IMO – LAB 1 Heurystyki zachłanne

Jan Bróździak 141142

Opis zadania:

Celem zadania było rozwiązanie zmodyfikowanego problemu komiwojażera, w którym należy ułożyć dwie rozłączne zamknięte ścieżki (cykle), każda zawierająca 50% wierzchołków (w przypadku nieparzystej liczby wierzchołków jedna ścieżka zawiera jeden wierzchołek więcej), minimalizując łączną długość obu ścieżek. Problem został rozwiązany dla instancji kroa100 i krob100 z biblioteki TSPLib, będących dwuwymiarowymi instancjami euklidesowymi.

Opis zaimplementowanych algorytmów w pseudokodzie:

Nearest Neigbour:

```
function nearest_neighbor(coords, distance_matrix):
    cycle1, cycle2, visited = initialize_two_cycles(coords)
    target_size = (total_vertices + 1) // 2

    while len(visited) < total_vertices:
        if len(cycle1) < target_size:
            cycle1 = insert_nearest_vertex(cycle1, visited, distance_matrix)

        if len(cycle2) < total_vertices // 2 and len(visited) < total_vertices:
            cycle2 = insert_nearest_vertex(cycle2, visited, distance_matrix)

    cycle1.append(cycle1[0])
    cycle2.append(cycle2[0])

    return cycle1, cycle2

function insert_nearest_vertex(cycle, visited, distance_matrix):
    last_vertex = cycle[-1]
    nearest_vertex = find_nearest_unvisited_vertex(last_vertex, visited,
distance_matrix)

    best_position = find_best_insertion_position(cycle, nearest_vertex,
distance_matrix)

    cycle.insert(best_position, nearest_vertex)
    visited.add(nearest_vertex)

    return cycle
```

Metoda rozbudowy cyklu (Greedy Cycle):

```
function greedy_cycle_method(coords, distance_matrix):

    cycle1, cycle2, visited = initialize_two_greedy_cycles(distance_matrix)

    while len(visited) < total_vertices:
        cycle1 = insert_best_vertex_greedy(cycle1, visited, distance_matrix)
        cycle2 = insert_best_vertex_greedy(cycle2, visited, distance_matrix)

    return cycle1, cycle2

function insert_best_vertex_greedy(cycle, visited, distance_matrix):
    best_increase = ∞
    best_vertex = null
    best_position = null

    for vertex in range(len(distance_matrix)):
        if vertex not in visited:
            for position in range(1, len(cycle) - 1):
                new_cycle = insert_vertex_at_position(cycle, vertex, position)
                increase = calculate_cycle_length(new_cycle, distance_matrix) -
calculate_cycle_length(cycle, distance_matrix)
                if increase < best_increase:
                    best_increase = increase
                    best_vertex = vertex
                    best_position = position + 1

    if best_vertex is not null:
        cycle.insert(best_position, best_vertex)
        visited.add(best_vertex)

    return cycle
```

Heurystyka z 2-żalem:

```
function expand_cycles_with_2_regret(coords, distance_matrix):

    cycle1, cycle2, visited = initialize_two_greedy_cycles(distance_matrix)

    while len(visited) < total_vertices:
        if len(visited) < total_vertices:
            cycle1 = insert_with_2_regret(cycle1, visited, distance_matrix)
        if len(visited) < total_vertices:
            cycle2 = insert_with_2_regret(cycle2, visited, distance_matrix)

    return cycle1, cycle2

function insert_with_2_regret(cycle, visited, distance_matrix):
    insertions = calculate_insertion_costs(cycle, visited, distance_matrix)

    insertions.sort_by_regret_and_cost()

    if insertions:
        highest_regret = insertions[0]
        best_vertex = highest_regret.vertex
        best_position = highest_regret.position
        cycle.insert(best_position, best_vertex)
        visited.add(best_vertex)

    return cycle
```

Wyniki eksperymentu obliczeniowego:

