Algorytmy Metaheurystyczne - LocalSearch

Piotr Maciejończyk

21 listopada 2023

1 Cel listy laboratoryjnej

Przedmiotem rozważań tej listy było sprawdzenie w praktyce działania heurystyki Local Search w odniesieniu do euklidesowego problemu komiwojażera. W tym celu należało zastosować tę heurystykę dla cykli wygenerowanych za pomocą minimalnego drzewa rozpinającego oraz dla losowych permutacji. Dodatkowo, aby przyspieszyć obliczenia, trzeba było odpowiednio tę heurystykę zmodyfikować.

Jednym ze sposobów na generowanie otoczenia dla problemu komiwojażera, jest tzw. otoczenie *invert*, gdzie ruch dany jest funkcją $invert(n, i, j)$, która polega na odwróceniu kolejności wierzchołków od i-tego do j-tego. **Zatem, jak duże jest to otoczenie?**

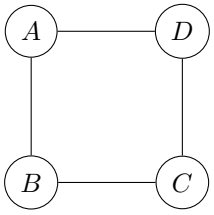
Chcemy policzyć wszystkie różne 'unikalne' dostępne ruchy dla pewnej permutacji π dla n wierzchołków. Nazwijmy zbiór wszystkich takich unikalnych ruchów jako S, który możemy opisać jako $S = \{invert(\pi, i, j) : 1 \leq i < j \leq n\}$. Moc tego zbioru to wielkość otoczenia:

$$|S| = (n-1) + (n-2) + (n-3) + ... + 1 = \frac{(1+(n-1))}{2} \cdot (n-1) = \frac{n(n-1)}{2}$$

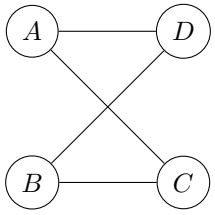
Trzeba jeszcze tylko pamiętać o tym, że $invert(\pi, 1, n)$ zmienia jedynie kierunek cyklu, zatem sam w sobie nie stanowi osobnego, unikalnego ruchu. Zatem wielkość otoczenia można opisać za pomocą wzoru:

$$\frac{n(n-1)}{2} - 1$$

Stosując otoczenie *invert* można zauważyć, że algorytm w kontekście grafów euklidesowych zawsze zwróci rozwiązanie bez krzyżujących się krawędzi. Dzieje się tak, ponieważ w metryce euklidesowej pętle są nieoptymalne i generują większe wagi, niż niekrzyżujące się krawędzie. Przyczyną tego jest fakt, że w tej metryce zachodzi nierówność trójkąta. Przykładowo:



Rysunek 1: Brak pętli



Rysunek 2: Pętla

Na rysunkach na poprzedniej stronie narysowałem dwa cykle - jeden z pętlą, drugi bez pętli. Obliczmy zatem ich wagi, wiedząc, że znajdujemy się w przestrzeni euklidesowej:

1. Rysunek 1. (brak pętli):

$$w_1 = d(A, D) + d(D, C) + d(C, B) + d(B, A)$$

2. Rysunek 2. (pętla):

$$w_2 = d(A, C) + d(C, B) + d(B, D) + d(D, A)$$

Jak można zauważyć na rysunkach, z nierówności trójkąta mamy: $d(A, C) \geq d(A, B)$ oraz $d(B, D) \geq d(C, D)$. Wniosek: $w_2 \geq w_1$, zatem pętle niepotrzebnie zwiększają wagę szukanego optymalnego cyklu.

Jednakże, w przypadku, w którym stosujemy zaokrąglenia (czyli dla naszych zadań), nierówność trójkąta nie musi zachodzić, przez co krawędzie w znalezionym optymalnym cyklu będą mogły się krzyżować. Uargumentuję to bazując na poprzednich rysunkach:

$$d(A, D) = d(D, C) = d(C, B) = d(B, A) = 1$$
$$d(A, C) = \sqrt{2} \approx 1$$

Zatem możliwe jest znaleźć cykl z Rysunku 2., ponieważ wszystkie jego krawędzie mają przypisaną taką samą wagę.

2 *LocalSearch* dla cykli na podstawie MST

Jedno z zadań polegało na zastosowaniu heurystyki *LocalSearch* dla cykli stworzonych na podstawie minimalnych drzew rozpinających. Dla każdego przykładu wykonałem $\sqrt{n}$ (gdzie n - l. wierzchołków) razy algorytm:

1. Wylosuj wierzchołek.
2. Skonstruuj cykl zaczynając przegląd drzewa od wylosowanego wierzchołka.
3. Zastosuj algorytm *LocalSearch* do tak uzyskanego cyklu.

