

Zadanie 3 – Algorytm mrówkowy

Sprawozdanie z implementacji i analizy algorytmu mrówkowego (ACO)

1. Opis algorytmu

Algorytm Mrówkowy (ACO) to metaheurystyka inspirowana zachowaniem prawdziwych mrówek, które znajdują najkrótszą drogę między mrowiskiem a pożywieniem dzięki śladom feromonowym.

a. Inicjalizacja

Na początku tworzona jest macierz odległości między wszystkimi miastami na podstawie ich współrzędnych. Następnie inicjalizowana jest macierz feromonowa – na każdej krawędzi (ścieżce między miastami) ustawiana jest początkowa, jednakowa wartość feromonu, dzięki temu żadna ścieżka nie jest na starcie faworyzowana.

b. Konstrukcja tras

W każdej iteracji populacja mrówek buduje swoje trasy:

- Każda mrówka jest umieszczana w losowo wybranym mieście początkowym.
- Mrówka wybiera kolejne miasto do odwiedzenia ze zbioru nieodwiedzonych jeszcze miast. Decyzja podejmowana jest na dwa sposoby:
 - **Eksploracja (ruch losowy):** Z małym prawdopodobieństwem mrówka wybiera miasto całkowicie losowo, co pozwala na odkrywanie nowych ścieżek.
 - **Eksplotacja (reguła probabilistyczna):** Wybór następuje metodą selekcji ruletkowej. Prawdopodobieństwo wyboru miasta zależy od poziomu feromonu τ_{ij} oraz heurystyki η_{ij} (odwrotności odległości $1/d_{ij}$), ważonych parametrami α i β .

c. Aktualizacja feromonu

Po tym, jak wszystkie mrówki ukończą swoje trasy, następuje aktualizacja globalnej macierzy feromonów, składająca się z dwóch procesów:

- Wyparowywanie: Wszystkie ślady feromonowe są osłabiane o współczynnik wyparowywania ρ (rho).

Zapobiega to przedwczesnej zbieżności algorytmu do lokalnego minimum i symuluje naturalny proces zanikania zapachu.

- Wzmocnienie śladów: Każda mrówka dokłada feromony na ścieżkach, które przebyła. Ilość dodanego feromonu jest odwrotnie proporcjonalna do całkowitej długości jej trasy. Im krótsza trasa, tym silniejszy ślad feromonowy zostanie pozostawiony.

d. Warunek stopu

Powyższe kroki są powtarzane przez zadaną liczbę iteracji T. Po zakończeniu pętli algorytm zwraca najlepszą trasę (o najmniejszym koszcie) znalezioną w całej historii symulacji.

2. Opis implementacji

- Reprezentacja grafu i obliczanie odległości:** Problem reprezentowany jest za pomocą macierzy odległości (sąsiedztwa). Wierzchołki (atrakcje) są przechowywane jako współrzędne, a odległości euklidesowe między każdą parą punktów są obliczane raz na początku działania programu.

```
def _calculate_distance_matrix(self):
    matrix = np.zeros((self.num_nodes, self.num_nodes))
    for i in range(self.num_nodes):
        for j in range(self.num_nodes):
            if i != j:
                matrix[i][j] = np.linalg.norm(self.coords[i] - self.coords[j])
    return matrix
```

- Sposób reprezentacji trasy i kosztu:** Każda mrówka jest obiektem klasy Ant, który przechowuje listę odwiedzonych identyfikatorów miast (*visited*) oraz sumaryczną długość przebytej trasy (*total_distance*).

```
class Ant:
    def __init__(self, start_node, num_nodes):
        self.visited = [start_node]
        self.current_node = start_node
        self.total_distance = 0.0
```

- Sposób inicjalizacji feromonów** Macierz feromonów jest inicjalizowana jako tablica o wymiarach NxN, wypełniona wartością 1.0. Dzięki temu na starcie każda ścieżka ma równe szanse wyboru.

```
self.pheromone_matrix = np.ones((self.num_nodes, self.num_nodes))
```

- Mechanizm konstrukcji trasy i reguła wyboru:** Mrówka wybiera następny wierzchołek spośród nieodwiedzonych.

```

if random.random() < p_random:
    next_node = random.choice(unvisited) # ruch losowy
else:
    # Wyliczanie prawdopodobieństw dla każdej trasy
    # obliczanie licznika dla każdego prawdopodobieństwa
    numerator = (tau ** alpha) * (heuristic ** beta)
    # selekcja ruletkowa
    next_node = np.random.choice(unvisited, p=probabilities)

```

e. Aktualizacja feromonów i wyparowywanie: Aktualizacja odbywa się globalnie po każdej iteracji w dwóch etapach:

Wyparowywanie: Mnożenie całej macierzy przez współczynnik ρ (ρ).

Wzmocnienie: Dodanie feromonu na ścieżkach przebytych przez mrówki. Ilość dodanego feromonu zależy od jakości trasy.

```

def _update_pheromones(self, ants):
    # f_ij = (1 - rho) * f_ij
    self.pheromone_matrix *= (1 - self.rho)

    for ant in ants:
        contribution = self.Q / ant.total_distance
        path = ant.visited
        for i in range(len(path) - 1):
            u, v = path[i], path[i+1]
            self.pheromone_matrix[u][v] += contribution
            self.pheromone_matrix[v][u] += contribution

```

3. Opis uruchomienia programu – instrukcja użytkownika

A. Uruchomienie i wymagania

Program napisany w języku *Python*. Wymaga bibliotek *numpy* oraz *matplotlib*.

Uruchomienie następuje z wiersza poleceń komendą:

python main.py

B. Format wejścia (Dane)

Program wczytuje pliki tekstowe (.txt) z katalogu *data/*.

Format pliku (3 kolumny): ID_punktu Współrzędna_X Współrzędna_Y.

C. Ustawienie parametrów

Konfiguracja odbywa się poprzez interaktywne menu w konsoli (brak konieczności edycji kodu). Użytkownik może definiować:

- Plik wejściowy.

- Parametry algorytmu: liczbę mrówek m , liczbę iteracji T , wagę (α, β), parowanie p oraz prawdopodobieństwo losowego ruchu (p_random).

D. Generowane wyniki

- Konsola:** Wyświetla czas obliczeń, długość najlepszej trasy oraz (w trybie eksperymentu) statystyki: średnią, medianę, min/max i odchylenie standardowe.
- Wykresy:** Po zakończeniu symulacji generowane jest okno z dwoma wykresami:
 - Postęp optymalizacji (spadek długości trasy w kolejnych iteracjach).
 - Wizualizacja mapy z narysowaną najlepszą trasą.

4. Eksperymenty

a. Opis eksperymentu

Celem przeprowadzonych eksperymentów było zbadanie wpływu kluczowych parametrów Algorytmu Mrówkowego (ACO) na jakość znajdowanych rozwiązań (minimalizację długości trasy) oraz czas obliczeń.

Badania przeprowadzono na zbiorze danych: *A-n32-k5.txt*.

Dla każdej badanej konfiguracji parametrów algorytm został uruchomiony 5 razy. Wyniki zostały uśrednione, a także wyznaczono odchylenie standardowe, aby ocenić stabilność rozwiązania.

Jako **konfigurację bazową (referencyjną)** przyjęto następujące parametry, które były stałe, o ile nie badano wpływu konkretnego z nich:

- Liczba mrówek (m): 50
- Liczba iteracji (T): 100
- Wpływ feromonów (α): 1.0
- Wpływ heurystyki (β): 5.0
- Współczynnik parowania (p): 0.5
- Prawdopodobieństwo losowe (p_random): 0.01
- Ilość feromonu (Q): 100.0

b. Wyniki

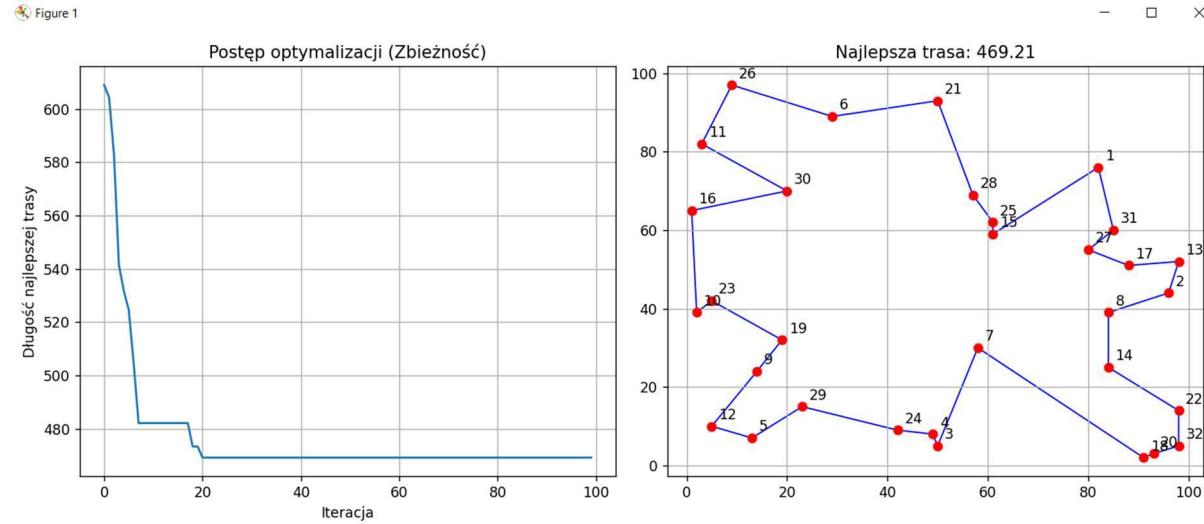
Eksperiment 1: Wpływ liczby mrówek (m) Zbadano wpływ liczby mrówek w populacji na zbieżność algorytmu. Przetestowano wartości $m = \{10, 50, 100\}$.

