Zadanie 3. Wykorzystanie ocen ruchów z poprzednich iteracji i ruchów kandydackich w lokalnym przeszukiwaniu

Oskar Kiliańczyk 151863 & Wojciech Kot 151879

1 Opis zadania

Celem eksperymentu jest poprawa efektywności czasowej lokalnego przeszukiwania w wersji stromej, wykorzystującego najlepsze sąsiedztwo z poprzedniego zadania. Porównujemy wersję bazową z dwiema modyfikacjami: uporządkowaną listą ruchów oraz ruchem kandydackim. Każdy algorytm uruchamiany jest 100 razy na każdej instancji, startując z losowych rozwiązań. Dla porównania uwzględniamy również najlepszą heurystykę konstrukcyjną z zadania 1.

2 Opisy algorytmów

3 Wyniki

3.1 Tabela wynikowa

Algorytm	Best	Avg	Worst	Best Time	Avg Time	Worst Time	Best Diff	Avg Diff
`split_paths_regret_TSP`	30426	32893	36854	0.0943059	0.105313	0.206992		
`traverse_steepest_edge`	35949	38818.8	41812	3.70818	4.1397	4.53379	325405	301537
`steepest_LM`	34643	38673	41570	1.02937	1.30407	1.92433	327271	301518
`steepest_kandydackie`	36656	39723.4	43659	0.889616	1.04703	1.90562	323577	298848

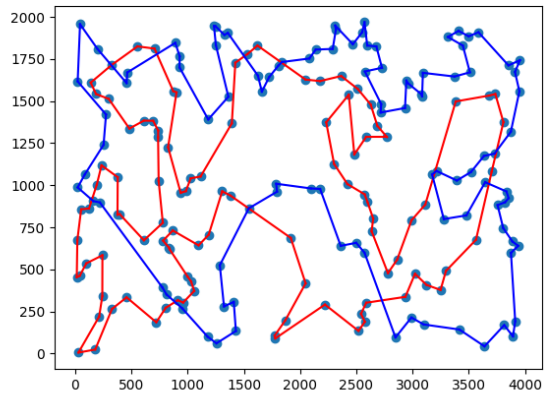
Tabela 1: Wyniki dla `kroA200` z naszym algorytmem

Algorytm	Best	Avg	Worst	Best Time	Avg Time	Worst Time	Best Diff	Avg Diff
`split_paths_regret_TSP`	31218	33102.7	36913	0.0939048	0.0999451	0.131889		
`traverse_steepest_edge`	36556	38791.3	41839	3.94789	4.60979	6.5536	329070	293891
`steepest_LM`	36309	38794.7	41938	0.97498	1.25639	1.55901	322191	293913
`steepest_kandydackie`	37372	39808.4	41720	0.861845	0.965179	1.08236	325756	292558

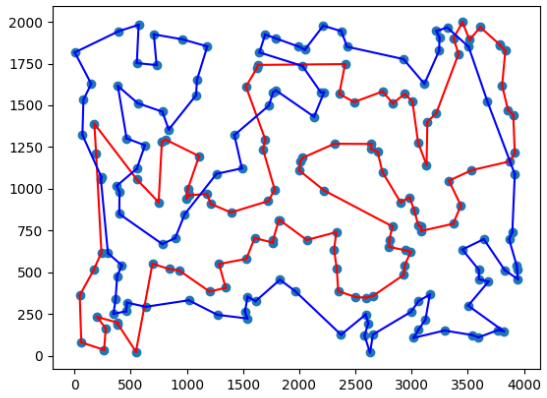
Tabela 2: Wyniki dla `kroB200` z naszym algorytmem