Moje wyniki zamieściłem w Tabeli 1. poniżej:

Filename	MST weight	Average solution weight	Average number of steps	The best solution weight	Iterations
xqf131	474	590.833	28.6667	577	12
xqg237	897	1086.5	36.25	1074	16
pma343	1179	1464.95	60.9474	1441	19
pka379	1151	1397.9	82.85	1389	20
bcl380	1444	1714.85	56.5	1700	20
pbl395	1124	1398.5	70.35	1385	20
pbk411	1180	1431.14	83.8095	1418	21
pbn423	1201	1489.57	71.7619	1466	21
pbm436	1269	1545	85.3333	1528	21
xql662	2240	2685.46	106.423	2651	26
xit1083	3253	3822.21	197.97	3792	33
icw1483	4015	4855.9	236.051	4818	39
djc1785	5541	6776.74	339.209	6725	43
dcb2086	5950	7322.67	336.087	7279	46
pds2566	6956	8541.61	403.02	8483	51

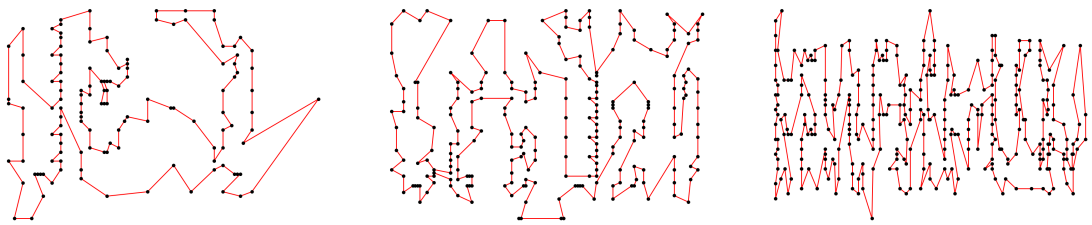
Tabela 1: Wyniki dla zadania 1.

Oprócz Tabeli 1. umieściłem poniżej grafy dla znalezionych przeze mnie cykli. Co pierwsze rzuca się w oczy, to fakt, że trudno jest się doszukać w nich krawędzi skrzyżowanych, czego przyczyną jest zastosowanie algorytmu *LocalSearch* wraz z generowaniem otoczenia za pomocą funkcji *invert*. Przekłada się to na relatywnie małe wagi otrzymanych cykli.

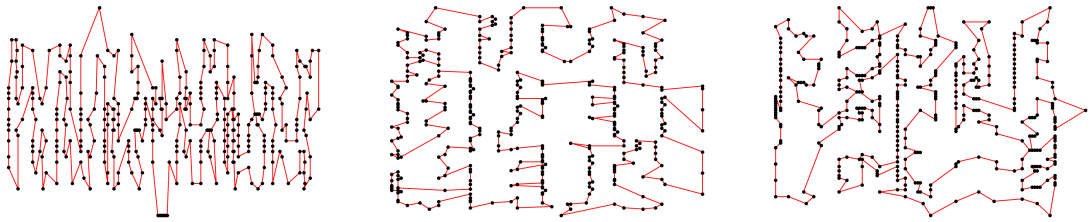
Nie oznacza to jednak, że krawędzie w ogóle się ze sobą nie krzyżują - wręcz przeciwnie. Jako, że pracowaliśmy w przestrzeni euklidesowej, gdzie wszystkie odległości były zaokrąglane do liczb całkowitych, to w pewnych miejscach nie musi zachodzić nierówność trójkąta.

W Tabeli 1. można również zauważyć, że dla każdego losowo wybranego wierzchołka początkowego otrzymujemy podobne wagi cykli, na co wskazuje podobieństwo średniej wagi otrzymanego rozwiązania do wagi najlepszego znalezionego rozwiązania dla każdego z przykładów.

