Implementacja i porównanie algorytmów rozwiązujących problem komiwojażera (TSP)

Marta Kurzych

30 stycznia 2026

Spis treści

1 Wstęp

Problem komiwojażera (ang. *Traveling Salesman Problem*, TSP) jest klasycznym problemem optymalizacji kombinatorycznej należącym do klasy NP-trudnych. Polega na znalezieniu najkrótszej drogi odwiedzającej każde z n miast dokładnie raz i powracającej do punktu startowego.

1.1 Cel projektu

Celem projektu jest implementacja i porównanie pięciu algorytmów rozwiązujących TSP:

- Algorytm siłowy (Brute Force)
- Algorytm najbliższego sąsiada (Nearest Neighbour)
- Algorytm Christofidesa
- Algorytm genetyczny (Genetic Algorithm)
- Algorytm mrówkowy (Ant Colony Optimization)

1.2 Definicja problemu

Dany jest graf pełny $G = (V, E)$, gdzie $V = \{v_1, v_2, \dots, v_n\}$ jest zbiorem wierzchołków, $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ zbiorem krawędzi, oraz $w : E \to \mathbb{R}^+$ funkcją wagi. Należy znaleźć cykl Hamiltona minimalizujący:

$$\sum_{i=1}^{n-1} w(v_{\pi(i)}, v_{\pi(i+1)}) + w(v_{\pi(n)}, v_{\pi(1)}) \tag{1}$$

gdzie π jest permutacją zbioru V.

2 Generator grafów

Generator tworzy losowe grafy pełne spełniające nierówność trójkąta:

$$\forall_{u,v,w \in V} : w(u, v) \leq w(u, w) + w(w, v) \tag{2}$$

Algorytm wykorzystuje mechanizm backtrackingu z limitami prób (50 prób na krawędź, 10 prób na wierzchołek). Dla każdej dodawanej krawędzi (u, v) losowana jest waga $w \in [1, 10]$ spełniająca nierówność trójkąta względem wszystkich istniejących trójek wierzchołków.

3 Zaimplementowane algorytmy

3.1 Algorytm siłowy (Brute Force)

Algorytm siłowy jest metodą dokładną sprawdzającą wszystkie permutacje wierzchołków.

3.1.1 Analiza złożoności

- **Złożoność czasowa:** $O(n!)$
- **Dokładność:** Gwarantuje znalezienie rozwiązania optymalnego

Ze względu na złożoność $O(n!)$ algorytm jest praktyczny jedynie dla $n < 10$.

3.2 Algorytm najbliższego sąsiada (Nearest Neighbour)

Algorytm zachłanny wybierający w każdym kroku najbliższe nieodwiedzone miasto.

3.2.1 Analiza złożoności

- **Złożoność czasowa:** $O(n^2)$
- **Dokładność:** Brak gwarancji optymalności; współczynnik aproksymacji może być arbitralnie duży

3.3 Algorytm Christofidesa

Algorytm aproksymacyjny z gwarancją jakości dla metrycznego TSP. Etapy algorytmu:

1. Konstrukcja minimalnego drzewa rozpinającego T
2. Identyfikacja wierzchołków nieparzystego stopnia O w T
3. Znalezienie minimalnego skojarzenia doskonałego M w podgrafie indukowanym przez O
4. Konstrukcja mulitgrafu $H = T \cup M$
5. Znalezienie cyklu Eulera w H
6. Przekształcenie cyklu Eulera w cykl Hamiltona poprzez eliminację powtórzeń

3.3.1 Analiza złożoności

- **Złożoność czasowa:** $O(n^3)$
- **Dokładność:** Współczynnik aproksymacji ≤ 1.5 dla metrycznego TSP

3.4 Algorytm genetyczny

Algorytm metaheurystyczny wykorzystujący mechanizmy ewolucji. Operuje na populacji permutacji z funkcją fitness $f(\pi) = \frac{1}{cost(\pi)+\epsilon}$, gdzie $\epsilon = 10^{-6}$.

3.4.1 Parametry implementacji

Liczba pokoleń: 1000, rozmiar populacji: 200, liczba rodziców: 20, selekcja turniejowa, krzyżowanie jednopunktowe, mutacja swap (10%), elityzm: 5 osobników.