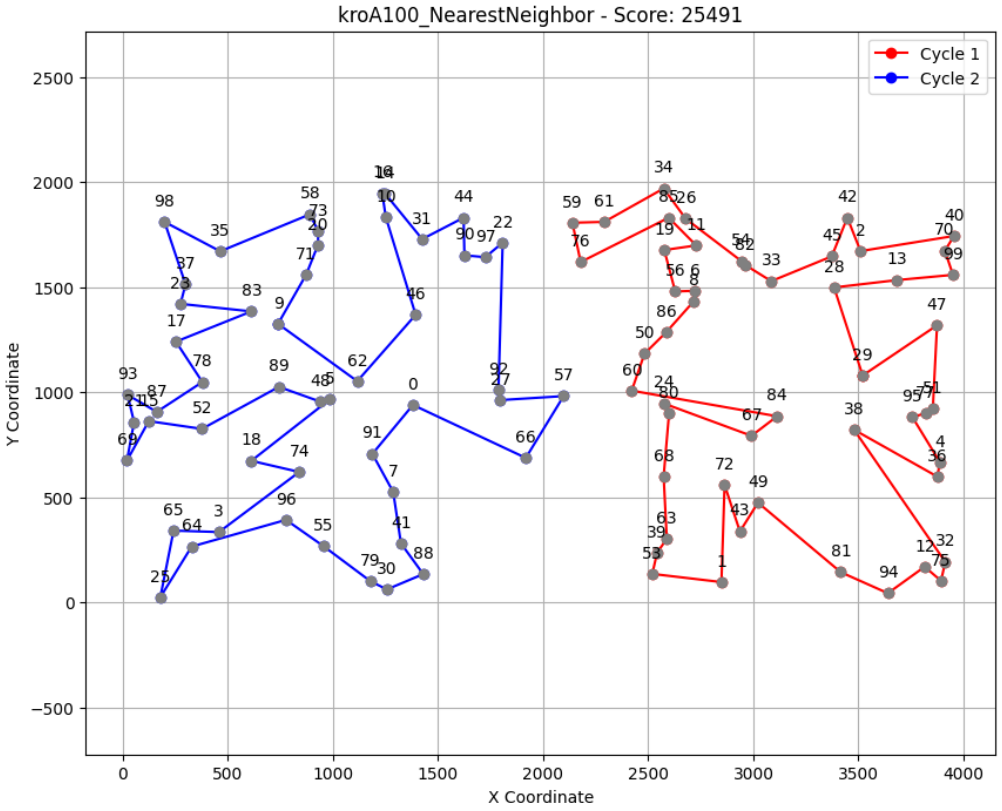
Został przeprowadzony eksperyment obliczeniowy. Dla każdej kombinacji instancja-metoda zostały policzone wartości średnia, maksimum oraz minimum. Wyniki są zaprezentowane w tabeli poniżej:

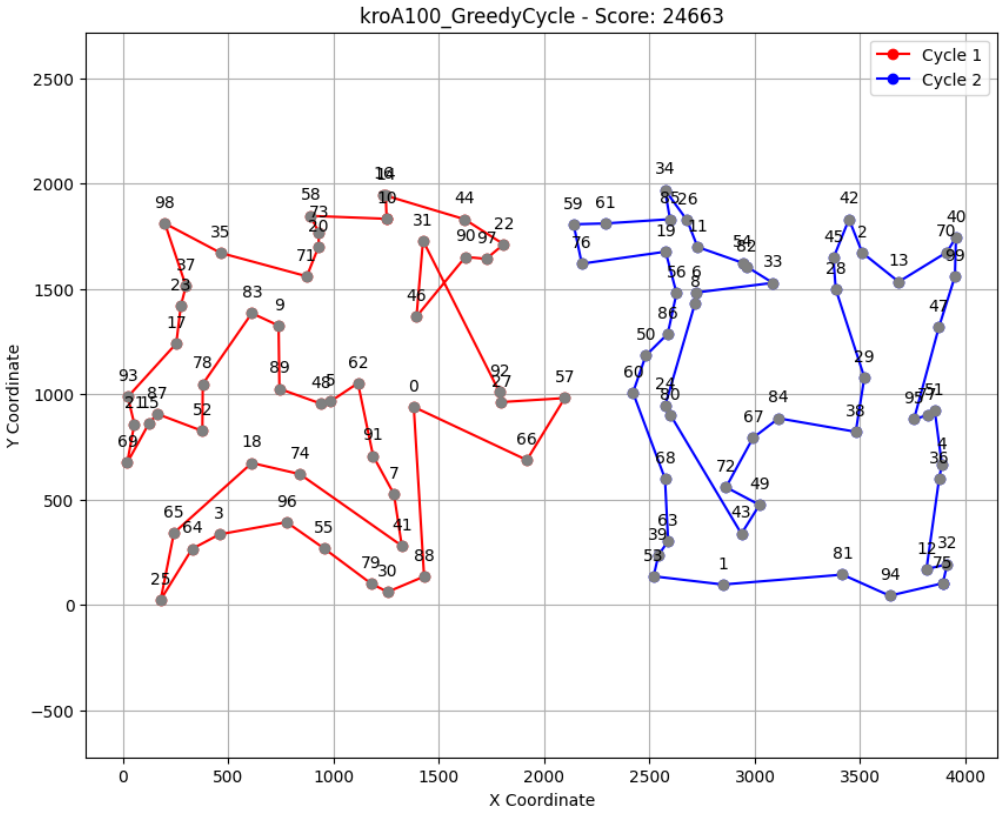
Instancja	Metoda	Mean	Min	Max
kroA100	NN	30545,57	25491	39797
kroA100	Greedy Cycle	28788,26	24663	29980
kroA100	2-Regret	26710,91	22914	29510
kroB100	NN	30674,03	26298	37496
kroB100	Greedy Cycle	28408,01	24992	30432
kroB100	2-Regret	27726,17	24172	29378