Tabela 1:

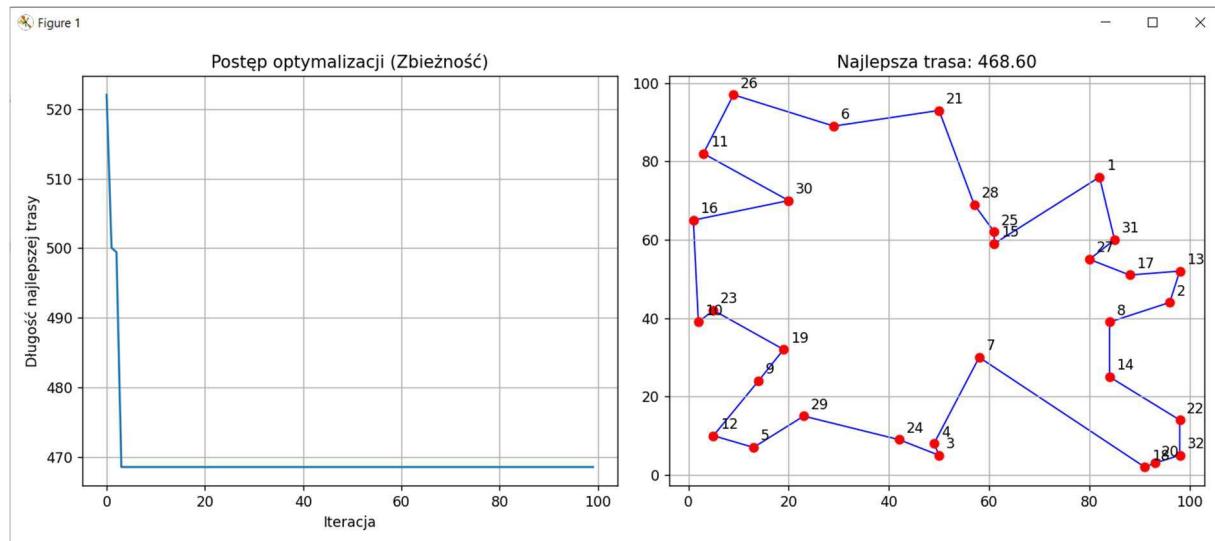
M	Średnia długość trasy	Najlepszy wynik	Najgorszy wynik	Odchylenie std.	Średni czas wykonania programu
10	474.13	469.21	481.82	4,31	1.2409 s
50	470.26	468.60	472.44	1,56	6.2117 s

100 468.31	467.71	469.21	0,73 12.2404 s
--------------	--------	--------	------------------

Wykresy przedstawiające wynik najlepszej trasy dla m = 10

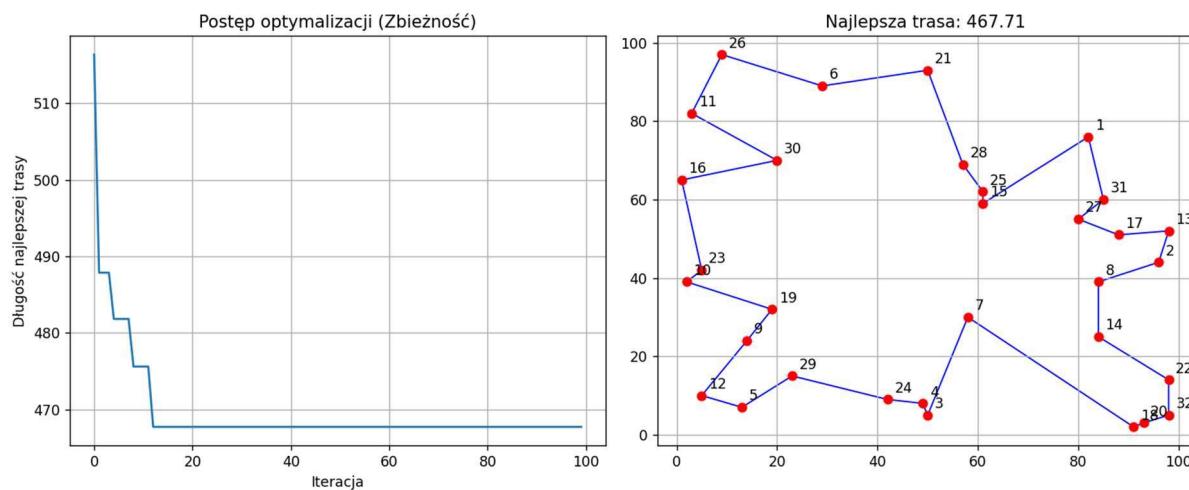


Wykresy przedstawiające wynik najlepszej trasy dla m = 50



Wykresy przedstawiające wynik najlepszej trasy dla m = 100

Figure 1



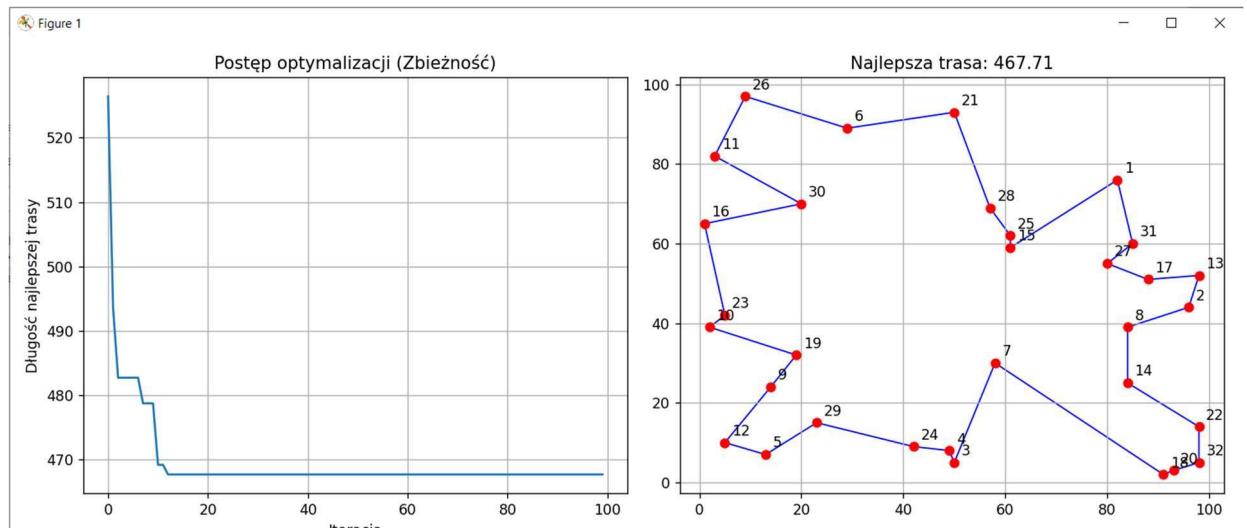
Eksperyment 2: Wpływ prawdopodobieństwa wyboru losowego (p_random)

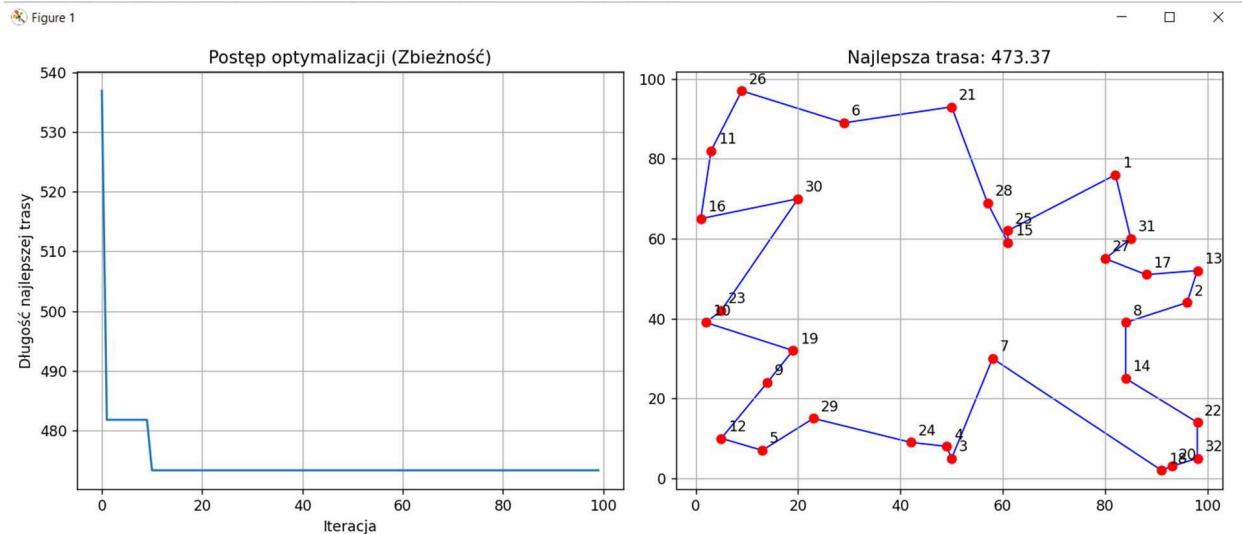
Zbadano, jak wprowadzenie losowości (ignorowanie feromonów) wpływa na unikanie maksimów lokalnych. Testowano $p_{\text{random}} = \{0.0, 0.05, 0.2\}$.

Tabela 2:

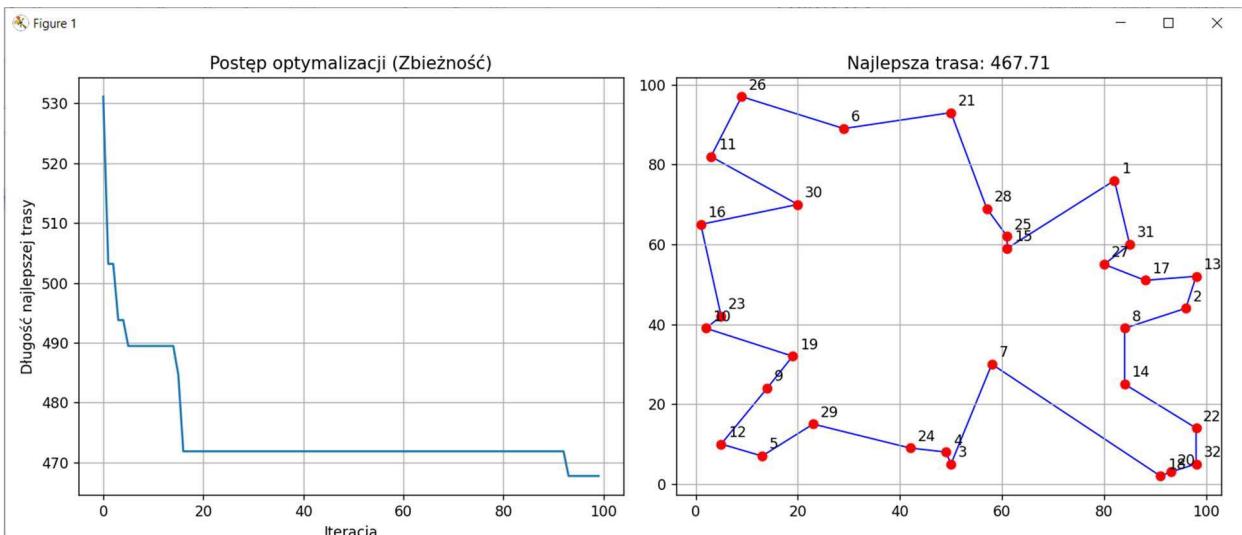
p_{random}	Średnia długość trasy	Najlepszy wynik	Najgorszy wynik	Odchylenie std.	Średni czas wykonania programu
0.0	469.21	467.71	473.37	2,20	6.2345 s
0.05	472.25	467.71	478.18	3,51	5.9410 s
0.20	499.40	486.54	523.21	12,76	5.2698 s