3.4.2 Analiza złożoności

- **Dokładność:** Heurystyczna, brak gwarancji optymalności

3.5 Algorytm mrówkowy (Ant Colony Optimization)

Algorytm metaheurystyczny inspirowany zachowaniem kolonii mrówek. Wykorzystuje ślady feromonowe τ_{ij} i informację heurystyczną $\eta_{ij} = \frac{1}{w(i,j)}$.

Prawdopodobieństwo wyboru krawędzi:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha \cdot [\eta_{il}]^\beta} \tag{3}$$

Aktualizacja feromonów: $\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$

3.5.1 Parametry implementacji

Liczba mrówek: 10, iteracje: 100, $\alpha = 1.0$, $\beta = 2.0$, $\rho = 0.5$, $Q = 100$.

3.5.2 Analiza złożoności

- **Dokładność:** Heurystyczna

4 Metodologia testowania

4.1 Środowisko testowe

Implementacja w języku Python 3.x z wykorzystaniem bibliotek: `networkx` (operacje na grafach), `pygad` (algorytm genetyczny), `acopy` (algorytm mrówkowy).

4.2 Plan testów

Dla każdego rozmiaru grafu $n \in \{5, 6, 7, 8, 9, 10, 12, 13, 15, 17, 18, 20, 22, 23, 25, 27, 28, 30\}$ wygenerowano 100 losowych instancji spełniających nierówność trójkąta. Zmierzono czas wykonania i długość znalezionej ścieżki.

5 Wyniki eksperymentów

5.1 Porównanie czasów wykonania

Węzły	**Brute Force**	**Nearest N.**	**Christofides**	**Genetic**	**Ant Colony**
5	0.000174	0.000016	0.000359	4.581518	0.010572
6	0.001119	0.000018	0.000371	6.633075	0.014674
7	0.008909	0.000022	0.000482	5.769299	0.019273
8	0.080014	0.000028	0.000575	7.610657	0.024255
9	0.809375	0.000031	0.000686	6.812918	0.030495
10	N/A	0.000034	0.000630	8.057302	0.036621
12	N/A	0.000042	0.000786	8.311062	0.051565
13	N/A	0.000047	0.000921	7.859096	0.059277
15	N/A	0.000057	0.001072	8.485046	0.079966
17	N/A	0.000074	0.001274	9.375810	0.104952
18	N/A	0.000075	0.001309	9.672899	0.113816
20	N/A	0.000087	0.001631	10.454814	0.142602
22	N/A	0.000100	0.001908	11.024130	0.174262
23	N/A	0.000108	0.001996	10.861792	0.191916
25	N/A	0.000129	0.002339	11.474389	0.231384
27	N/A	0.000143	0.002729	11.940552	0.273949
28	N/A	0.000155	0.002912	12.557491	0.298931
30	N/A	0.000170	0.003331	13.133853	0.347796

Tabela 1: Średni czas wykonania w sekundach (100 instancji na rozmiar)

5.2 Porównanie jakości rozwiązań

Węzły	Brute Force	Nearest N.	Christofides	Genetic	Ant Colony
5	24.24	25.17	24.61	24.74	24.26
6	27.44	28.98	28.36	27.91	27.52
7	30.21	32.40	31.44	32.07	30.27
8	33.86	36.45	35.60	34.60	33.94
9	35.91	39.34	38.10	38.28	35.93
10	N/A	43.55	42.51	41.55	39.82
12	N/A	49.82	48.57	48.17	45.02
13	N/A	52.56	51.34	50.96	47.30
15	N/A	60.27	59.50	59.12	54.26
17	N/A	66.23	65.88	65.66	59.75
18	N/A	69.62	69.14	68.62	63.10
20	N/A	76.22	76.09	75.19	69.24
22	N/A	82.62	83.19	82.39	75.72
23	N/A	87.06	86.99	87.09	79.10
25	N/A	92.02	91.52	92.02	84.23
27	N/A	99.17	100.64	101.26	92.38
28	N/A	104.17	104.81	106.01	97.31
30	N/A	110.21	112.01	112.09	104.32