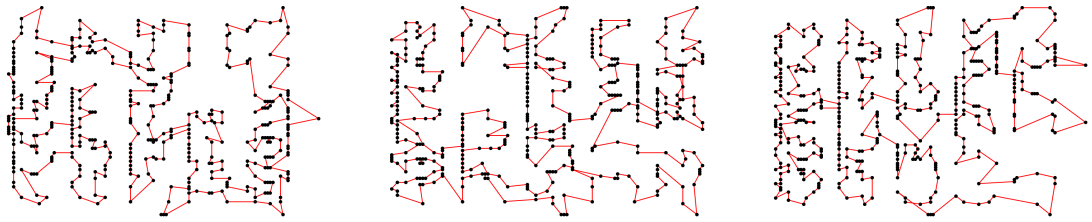
W porównaniu do wyników z Tabeli 2. oraz z Tabeli 3., program ten zwrócił najlepsze rezultaty. Średnie (oraz najlepsze) wagi otrzymanych rozwiązań są najmniejsze, a liczba potrzebnych kroków do wykonania algorytmu jest znacząco niższa.



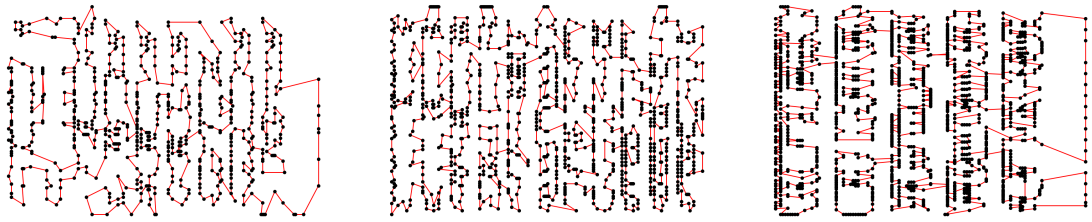
(a) xqf131 (b) xqg237 (c) pma343



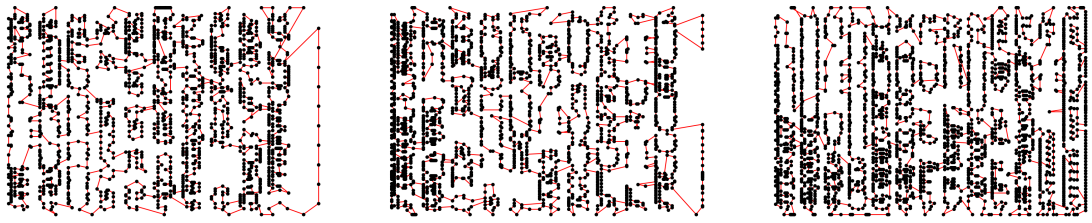
(d) pka379 (e) bcl380 (f) pbl395



(g) pbk411 (h) pbn423 (i) pbm436



(j) xql662 (k) xit1083 (l) icw1483



(m) djc1785 (n) dcb2086 (o) pds2566

Rysunek 3: Utworzone cykle na podstawie drzew MST

3 *LocalSearch* dla losowych cykli

Filename	Average solution weight	Average number of steps	The best solution weight	Iterations
xqf131	648.527	132.641	611	131
xqg237	1259.6	260.253	1200	237
pma343	1769.31	403.408	1685	343
pka379	1731.4	448.625	1654	379
bcl380	1927.12	448.305	1822	380
pbl395	1529.61	458.397	1461	395
pbk411	1617.68	483.998	1553	411
pbn423	1659.34	495.492	1575	423
pbm436	1720.67	513.872	1653	436
xql662	3028.49	812.048	2905	662
xit1083	4271.19	1386.51	4173	100
icw1483	5122.12	1930.37	5010	100
djc1785	7129.9	2348.59	6997	100
dcb2086	7722.32	2800.4	7543	100
pds2566	9011.12	3484.93	8862	100

Tabela 2: Wyniki dla zadania 2.

W porównaniu do wyników otrzymanych w poprzednim zadaniu średnia liczba kroków algorytmu znacząco wzrosła. Wynik ten nie powinien dziwić, ponieważ dla losowo wygenerowanych cykli potrzebne jest więcej "poprawek", niż dla cykli powstałych na bazie drzew MST, które znajdują się bliżej optymalnego rozwiązania.

Wagi znalezionych najlepszych rozwiązań dla losowych cykli są większe niż w poprzednim zadaniu, lecz nadal są one zbliżone do siebie. Wynik może i jest podobny, lecz generowanie optymalnego rozwiązania dla losowych cykli okazuje się być bardzo czasochłonne.