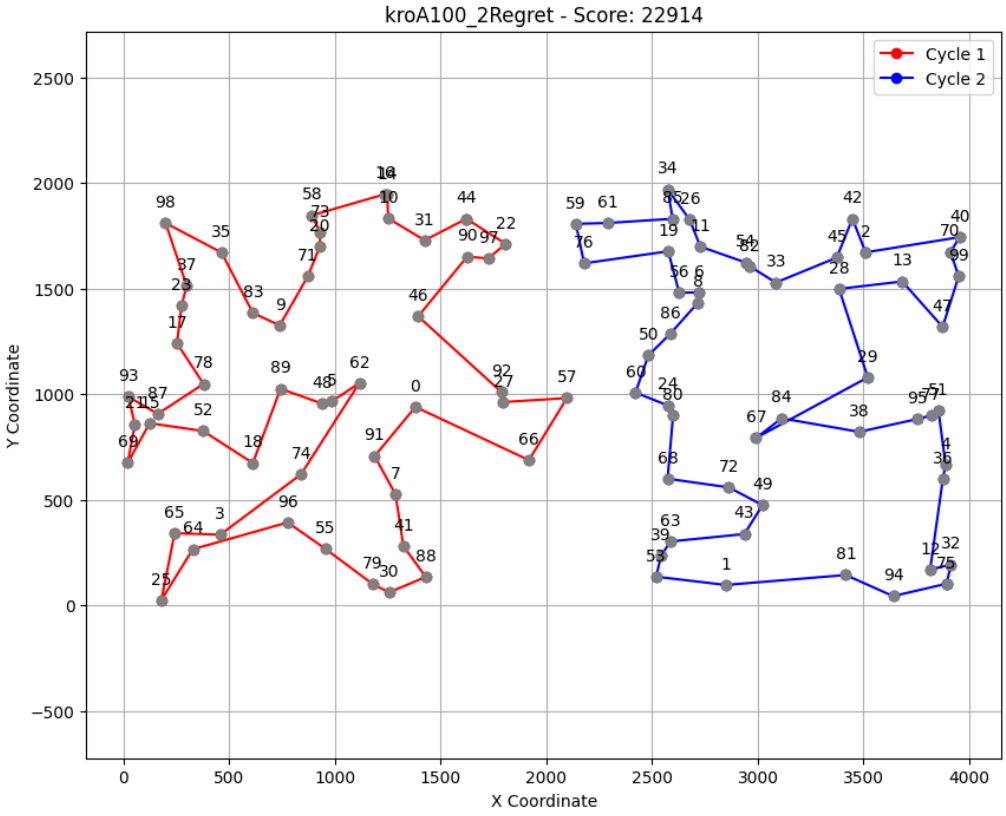
Analizując powyższe wyniki, można zauważyć, że najlepsze wyniki daje wyniki heurystyka z 2-żalem. Warto w tym miejscu nadmienić, że jest to żal ważony z wagą = 0,37. Gdy program był uruchamiany bez parametru wagi, to heurystyka z 2 żalem dawała gorsze wyniki.