Tabela 2: Średnia długość znalezionej ścieżki

5.3 Współczynnik aproksymacji

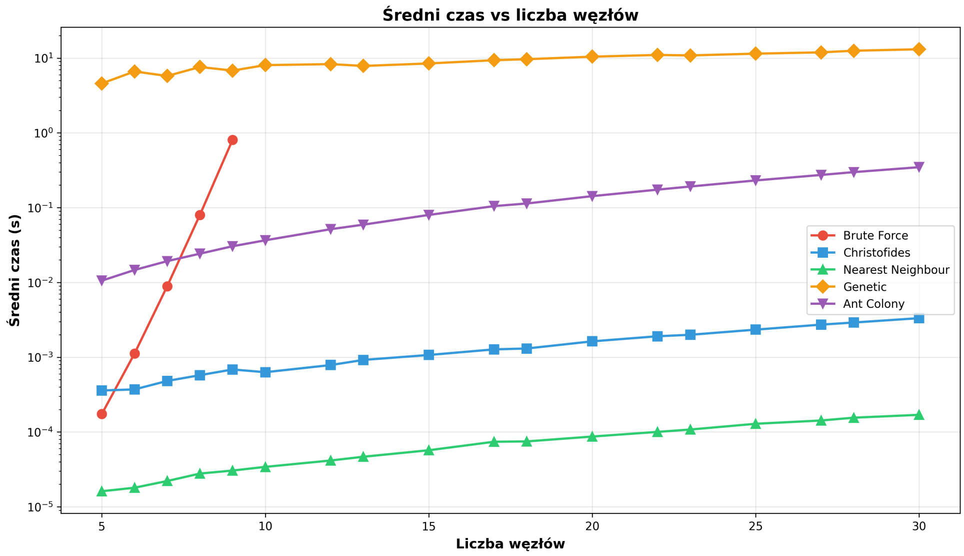
Dla małych instancji ($n < 10$) obliczono odchylenie od rozwiązania optymalnego:

Węzły	Nearest N.	Christofides	Genetic	Ant Colony
5	4.12%	1.51%	2.05%	0.07%
6	5.75%	3.19%	1.74%	0.39%
7	7.55%	4.02%	6.41%	0.23%
8	7.90%	5.06%	2.22%	0.24%
9	9.71%	6.09%	6.69%	0.06%
Średnia	7.01%	3.97%	3.82%	0.20%

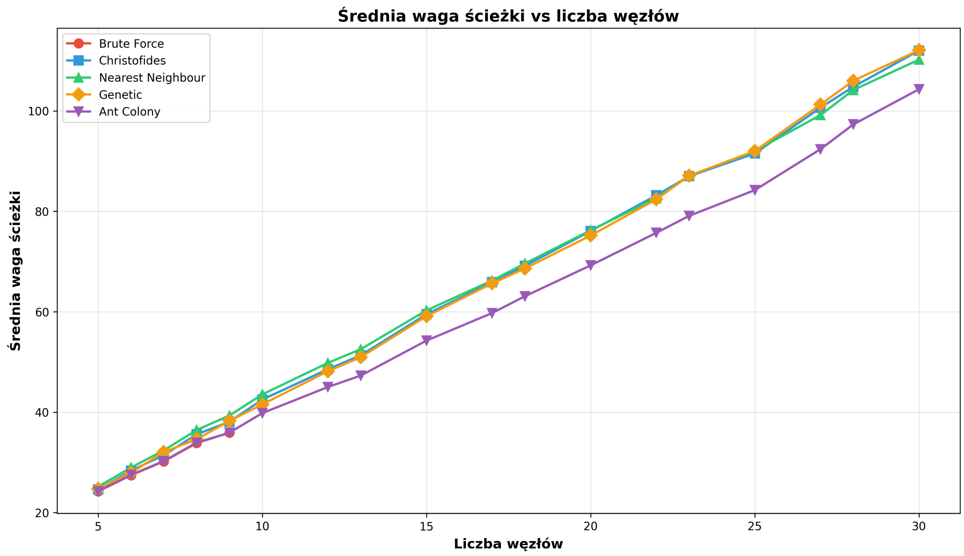
Tabela 3: Odchylenie od rozwiązania optymalnego

Ant Colony osiąga średnie odchylenie 0.20%. Christofides spełnia gwarancję teoretyczną z zapasem. Genetic wykazuje zmienność wyników. Nearest Neighbour — najgorsze odchylenie rosnące z n.