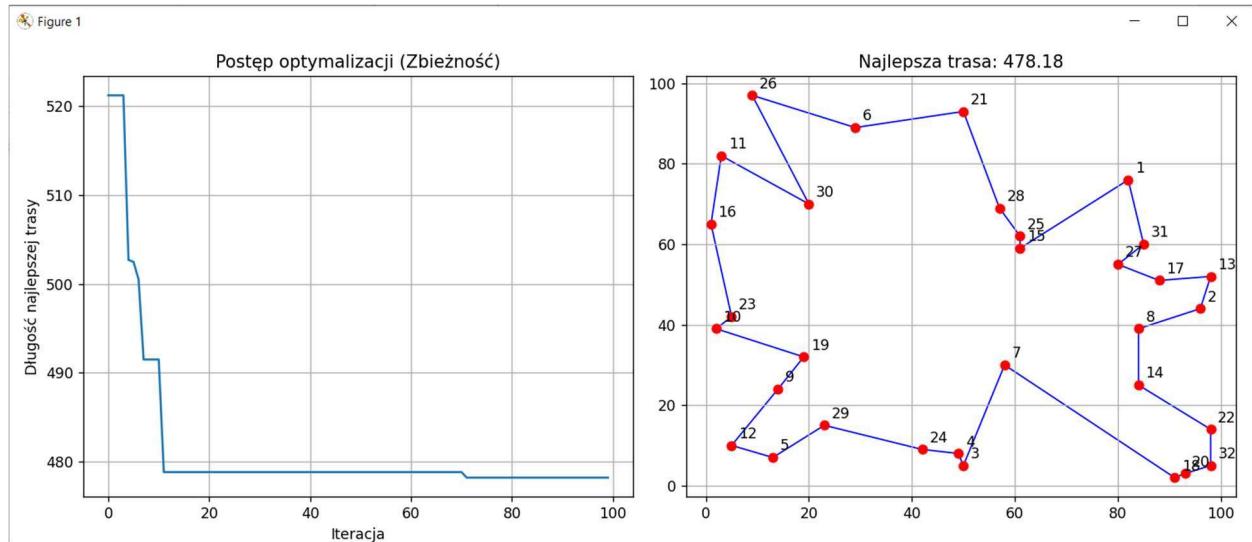
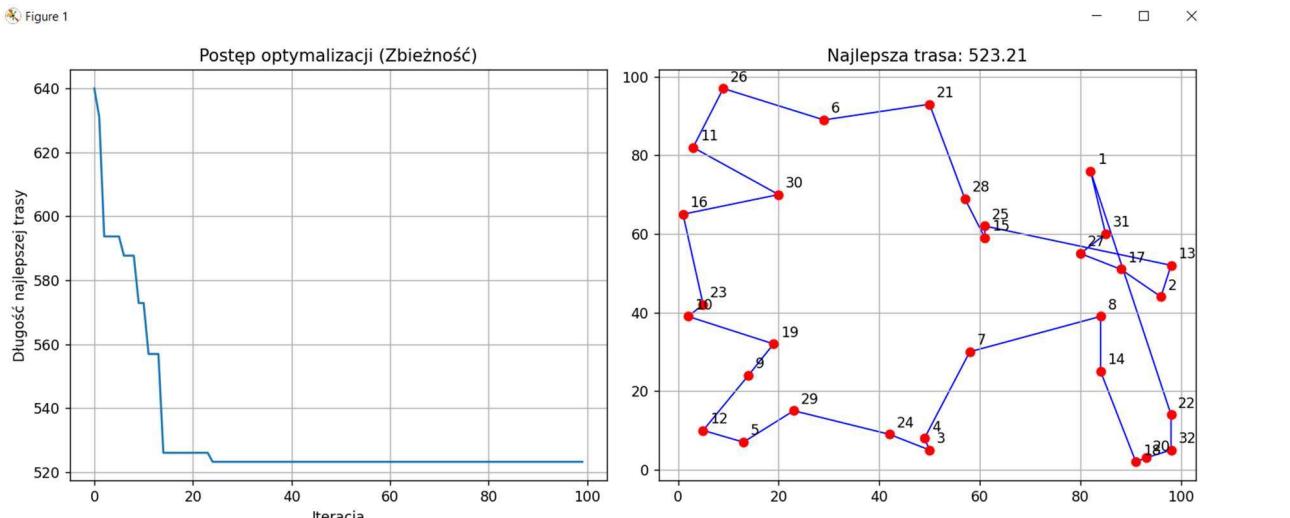
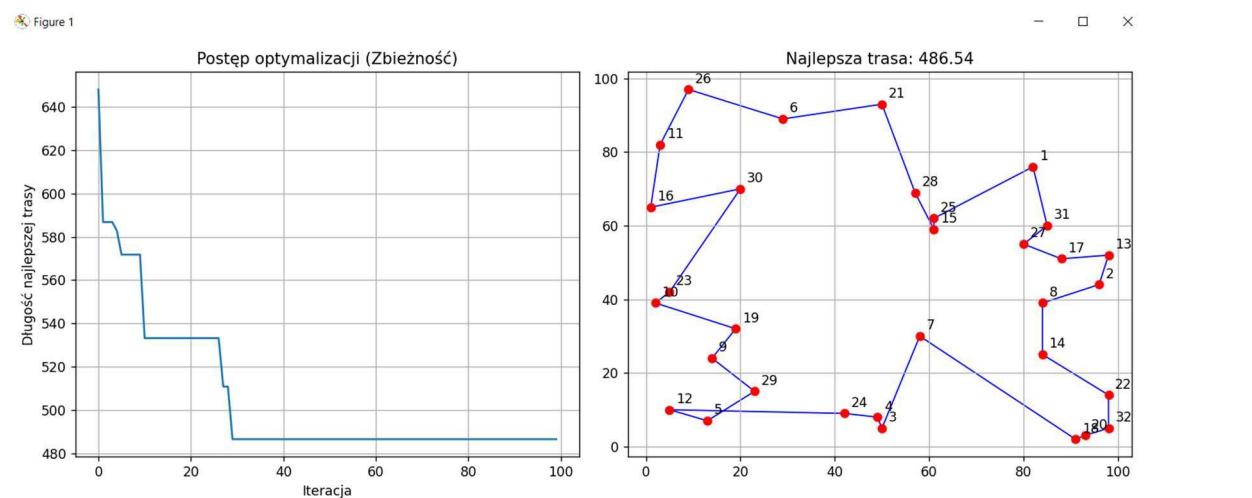
Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $p_{\text{random}} = 0.0$





Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla p_random =0.05



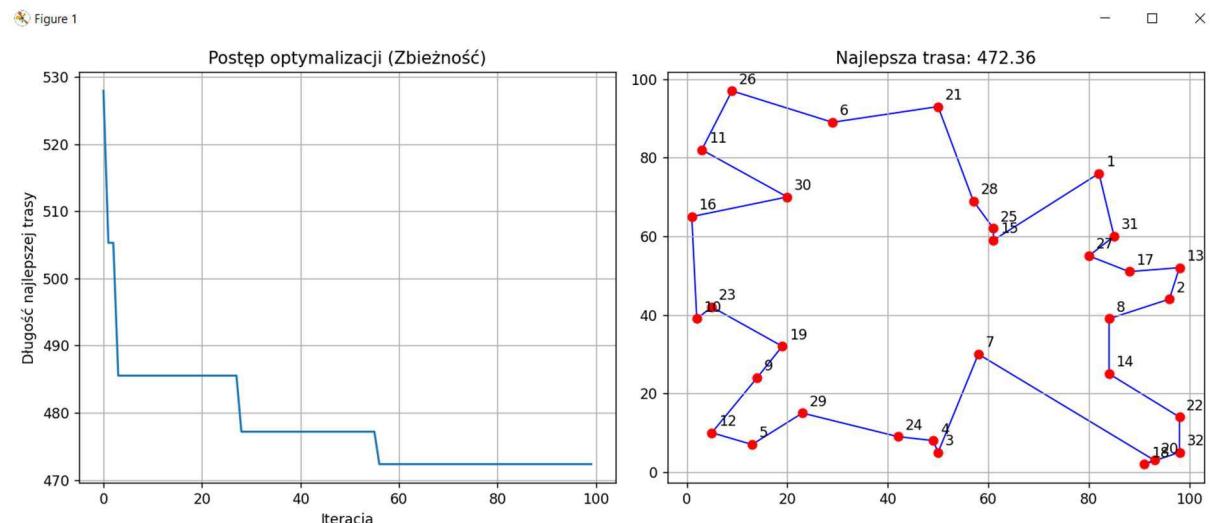
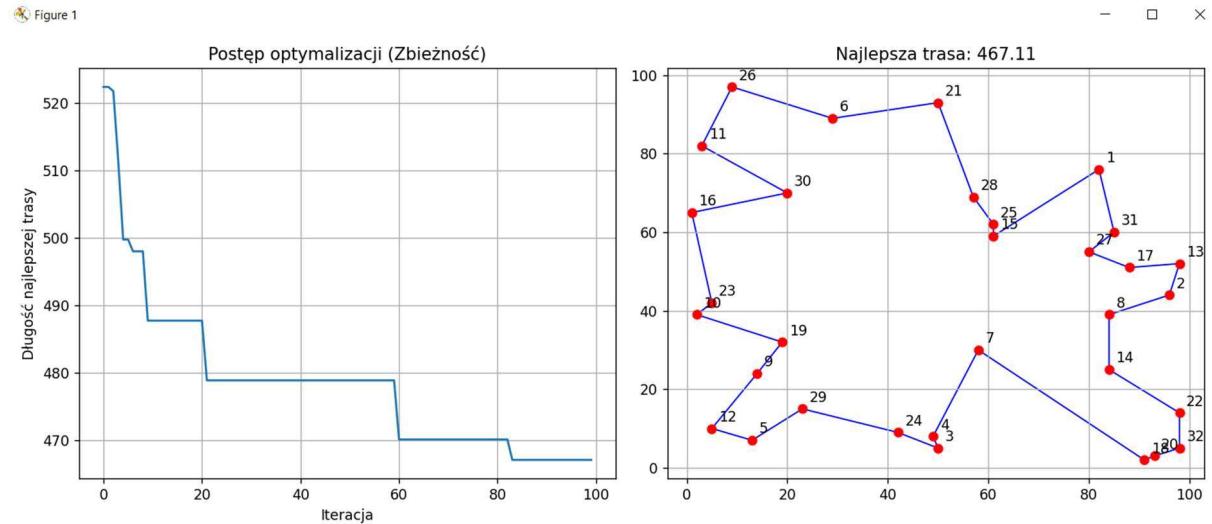
Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $p_{random} = 0.20$ 

Eksperyment 3: Wpływ wagi feromonów (a) Parametr ten określa, jak bardzo mrówki sugerują się pozostawionym śladem zapachowym. Testowano $\alpha = \{0.5, 1.0, 5.0\}$.

