Optymalizacja tras dostaw z użyciem algorytmu genetycznego

1. Wyniki testowe

Wstęp

Przeprowadzono dwa typy testów z zamiarem oceny skuteczności GA oraz porównania go do innych algorytmów do rozwiązywania TSP:

- Testy na losowych instancjach – Dla rozmiarów problemu n ∈ {5, 10, 20, 50, 100} wygenerowano po 10 losowych instancji. W tych testach nie jest znana optymalna długości trasy, więc porównywane są tylko czas działania i długość znalezionej trasy pomiędzy algorytmami.
- Testy na instancjach z TSPLIB – Przeprowadzono testy na czterech instancjach z TSPLIB: n ∈ {11, 29, 52, 100, 442}. W tym przypadku biblioteka udostępnia prawidłowe rozwiązanie, zatem można porównać odległość rozwiązania od optimum.

GA porównano z algorytmami Brute Force (dla n do maksymalnie 10), zachłannym (Nearest Neighbour), losowym, oraz 2-OPT.

Surowe wyniki testów są zapisane w plikach test_results.csv (dla losowych instancji) oraz tsplib_test_results.csv (dla instancji z tsplib), a ich wizualizacje w pliku route_optimizer.ipynb.

Średnia znaleziona długość trasy - losowe instancje

Ilość miast	GA	GA + 2-opt	NN	NN + 2-opt	Random	Random + 2-opt
5	2413.54	2413.54	2532.31	2463.97	3029.86	2999.67
10	2834.88	2834.88	3064.90	2916.80	5025.41	2999.67
20	4115.63	4115.63	4397.18	4267.32	11254.18	4578.34
50	6221.27	6175.51	7057.40	6409.56	26706.88	6541.67
100	8467.22	8518.56	9876.50	8622.70	52471.06	9054.61

Znaleziona długość trasy - instancje z TSPLIB

Ilość miast	Optimal	GA	GA + 2-opt	NN	NN + 2-opt	Random	Random + 2-opt
11	3357.82	3423.65	3423.65	3797.58	3471.94	6056.86	3423.65
29	9074.14	9280.95	9280.95	10211.18	9770.09	25715.19	10240.66
52	7544.36	8148.70	8148.70	8980.91	8384.18	29303.60	9038.30
100	21285.44	22361.12	22303.25	26856.38	23641.41	175162.07	24951.32
442	50783.54	63552.40	58471.90	61984.04	54162.12	759484.84	58323.50

2. Opis danych wejściowych

Dane wejściowe algorytm uzyskuje na dwa sposoby:

- generowane losowo instancje problemu funkcją `generate_random_instance` - funkcja ta zwraca listę punktów 2D. Każdy punkt reprezentuje współrzędne jednego miasta.

- dane z TSPLIB. Wczytywanie ich odbywa się za pomocą funkcji `load_tsplib_instance.` Funkcja ta przyjmuje ścieżkę do pliku .tsp, gdzie zapisane są instancje problemu w formacie zgodnym z tą biblioteką, a następnie zwraca listę punktów 2D reprezentujących współrzędne miast.

3. Analiza wyników i możliwe usprawnienia

Analiza wyników

Zaimplementowany algorytm genetyczny okazał się skuteczną metodą rozwiązywania problemu komiwojażera. W ogólnych przypadkach regularnie przewyższał jakością rozwiązania zarówno prosty algorytm zachłanny Nearest Neighbour, jak i 2-opt. Najlepsze rezultaty osiągnięto, łącząc GA z 2-opt – taka hybryda często dawała wyniki bliskie znanego optimum. Warto jednak zaznaczyć, że w niektórych przypadkach (np. instancja pcb442) prostsze algorytmy potrafiły znaleźć lepsze rozwiązanie niż GA.

Główną wadą GA jest zdecydowanie czas działania – złożoność obliczeniowa algorytmu rośnie znacznie szybciej niż w przypadku NN czy 2-opt. Dla instancji ze 100 miastami, GA potrzebował średnio ok. 35 sekund, podczas gdy NN średnio kończył w 0.02s, a 2-opt (z losowej permutacji) – średnio w ok. 1.7s. To pokazuje, że cena za lepsze rozwiązanie jest znaczna.

Model wykazywał dobrą zbieżność, a zastosowany mechanizm wczesnego stopu (brak poprawy przez 100 pokoleń) pozwalał skrócić czas działania, gdy rozwiązanie było już wystarczająco dobre. Należy jednak zauważyć, że dla większych instancji (powyżej 100 miast) limit 3000 pokoleń może być niewystarczający.

Podsumowując, stworzony model dobrze sprawdza się jako metoda metaheurystyczna, szczególnie w połączeniu z lokalnym ulepszaniem. Przy większych instancjach należy jednak uważać na czas wykonania.

Możliwe usprawnienia

- elitaryzm - Obecnie algorytm nie zachowuje najlepszych osobników w kolejnych pokoleniach. Wprowadzenie elitaryzmu w postaci kopiowania kilku najlepszych rozwiązań do następnej generacji bez zmian mogłoby zapobiec degradacji wyników i przyspieszyć konwergencję.
- rozmiar populacji - Aktualnie populacja ustalana jest jako 5 * liczba_miast. Ten współczynnik równy 5 został przeze mnie przyjęty arbitralnie i może być zbyt wysoki, co wpływa na czas działania algorytmu. Wprowadzenie tej wartości jako parametru w grid searchu (np. testowanie zakresu 2x–5x) pozwoliłoby lepiej dobrać kompromis między jakością wyników a czasem działania.
- lokalna optymalizacja - wyniki testów pokazują, że połączenie GA z 2-opt daje bardzo dobre wyniki. Jednak w testach stosowałem 2-opt jedynie jako optymalizację już znalezionego przez GA rozwiązania. Możliwym usprawnieniem byłoby zastosowanie 2-opt w trakcie działania algorytmu, np. po każdym krzyżowaniu lub co kilka generacji, dla wybranej części populacji. Taka hybryda genetyczno-lokalna może znacząco poprawić jakość populacji