5.4 Wykresy



Rysunek 1: Zależność czasu wykonania od liczby węzłów (skala logarytmiczna)



Rysunek 2: Porównanie jakości rozwiązań dla różnych algorytmów

6 Analiza wyników

6.1 Wydajność czasowa

Wyniki potwierdziły teoretyczne przewidywania:

- **Brute Force:** Wzrost faktorialny czasu wykonania. Dla $n = 9$ czas wynosił 809 ms wobec 0.174 ms dla $n = 5$ (wzrost $\approx 4650\times$). Praktyczny jedynie dla $n < 10$.
- **Nearest Neighbour:** Najszybszy algorytm z czasem < 1 ms dla wszystkich testowanych rozmiarów.

- **Christofides:** Czas wykonania od 0.36 ms ($n = 5$) do 3.33 ms ($n = 30$).
- **Genetic:** Najdłuższy czas wykonania: 4.6–13.1 s. Czas względnie niezależny od n przy ustalonych parametrach.
- **Ant Colony:** Czas od 10.6 ms ($n = 5$) do 348 ms ($n = 30$).

6.2 Jakość rozwiązań

- **Ant Colony:** Najwyższa dokładność. Odchylenie od optimum 0.06–0.39% dla $n \leq 9$. Konsekwentnie najlepsze wyniki dla wszystkich rozmiarów grafów.
- **Christofides:** Spełnia gwarancję 1.5-aproksymacji (faktyczne odchylenie 1.51–6.09% dla $n \leq 9$).
- **Genetic:** Zmienna skuteczność. Odchylenie od optimum 1.74–6.69%. Wyniki zależne od losowości procesu ewolucyjnego.
- **Nearest Neighbour:** Najgorsze wyniki. Odchylenie 4.12–9.71% dla małych grafów, rosnące wraz z n.

6.3 Kompromis czas-jakość

Algorytm	**Czas dla $n = 30$ [ms]**	**Długość ścieżki**
Nearest Neighbour	0.17	110.21
Christofides	3.33	112.01
Ant Colony	347.80	**104.32**
Genetic	13133.85	112.09

Tabela 4: Porównanie czasu i jakości dla $n = 30$

Algorytm mrówkowy oferuje najkorzystniejszy stosunek jakości do czasu wykonania. Christofides zapewnia gwarancję teoretyczną przy niskim czasie. Nearest Neighbour jest najszybszy, ale uzyskuje najgorsze wyniki. Genetic nieefektywny przy obecnych parametrach.

6.4 Skalowalność

Algorytm	$t(n = 5)$ **[ms]**	$t(n = 30)$ **[ms]**	**Wzrost**
Brute Force	0.174	—	faktorialny
Nearest Neighbour	0.016	0.170	10.6×
Christofides	0.359	3.331	9.3×
Ant Colony	10.572	347.796	32.9×
Genetic	4581.518	13133.853	2.9×

Tabela 5: Skalowalność czasowa algorytmów

Nearest Neighbour i Christofides wykazują najlepszą skalowalność (wzrost $\approx 10\times$). Ant Colony skaluje się kwadratowo ($\approx 33\times$). Genetic względnie stały czas przy ustalonych parametrach. Brute Force niepraktyczny dla $n \geq 10$.

7 Wnioski

7.1 Podsumowanie

Przeprowadzono kompleksowe porównanie pięciu algorytmów na zbiorze 1800 instancji problemu TSP. Główne wnioski:

1. **Ant Colony Optimization** wykazał najwyższą skuteczność praktyczną. Średnie odchylenie od optimum wyniosło 0.20% dla małych grafów. Dla $n = 30$ przewaga nad konkurencją: 6–8%. Czas wykonania: 10–350 ms.

2. **Christofides** potwierdził gwarancję teoretyczną 1.5-aproksymacji (faktyczne odchylenie 1.51–6.09%). Czas wykonania < 4 ms.

3. **Nearest Neighbour** — najszybszy algorytm (czas < 1 ms), ale najgorsze wyniki (odchylenie 4–10%).

4. **Genetic Algorithm** nieefektywny przy obecnych parametrach (5–13 s). Wymaga optymalizacji liczby pokoleń i rozmiaru populacji.

5. **Brute Force** praktyczny jedynie dla $n < 10$ ze względu na złożoność $O(n!)$. Wartość referencyjna do weryfikacji innych algorytmów.