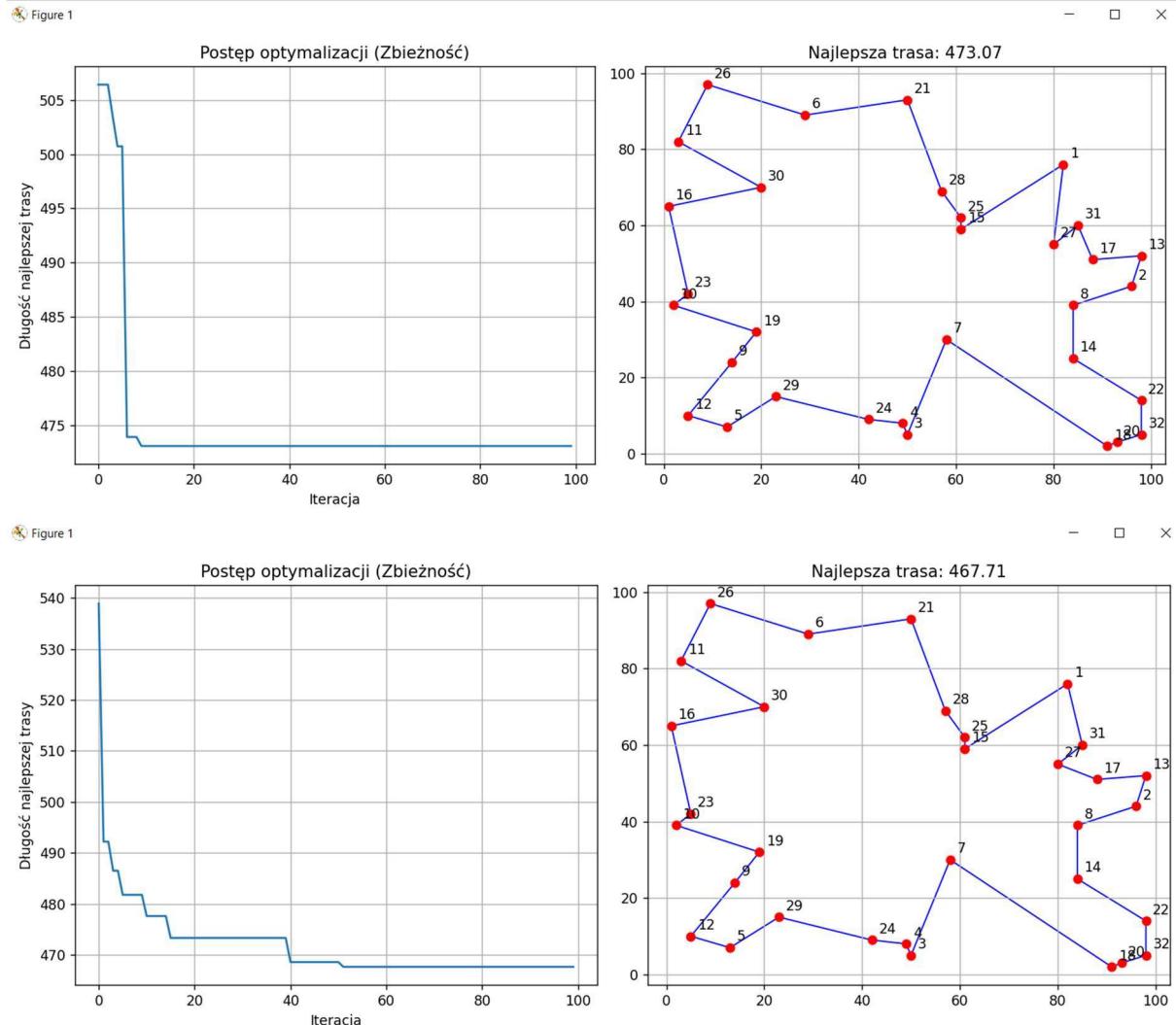
Tabela 3:

α	Średnia długość trasy	Najlepszy wynik	Najgorszy wynik	Odchylenie std.	Średni czas wykonania programu
0.5	468.46	467.11	472.36	1,98	6.3014 s
1.0	469.26	467.71	473.07	1,99	6.1248 s
5.0	490.47	484.79	494.19	3,56	6.3040 s

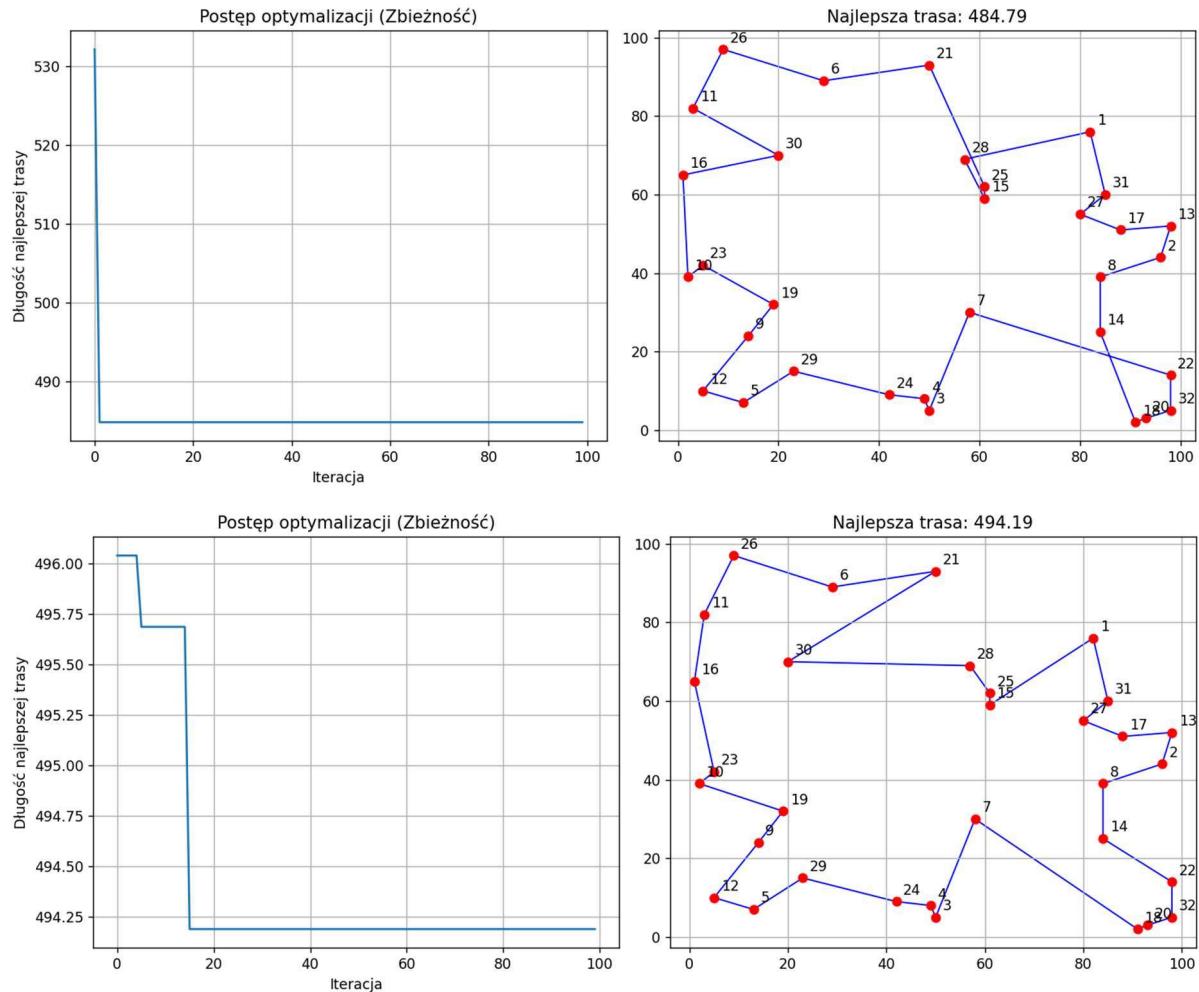
Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\alpha = 0.5$



Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\alpha = 1.0$



Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\alpha = 5.0$

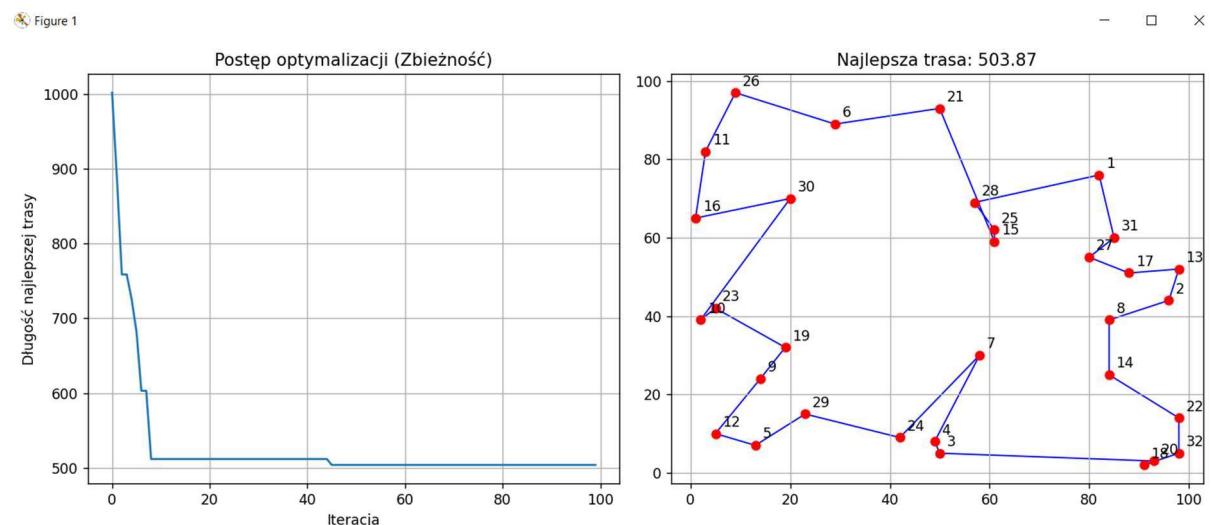
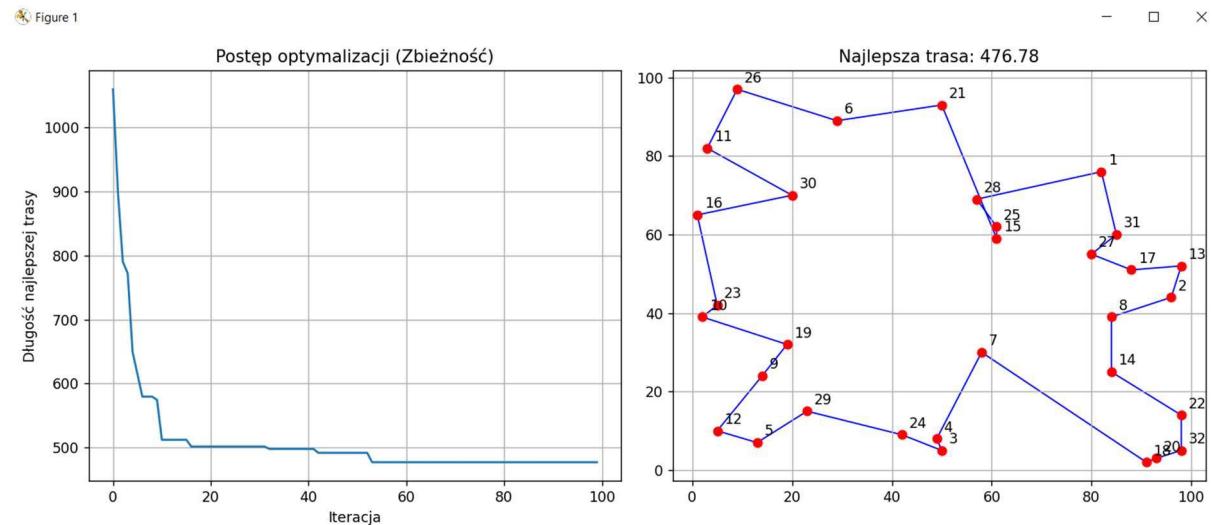