4 Zmodyfikowany *LocalSearch*

Filename	Average solution weight	Average number of steps	The best solution weight	Iterations
xqf131	1037.6	195.153	825	131
xqg237	2184.31	367.392	1798	237
pma343	3051.81	551.825	2562	343
pka379	3009.87	606.29	2512	379
bcl380	3807.59	603.834	2991	380
pbl395	2954.83	623.843	2414	395
pbk411	3169.51	649.479	2582	411
pbn423	3241.56	664.508	2553	423
pbm436	3359.33	696.376	2777	436
xql662	6041.56	1087.65	4996	662
xit1083	9058.17	1826.14	7637	100
icw1483	11272.8	2518.88	9531	100
djc1785	15163.9	3097.98	13157	100
dcb2086	17481.5	3638.47	14881	100
pds2566	20870.6	4511.04	18428	100

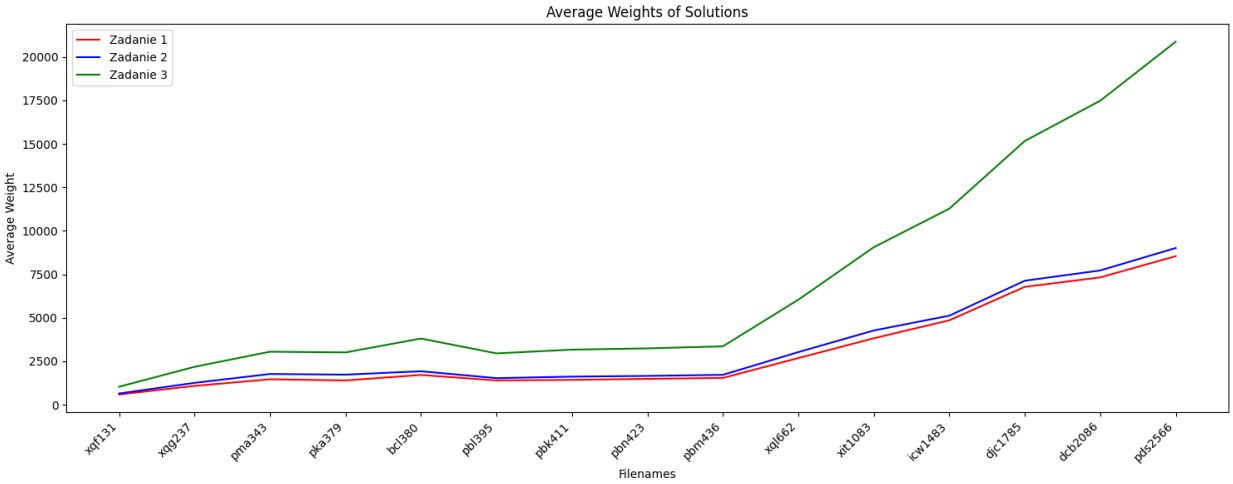
Tabela 3: Wyniki dla zadania 3.

Wyniki w Tabeli 3. zostały utworzone na podstawie rezultatów zwróconych przez zmodyfikowany algorytm *LocalSearch*. Modyfikacja ta polegała na ograniczeniu przeglądanego sąsiedztwa, poprzez wybranie najlepszego sąsiada jedynie z n losowo wybranych.