3.2 Wizualizacja wyników

3.2.1 Algorytm stromy, bazowy

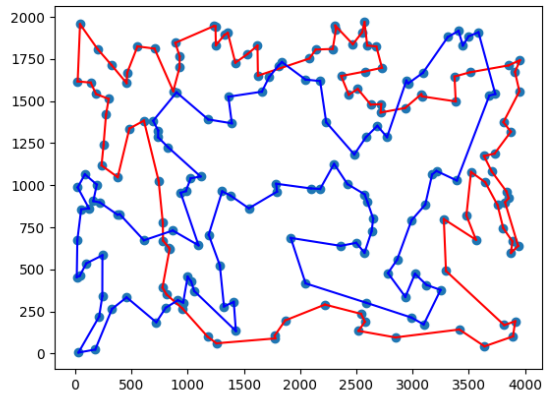


Rysunek 1: kroA200, losowy start

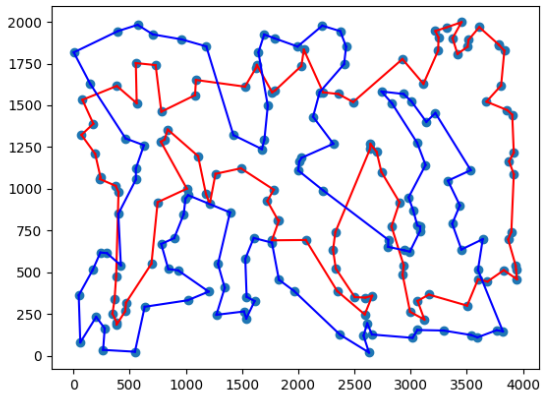


Rysunek 2: kroB200, losowy start

3.2.2 Algorytm stromy z listą ruchów

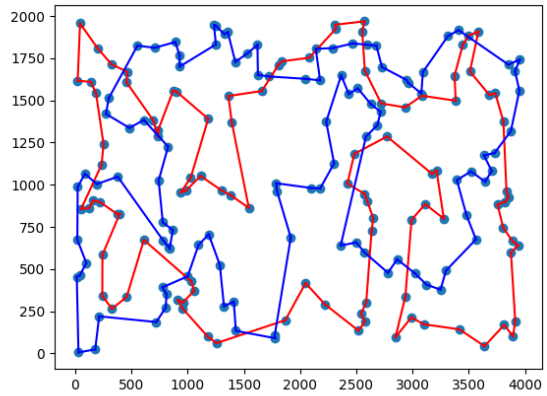


Rysunek 3: kroA200, losowy start

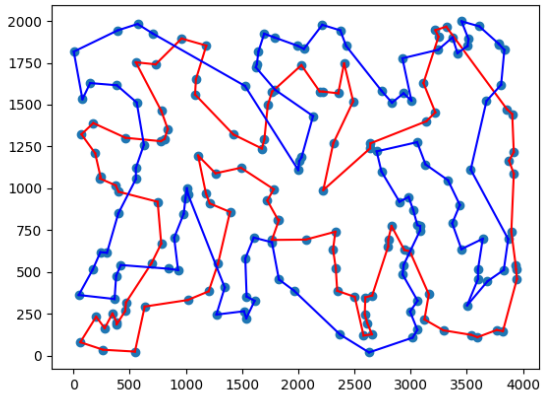


Rysunek 4: kroB200, losowy start

3.2.3 Algorytm stromy z mechanizmem ruchów kandydackich

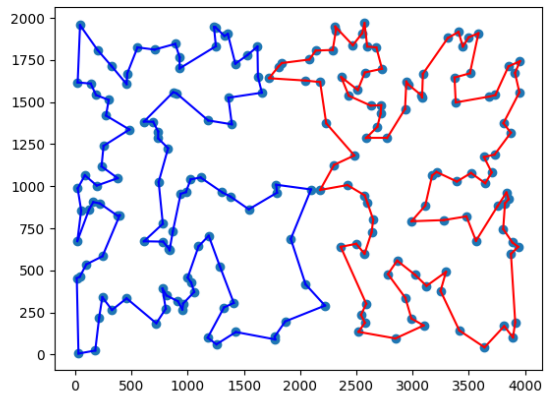


Rysunek 5: kroA200, losowy start

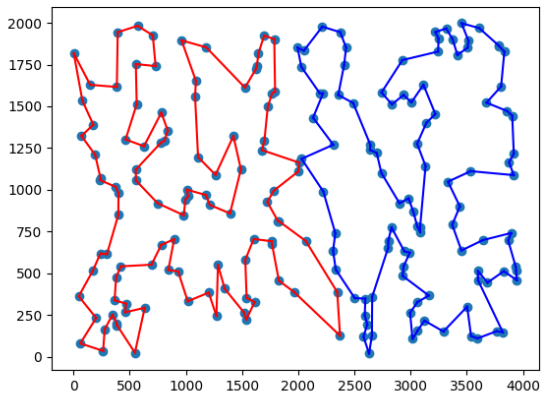


Rysunek 6: kroB200, losowy start

3.2.4 Heurystyka konstrukcyjna



Rysunek 7: kroA200, losowy start



Rysunek 8: kroB200, losowy start

4 Wnioski i analiza wyników

Na podstawie wyników można zauważyć, że algorytm stromy z listą ruchów oraz algorytm stromy z mechanizmem ruchów kandydackich są znacznie bardziej efektywne niż algorytm bazowy. W przypadku instancji kroA200 algorytm stromy z listą ruchów osiągnął lepszy wynik, natomiast w przypadku instancji kroB200 algorytm stromy z mechanizmem ruchów kandydackich okazał się lepszy. Niezależnie od instancji algorytm korzystający z ruchów kandydackich okazał się odrobinę bardziej efektywny czasowo. Wyniki naszej heurystyki konstrukcyjnej są wciąż najlepsze, jednak jest to przede wszystkim zasługa dobrego startowego podziału, czyli zastosowania swego rodzaju wiedzy dziedzinowej, a to sugeruje, że algorytmy lokalnego przeszukiwania mogą być użyte do poprawy wyników heurystyki konstrukcyjnej, ale nie jako osobna metoda znajdowania optimum rozpoczynając z rozwiązań losowych. Poprzednie zadanie pokazało, że algorytmy lokalnego przeszukiwania są bardziej efektywne w przypadku, gdy startujemy z rozwiązań konstrukcyjnych.

5 Link do repozytorium

Kod źródłowy w repozytorium GitHub dostępny pod linkiem:
Repozytorium.