Eksperyment 4: Wpływ wagi heurystyki (β) Parametr ten decyduje, jak ważna jest odległość do następnego miasta (wybór zachłanny). Testowano $\beta = \{1.0, 5.0, 10.0\}$.

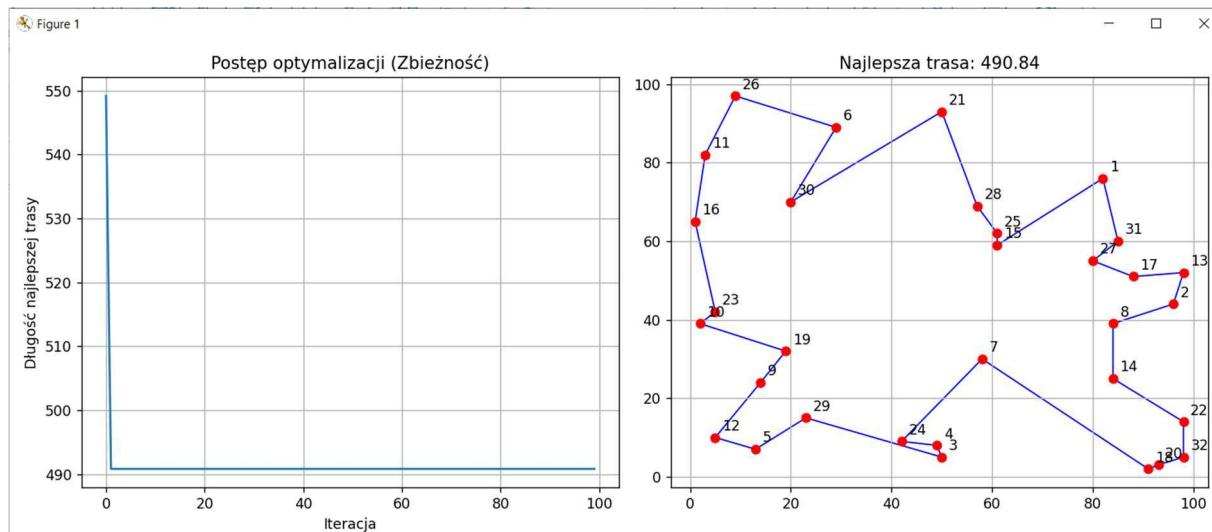
Tabela 4:

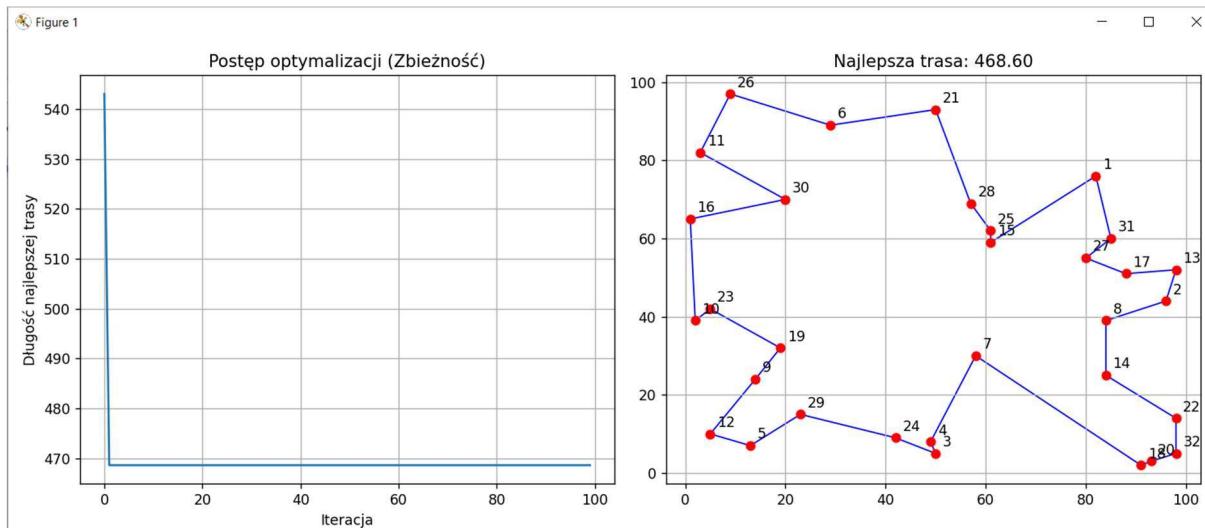
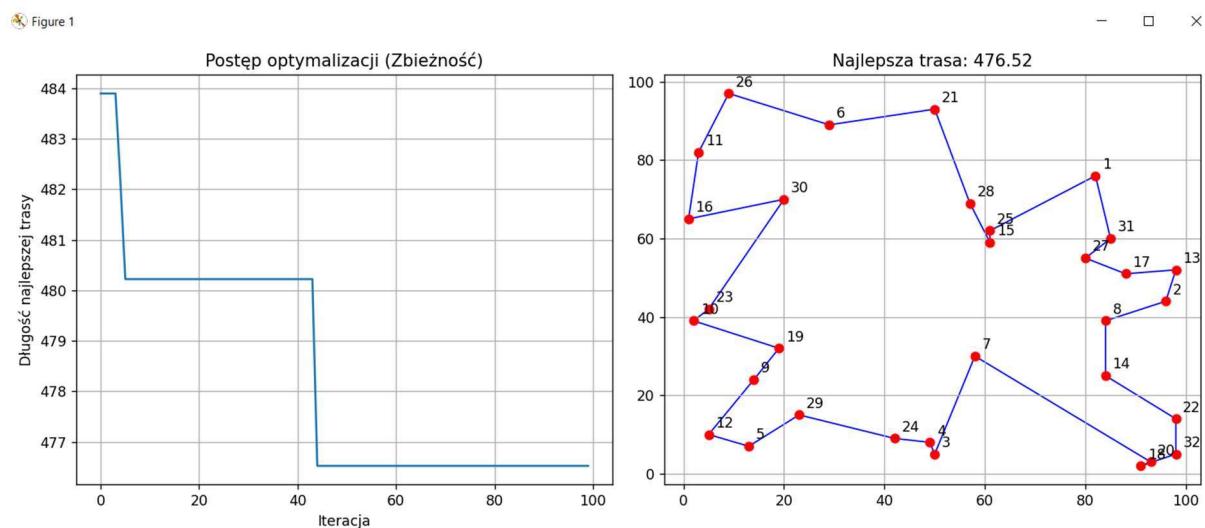
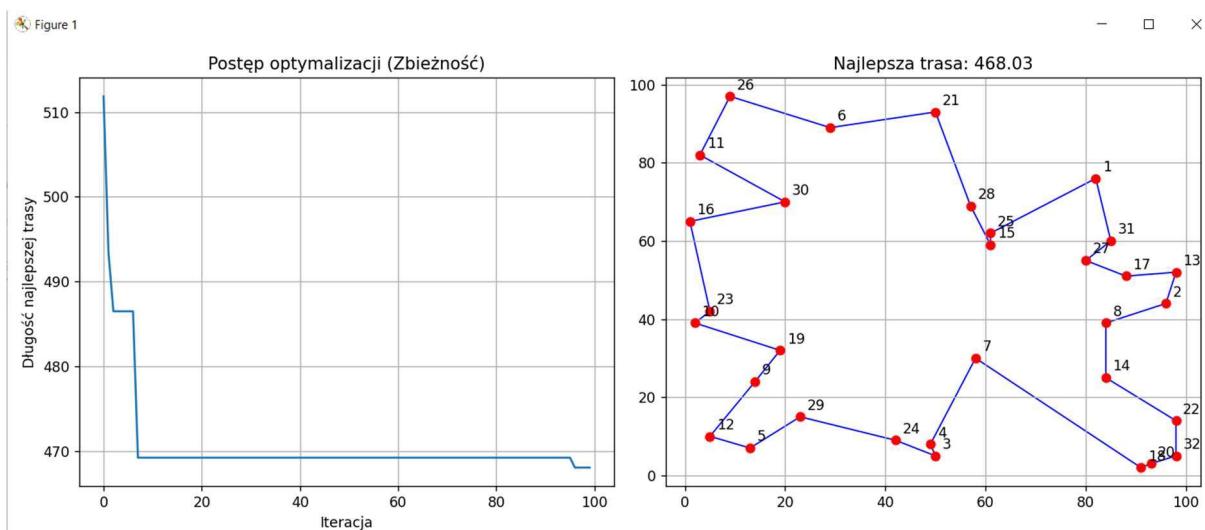
β	Średnia długość trasy	Najlepszy wynik	Najgorszy wynik	Odchylenie std.	Średni czas wykonania programu
1.0	490.10	476.78	503.87	11,24	6.0519 s
5.0	476.99	468.60	490.84	8,84	6.8157 s
10.0	472.75	468.03	476.52	2,73	6.0719 s

Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\beta = 1.0$



Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\beta = 5.0$



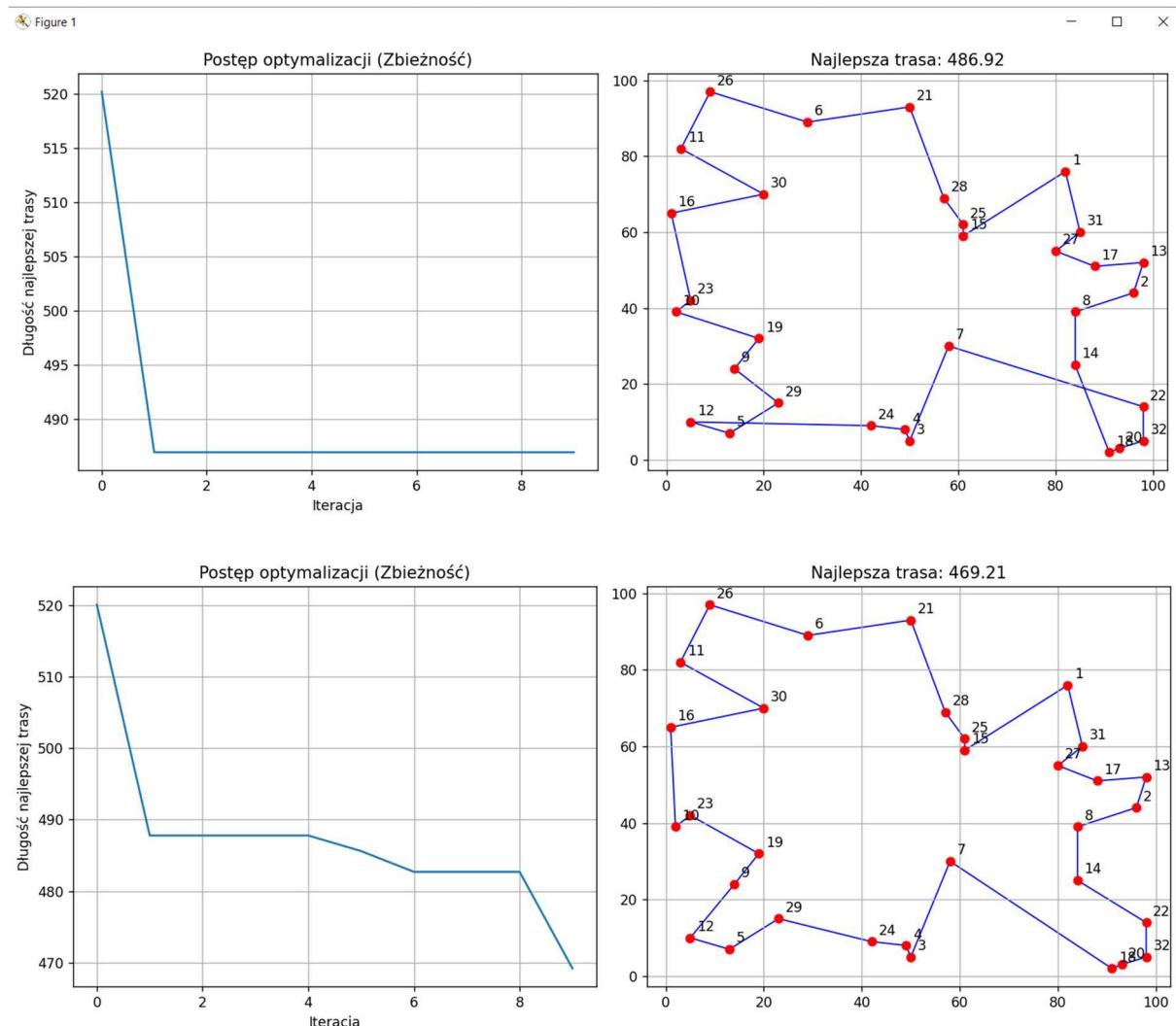
Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\beta = 10.0$ 

Eksperyment 5: Wpływ liczby iteracji (T) Sprawdzono, jak długo algorytm jest w stanie poprawiać rozwiązanie. Testowano T= {10, 100, 300}.

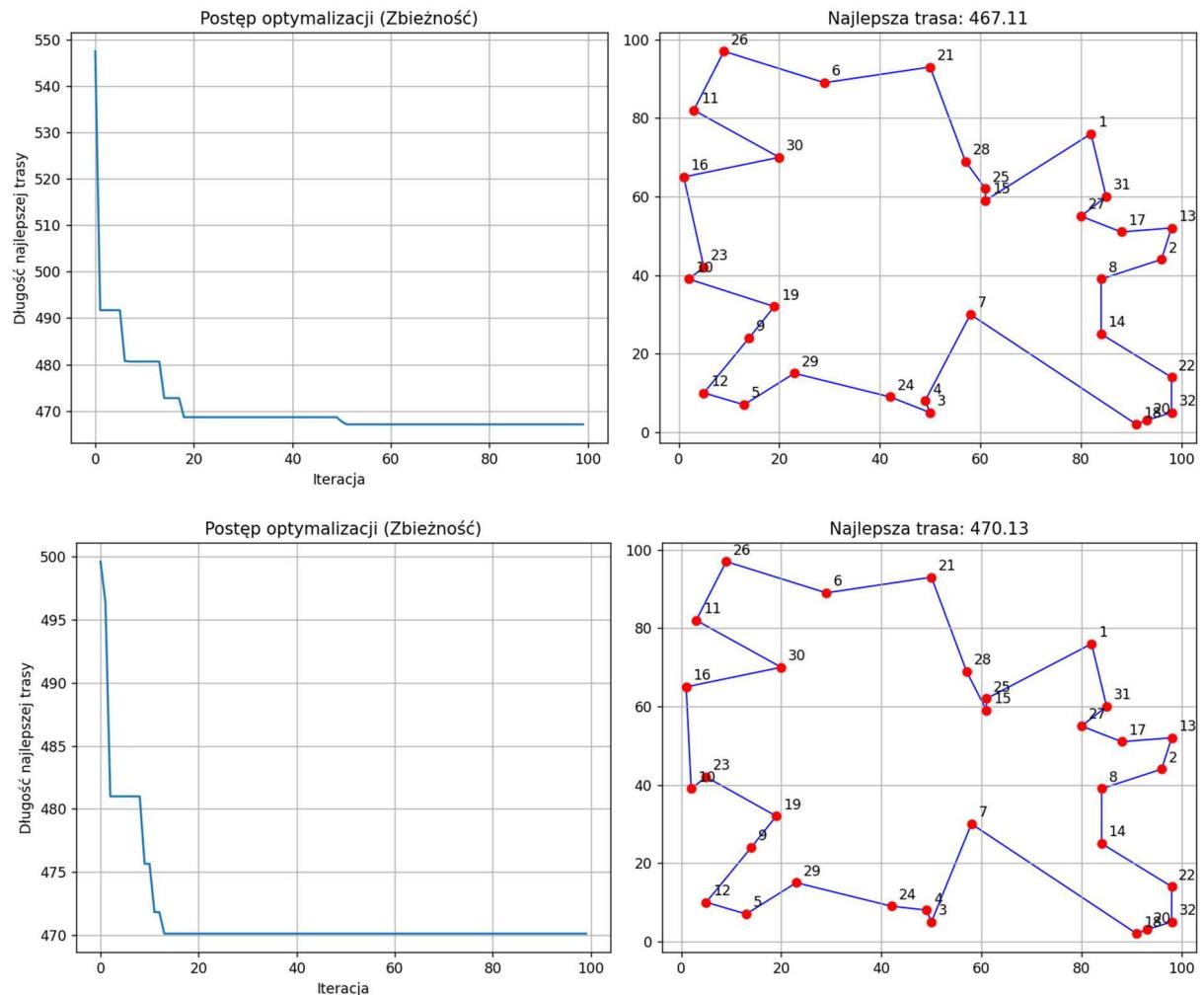
Tabela 5:

T	Średnia długość trasy	Najlepszy wynik	Najgorszy wynik	Odchylenie std.	Średni czas wykonania programu
10	477.16	469.21	486.92	6,41	0.6346 s
100	468.37	467.11	470.13	1,12	5.9263 s
300	467.65	467.11	468.03	0,30	18.6981 s

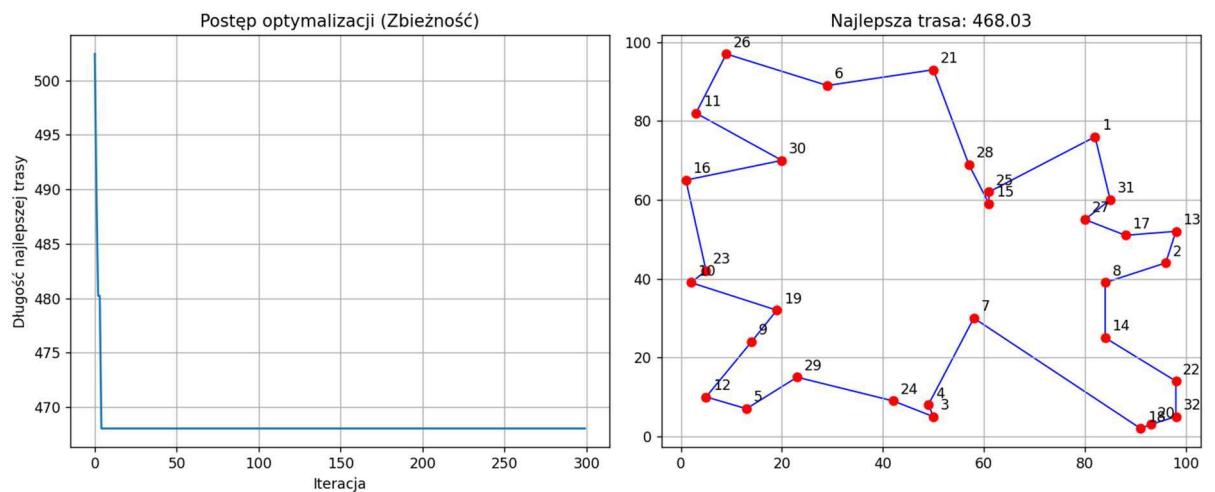
Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla T = 10