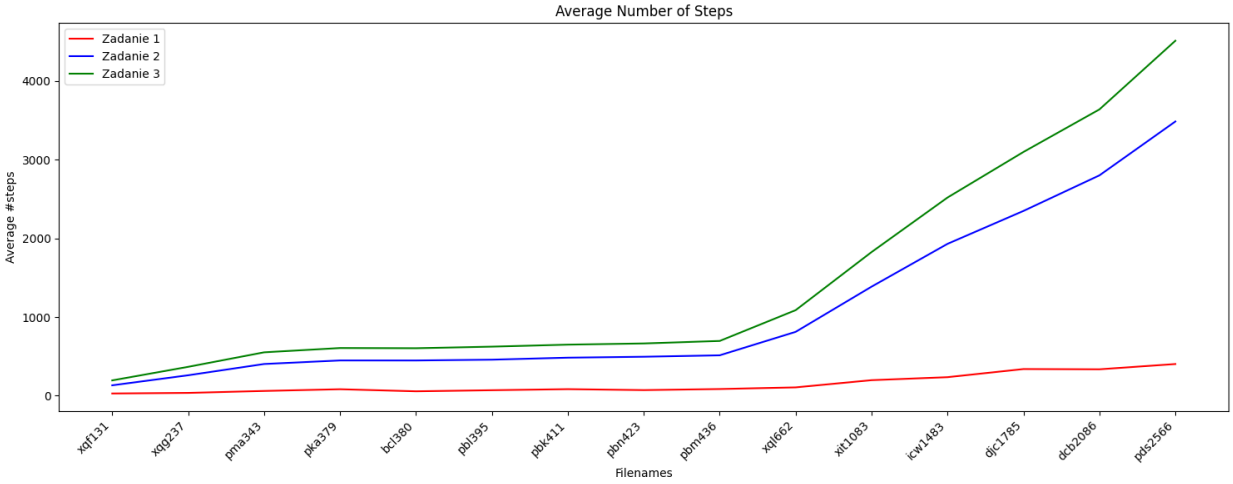
Sam program generujący wyniki znacząco szybciej znajdował rozwiązania, niż program korzystający z niezmodyfikowanego algorytmu. Jednakże, tak wprowadzone ograniczenia mocno wpłynęły na rezultaty. Średnia waga znalezionego rozwiązania jest dla prawie każdego przykładu ponad 2 razy większa w porównaniu do wyników z poprzedniego zadania. Dodatkowo, średnia liczba potrzebnych kroków wzrosła, tak samo jak wagi najlepszych znalezionych rozwiązania (dla ostatniego przykładu optymalne znalezione rozwiązania ma wagę ponad 2-krotnie większą!).

Zatem wyniki zwracane są bardzo szybko, lecz dzieje się to kosztem odchylenia od optymalnego rozwiązania.

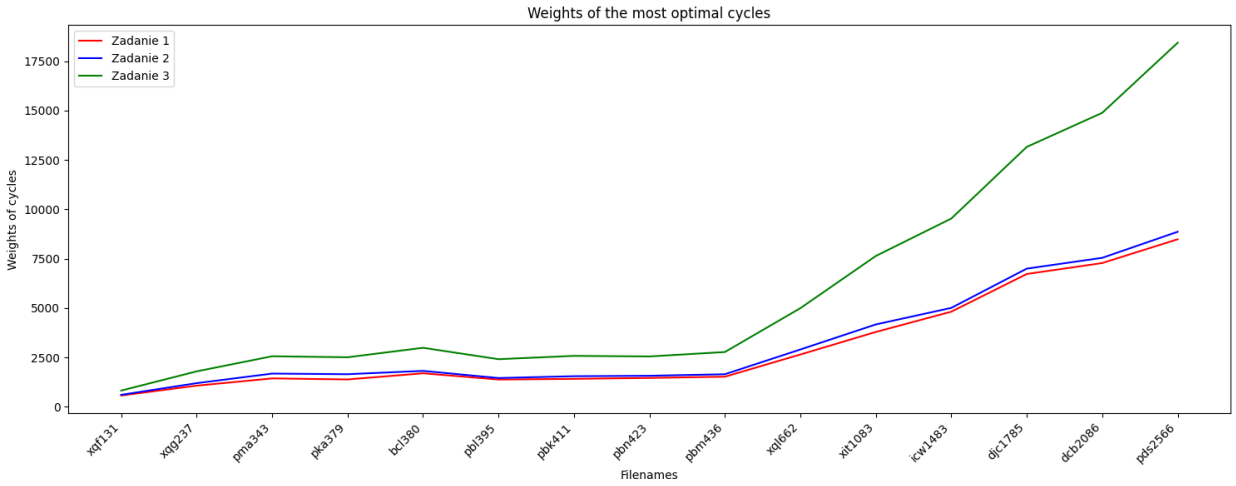
5 Porównanie wyników



Rysunek 4: Średnie wagi otrzymanych rozwiązań



Rysunek 5: Średnia liczba kroków algorytmu



Rysunek 6: Wagi najbardziej optymalnych rozwiązań

Statystyki dla cykli tworzonych na podstawie drzew MST pokazują, że jest to bardzo optymalny sposób wyznaczania takich cykli. Wyniki otrzymane za pomocą algorytmu *LocalSearch* dla losowych cykli są do nich bardzo mocno podobne, lecz aby je uzyskać potrzebna jest większa liczba kroków i sam sposób jest czasochłonny. Zmodyfikowany *LocalSearch* znajduje rozwiązania szybciej, lecz dzieje się to kosztem jakości tych rozwiązań - ich wagi są duże w porównaniu do poprzednich metod.