierający listy wierzchołków dla obu cykli

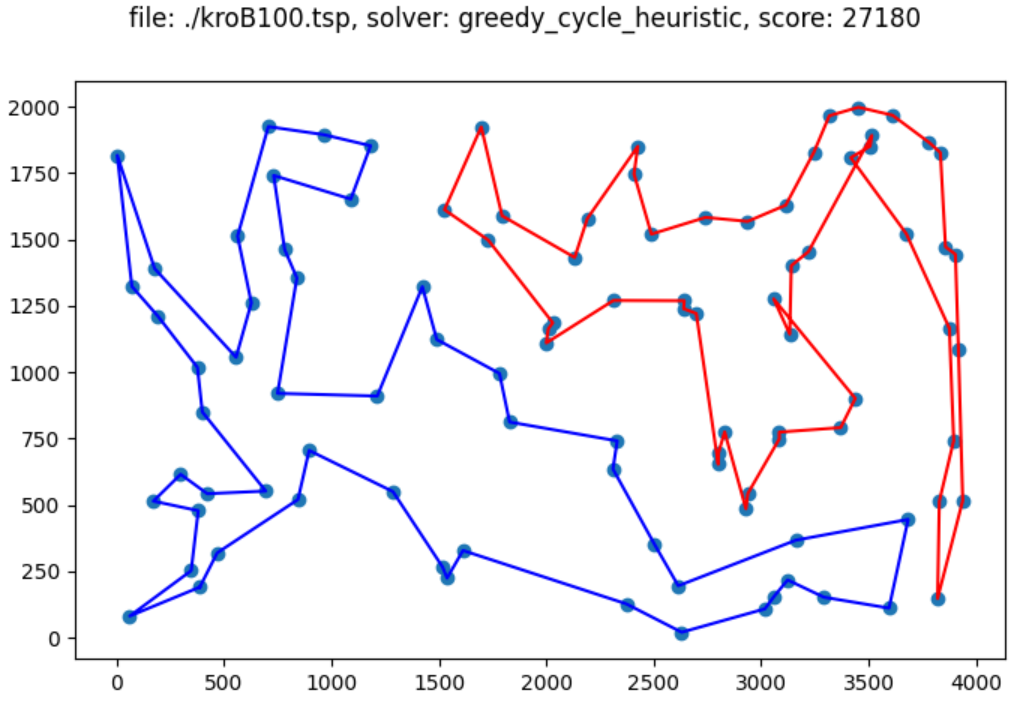
2.3 Wizualizacje

- Wynik dla kroA100:



Rys. 3. Obraz najlepszej ścieżki uzyskany dla heurystyki najbliższego sąsiada dla danych kroA100.

- Wynik dla kroB100:



Rys. 4. Obraz najlepszej ścieżki uzyskany dla heurystyki najbliższego sąsiada dla danych kroB100.

Heurystyka oparta na 2-żalu (regret heuristic)

3.1 Opis algorytmu

Regret cycle heuristics to rodzaj algorytmu heurystycznego, który ma na celu rozwiązanie problemu komiwojażera. Algorytm ten polega na wybraniu losowego punktu startowego i iteracyjnym dołączaniu kolejnych punktów do trasy. W przeciwieństwie do algorytmu najbliższego sąsiada, wybór kolejnych punktów nie jest oparty jedynie na odległości, ale na tak zwanym żalu, czyli różnicy między odległością danego punktu a dwoma najbliższymi punktami w trasy. Wybierany jest punkt, który ma największą wartość żalu, co powoduje, że algorytm bardziej złożony, ale często daje lepsze wyniki.

3.2 Pseudokod

Znajdź najbardziej oddalony wierzchołek od bieżącego, od którego zacznie się budowa drugiego cyklu

Inicjalizuj listę *remaining_nodes* zawierająca potencjalnych kandydatów do wydłużenia ścieżki

Inicjalizuj cykle, pierwszy zawierający wierzchołek wybrany, drugi najbardziej od niego oddalony