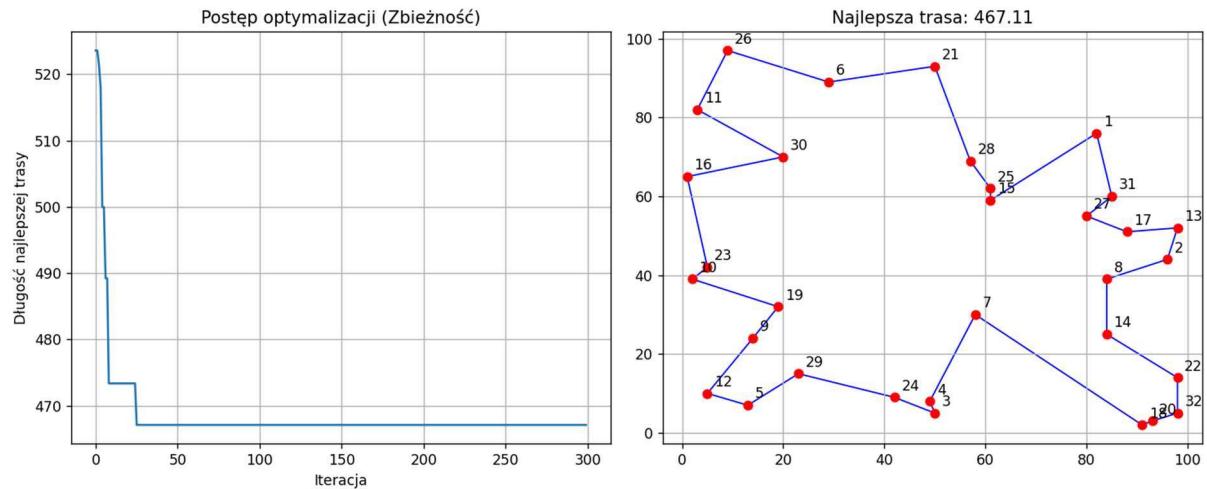


Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla T = 100



Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla T = 300



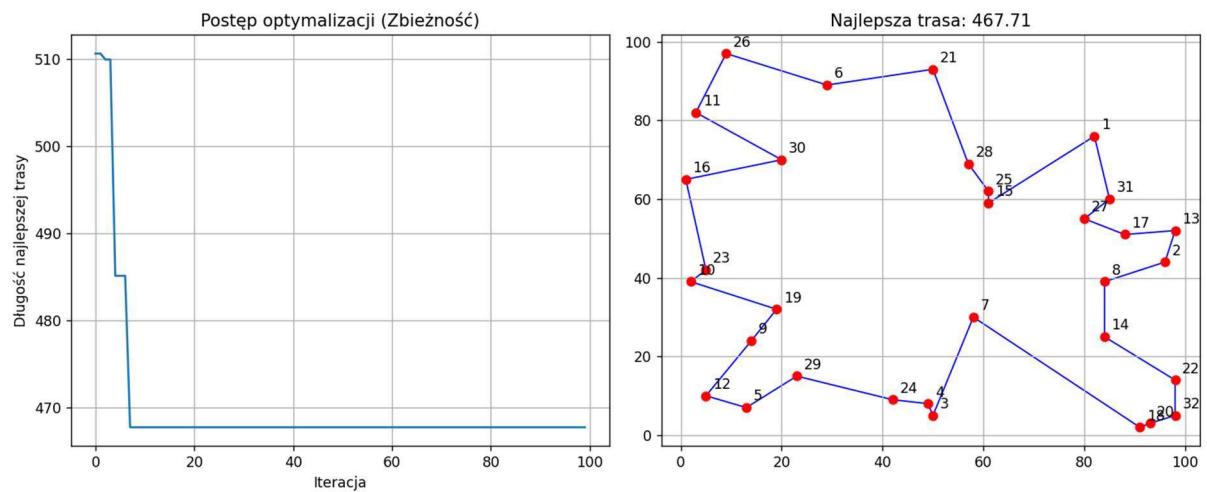


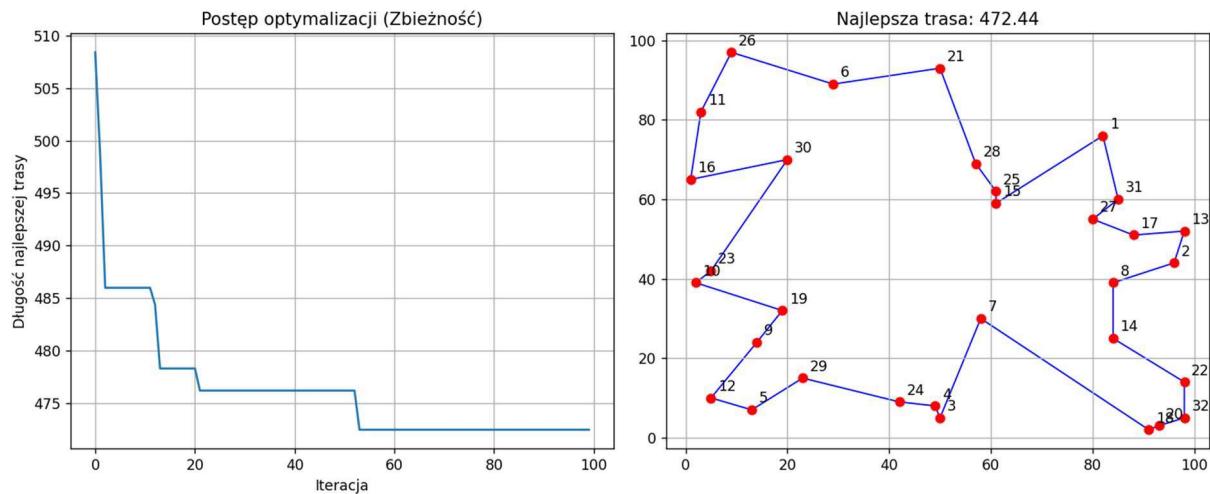
Eksperyment 6: Wpływ parowania feromonów (ρ) Parowanie odpowiada za "zapominanie" starych tras. Testowano $\rho = \{0.1, 0.5, 0.9\}$.

Tabela 6:

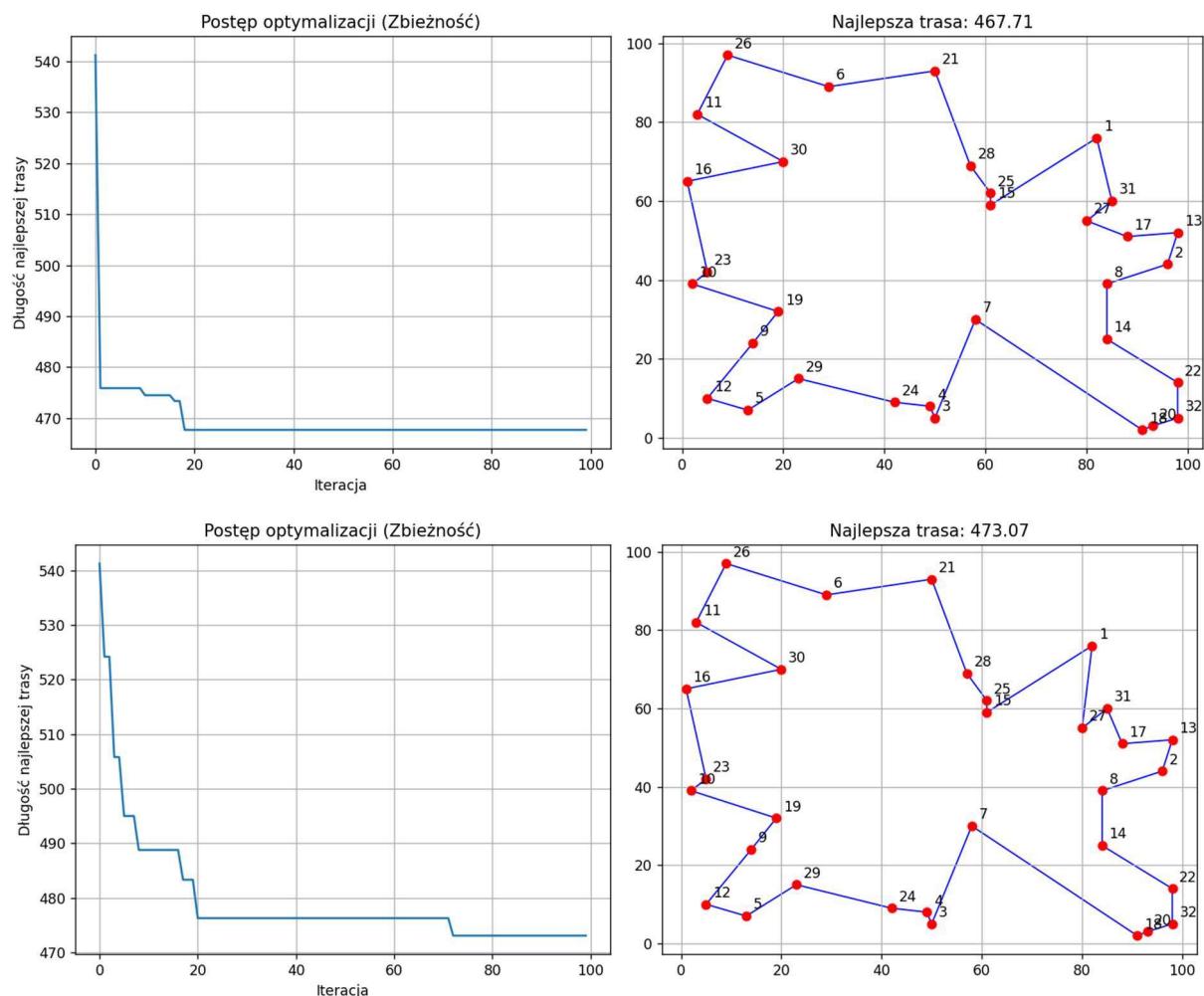
ρ	Średnia długość trasy	Najlepszy wynik	Najgorszy wynik	Odchylenie std.	Średni czas wykonania programu
0.1	469.43	467.71	472.44	1,60	6.2492 s
0.5	470.97	467.71	473.07	2,12	6.0399 s
0.9	470.45	467.71	473.37	2,12	6.2412 s

Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\rho = 0.1$.

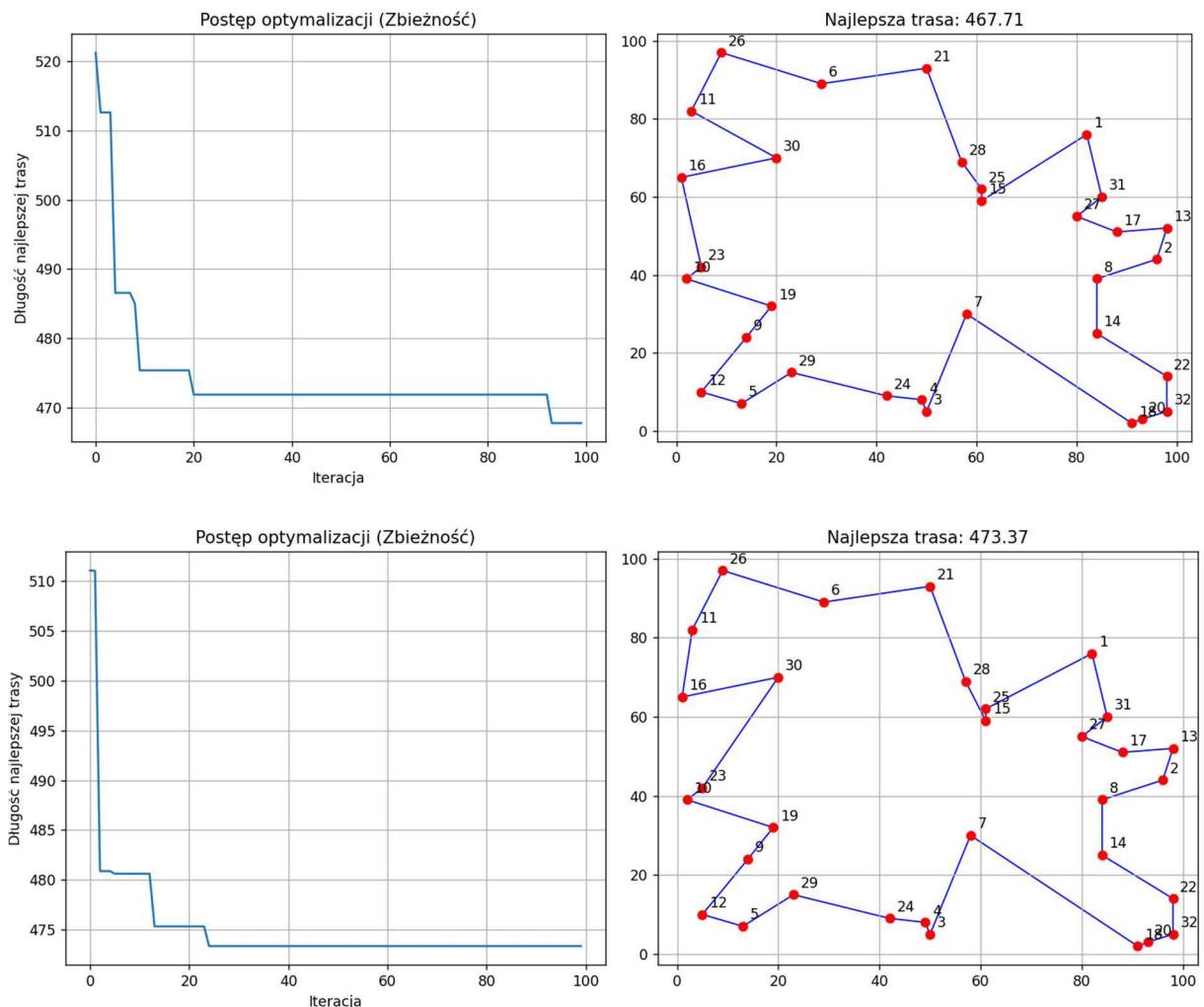




Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\rho = 0.5$.



Wykresy przedstawiające wynik najlepszej i najgorszej trasy dla $\rho = 0.9$.



5. Pytania

Jaki wpływ na wyniki ma liczebność mrówek?

Zwiększenie liczebności mrówek (m) prowadzi do poprawy jakości rozwiązania oraz znaczącego wzrostu stabilności algorytmu. Jak pokazują wyniki, przy wzroście populacji z 10 do 100 mrówek średnia długość trasy spadła z 474,13 do 468,31, co świadczy o dokładniejszej eksploracji przestrzeni rozwiązań. Kluczowa zmiana nastąpiła w odchyleniu standardowym, które zmalało z 4,31 do 0,73, co oznacza, że przy większej populacji algorytm generuje bardzo powtarzalne wyniki. Poprawa ta odbywa się jednak kosztem czasu obliczeń, który wzrósł niemal liniowo z 1,24 s (dla $m=10$) do 12,24 s (dla $m=100$). Można zauważyć, że przejście z 50 na 100 mrówek daje już tylko niewielką poprawę wyniku przy dwukrotnym wydłużeniu czasu działania.

Jaki wpływ na działanie algorytmu na wprowadzenie prawdopodobieństwa wyboru przez mrówkę losowej atrakcji?

W badanym przypadku najlepsze rezultaty osiągnięto przy braku losowości ($p_random = 0.0$), uzyskując najniższą średnią długość trasy (469,21) oraz najmniejsze odchylenie standardowe (2,20), co wskazuje na stabilną zbieżność algorytmu. Wprowadzenie niewielkiej losowości (0.05) pozwoliło znaleźć to samo najlepsze rozwiązanie (467,71), jednak średnia jakość wyników nieznacznie spadła do 472,25. Zwiększenie parametru do 0.20 drastycznie pogorszyło wyniki (średnia 499,40) i zwiększyło niestabilność (odchylenie 12,76), co oznacza, że zbyt duża losowość zakłóca wykorzystanie śladów feromonowych. Można zauważyc, że większa losowość nieznacznie skraca czas obliczeń (z 6,23 s do 5,27 s), ponieważ wybór losowy jest obliczeniowo prostszy niż selekcja ruletkowa.

Jaki wpływ na wyniki ma waga feromonów na potrzeby wyboru ścieżki przez mrówki - współczynnik α ?

Z eksperymentów wynika, że niższe wartości współczynnika α sprzyjają znajdowaniu lepszych rozwiązań. Najkorzystniejszy wynik średni (468,46) oraz najlepsze znalezione rozwiązanie (467,11) uzyskano dla $\alpha=0.5$, co sugeruje, że umiarkowany wpływ feromonów pozwala na efektywną eksplorację. Zwiększenie wartości do $\alpha=1.0$ dało bardzo zbliżone rezultaty (średnia 469,26), jednak dalsze podniesienie parametru do $\alpha=5.0$ spowodowało znaczące pogorszenie jakości wyniku, której średnia długość wzrosła do 490,47. Dla wysokiego α odnotowano również wzrost odchylenia standardowego do 3,56, co świadczy o mniejszej stabilności algorytmu. Wskazuje to, że zbyt silne podążanie za śladem feromonowym (wysokie α) prowadzi do przedwczesnej zbieżności do lokalnie optymalnych rozwiązań, ignorując potencjalnie lepsze ścieżki. Czas obliczeń pozostał na stałym poziomie ok. 6,1–6,3 s niezależnie od wartości parametru.

Jaki wpływ na wyniki ma waga heurystyki na potrzeby wyboru ścieżki przez mrówkę - współczynnik β ?

Zwiększenie wagi heurystyki (β) ma wyraźnie pozytywny wpływ na jakość i stabilność otrzymywanych rozwiązań. Dla $\beta = 1.0$ średnia długość trasy wynosiła 490,10, podczas gdy dla $\beta = 10.0$ spadła ona do 472,75, co oznacza, że silniejsze preferowanie krótkich odcinków (zachłanność) sprzyja optymalizacji w tym problemie. Wzrost wartości parametru do 10.0 pozwolił również na znalezienie najlepszego rozwiązania w tej serii (468,03). Najbardziej znacząca zmiana nastąpiła w stabilności algorytmu – odchylenie standardowe zmalało ponad czterokrotnie, z 11,24 (dla $\beta = 1.0$) do 2,73 (dla $\beta = 10.0$). Wyniki te potwierdzają, że w tym problemie lokalna informacja o odległości (heurystyka) jest kluczowa dla efektywności algorytmu mrówkowego.

Jaki wpływ na wyniki ma liczba iteracji?

Liczba iteracji (T) ma bezpośrednie przełożenie na zdolność algorytmu do znalezienia i utrwalenia optymalnej trasy. Przy małej liczbie iteracji ($T=10$) średni wynik wynosił

477,16, a odchylenie standardowe było wysokie (6,41), co oznacza, że algorytm często przerwał pracę przed osiągnięciem zbieżności. Zwiększenie liczby iteracji do 100 przyniosło znaczną poprawę średniego wyniku do 468,37 oraz spadek odchylenia do 1,12, co wskazuje na stabilizację rozwiązania. Dalsze zwiększanie T do 300 dało już tylko minimalną poprawę średniej (467,65) przy trzykrotnym wzroście czasu obliczeń (z 5,93 s do 18,70 s), co sugeruje, że dla tego problemu 100 iteracji jest wartością optymalną.

Jaki wpływ na wyniki współczynnik wyparowywania feromonów?

W badanym zakresie współczynnik wyparowywania feromonów (ρ) wykazał stosunkowo niewielki wpływ na efektywność algorytmu. Niezależnie od przyjętej wartości parametru (0,1, 0,5, 0,9), w każdej próbie udało się znaleźć to samo najlepsze rozwiązanie o koszcie 467,71. Średnia długość trasy była najbardziej korzystna dla niskiego parowania ($\rho = 0,1$) i wynosiła 469,43, podczas gdy dla wyższych wartości nieznacznie wzrosła do poziomu ok. 470,5–471,0. Zaobserwowano, że wolniejsze wyparowywanie $\rho = 0,1$ sprzyja większej powtarzalności wyników, co potwierdza najwyższe odchylenie standardowe równe 1,60. Dla wartości $\rho = 0,5$ oraz $\rho = 0,9$ odchylenie to wzrosło do 2,12, co sugeruje, że zbyt szybkie usuwanie śladów może wprowadzać niestabilność. Czas wykonywania obliczeń był zbliżony we wszystkich przypadkach i wynosił średnio około 6,0–6,2 sekundy.

6. Wnioski

Przeprowadzone eksperymenty wykazały, że Algorytm Mrówkowy (ACO) jest skuteczną metodą rozwiązywania problemu komiwojażera, jednak jego efektywność ściśle zależy od doboru parametrów sterujących.

- Kluczowa rola heurystyki:** Największy wpływ na poprawę wyników miało zwiększenie wagi heurystyki β . Ustawienie wartości β w zakresie 5.0 – 10.0 pozwalało algorytmowi znacznie szybciej znajdować optymalne trasy, co sugeruje, że w badanym problemie informacja o lokalnej odległości jest ważniejsza niż ślad feromonowy.
- Liczebność populacji:** Zwiększenie liczby mrówek poprawia stabilność wyników (mniejsze odchylenie standardowe), ale odbywa się to kosztem wzrostu czasu obliczeń. Optymalnym kompromisem wydaje się populacja $m=100$, która zapewnia wysoką powtarzalność wyników przy akceptowalnym czasie działania.
- Dobór parametrów feromonowych:** Najlepsze rezultaty uzyskano dla umiarkowanego wpływu feromonów ($\alpha=0,5 - 1,0$) oraz niskiego współczynnika wyparowywania ($\rho=0,1$). Zbyt silne przywiązanie do feromonów ($\alpha=5,0$) lub zbyt szybkie ich usuwanie ($\rho=0,9$) prowadziło do pogorszenia jakości rozwiązań.
- Iteracje:** Algorytm wykazuje cechy szybkiej zbieżności – większość optymalizacji następuje w pierwszych 100 iteracjach. Dalsze wydłużanie symulacji (do 300) przynosi nieproporcjonalnie małe korzyści w stosunku do zużytego czasu obliczeniowego.