Powtarzaj dla każdego cyklu w słowniku cycles

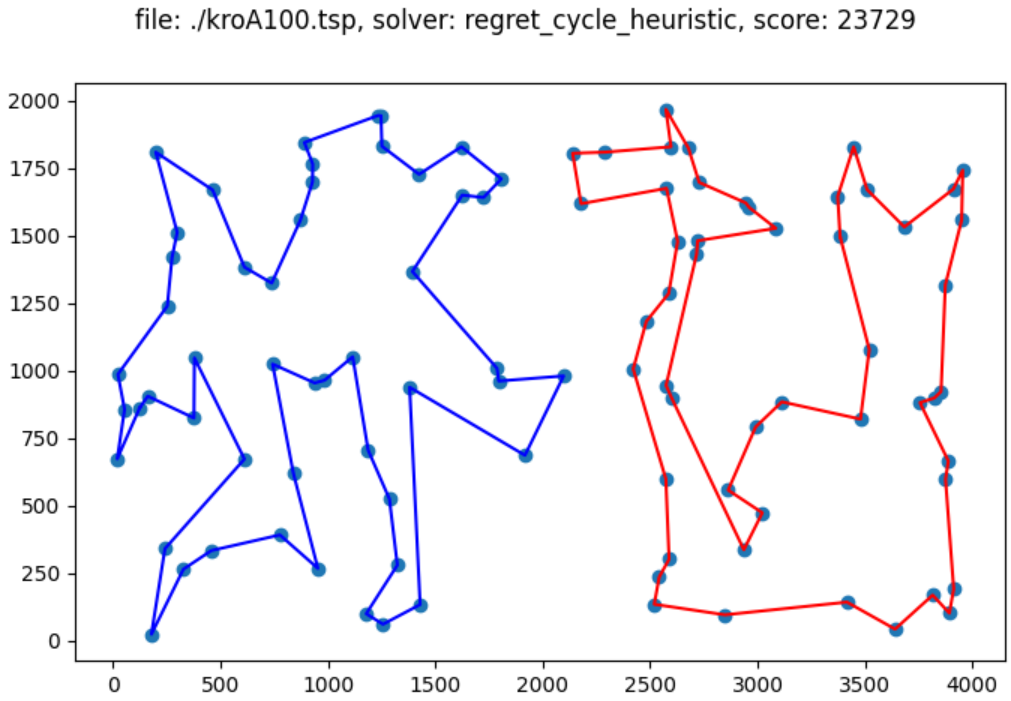
> Dla każdego cyklu obliczanie wartości punktacji dla każdego pozostałego wierzchołka, zgodnie z funkcją *score_diff*, która oblicza różnicę w koszcie między dodaniem nowego wierzchołka do cyklu a pozostawieniem cyklu bez wierzchołka, wykorzystując macierz odległości między wierzchołkami.
>
> Wybieranie wierzchołka, dla którego wartość punktacji jest najwyższa, uwzględniając wagę wprowadzoną przez wartość regret
>
> Wstawienie wybranego wierzchołka do odpowiedniego cyklu na odpowiedniej pozycji
>
> Usunięcie wierzchołka z listy pozostałych wierzchołków

Dopóki lista *remaining_nodes* nie jest pusta

Zwróć słownik zawierający listy wierzchołków dla obu cykli

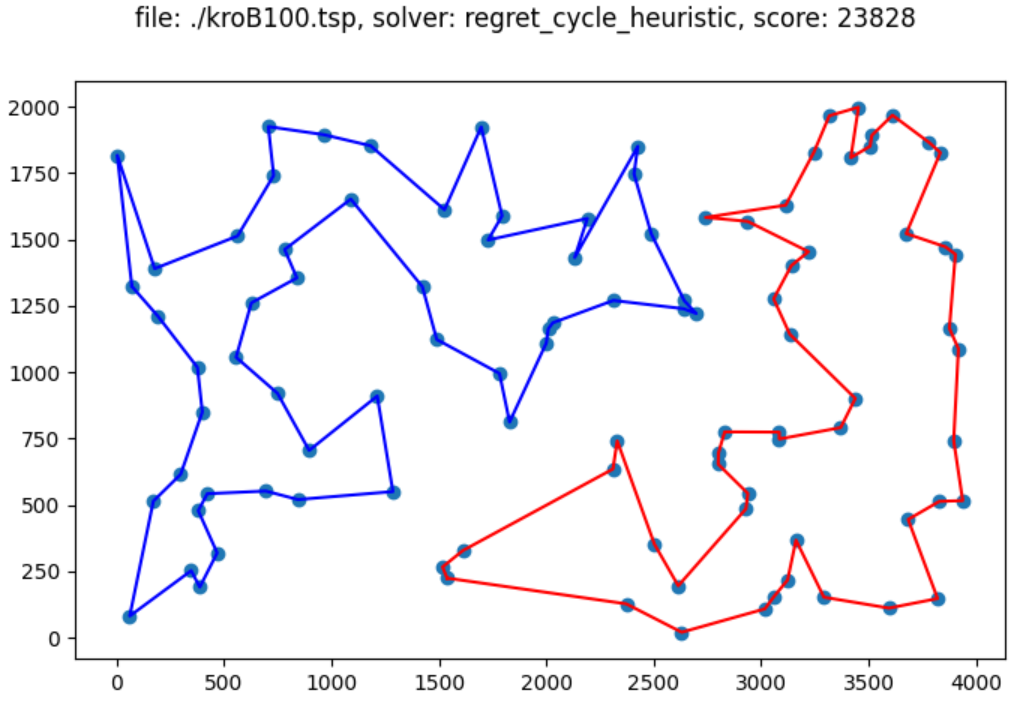
3.3 Wizualizacje

- Wynik dla kroA100:



Rys. 5. Obraz najlepszej ścieżki uzyskany dla heurystyki najbliższego sąsiada dla danych kroA100.

- Wynik dla kroB100:



Rys. 6. Obraz najlepszej ścieżki uzyskany dla heurystyki najbliższego sąsiada dla danych kroB100.

Podsumowanie

Porównanie wyników eksperymentu obliczeniowego

Algorithms	kroA100			kroB100		
	min	*mean*	*max*	*min*	*mean*	*max*
Nearest neighbour	27208	32496	35514	26342	32325	40264
Greedy cycle	26304	28708	29980	27180	28538	30197
Regret heuristic	23729	26573	32441	23828	27561	29793

Rys. 7. Wyniki eksperymentu

Wnioski

Zgodnie z oczekiwaniami, najlepsze wyniki, poza maksymalnym w kroA100, otrzymaliśmy dla algorytmu opartego na 2-żalu – docelowo ten algorytm miał optymalizować problem, co zostało osiągnięte. Wyniki dla tego algorytmu są zawsze lepsze od wyników w algorytmie najbliższego sąsiada oraz rozbudowy cyklu (oprócz wspomnianej wyżej wartości maksymalnej dla kroA100). Natomiast algorytm rozbudowy cyklu jest lepszy od algorytmy najbliższego sąsiada (oprócz wartości minimalnej dla kroB100).

Kod programu

https://github.com/imo/solution.py