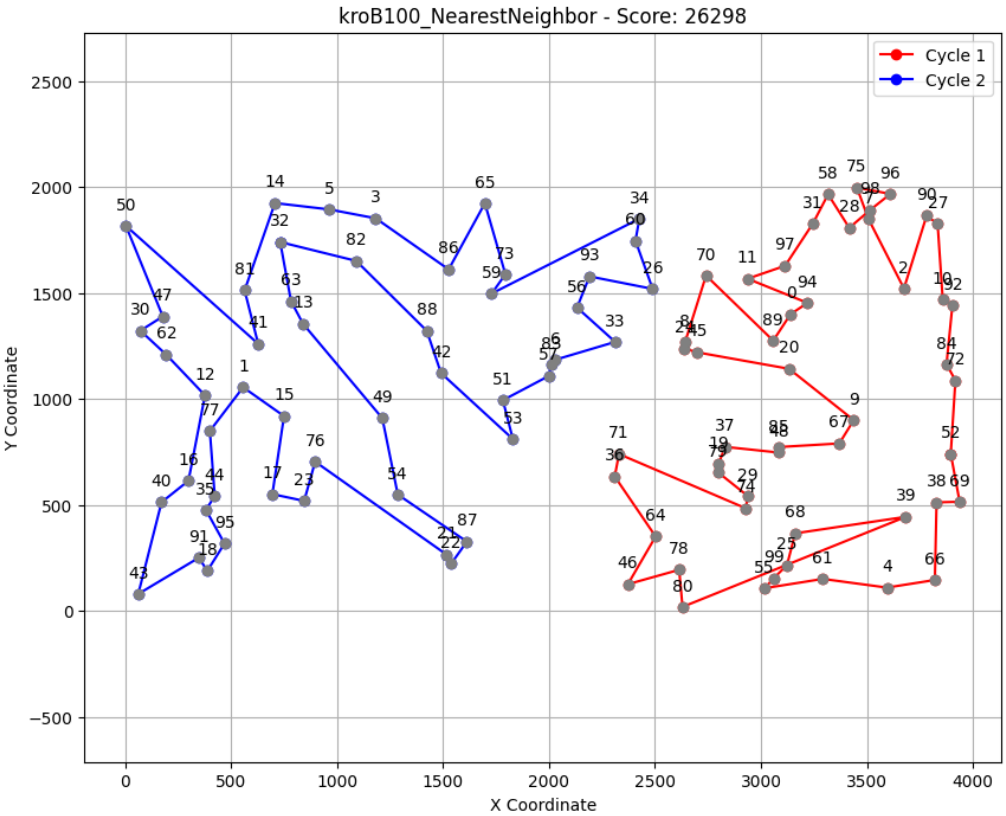
Wizualizacja najlepszych rozwiązań

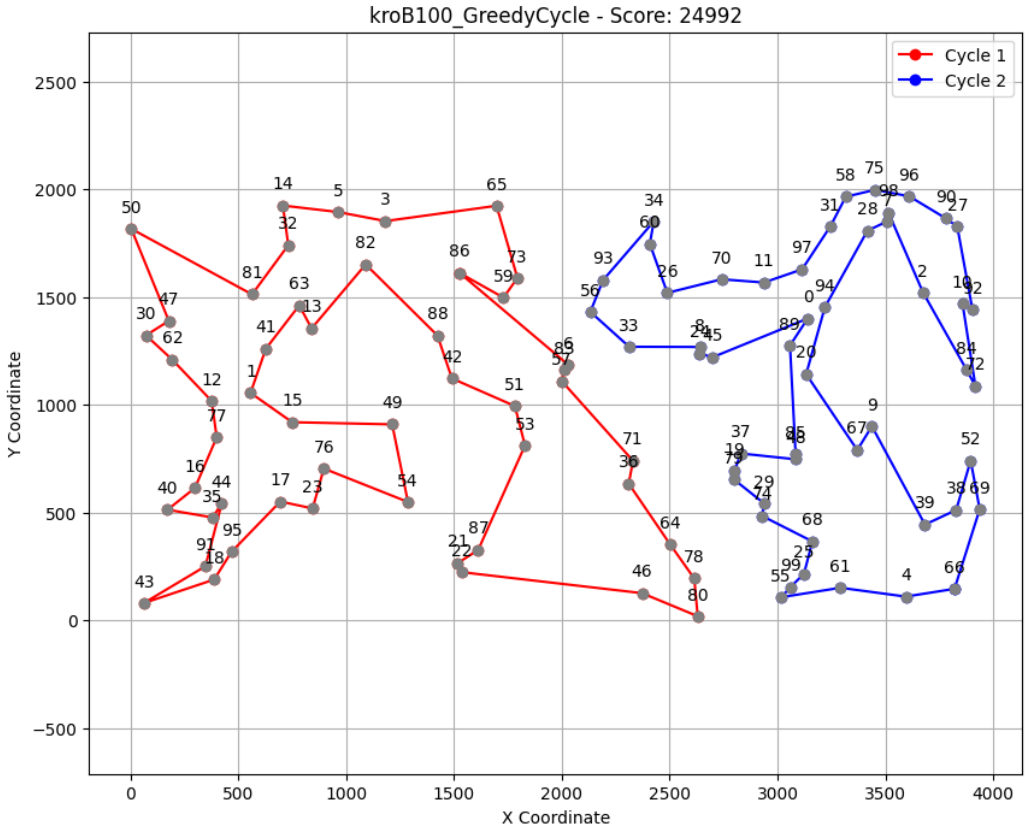
Poniżej znajduje się wizualizacja najlepszych rozwiązań:

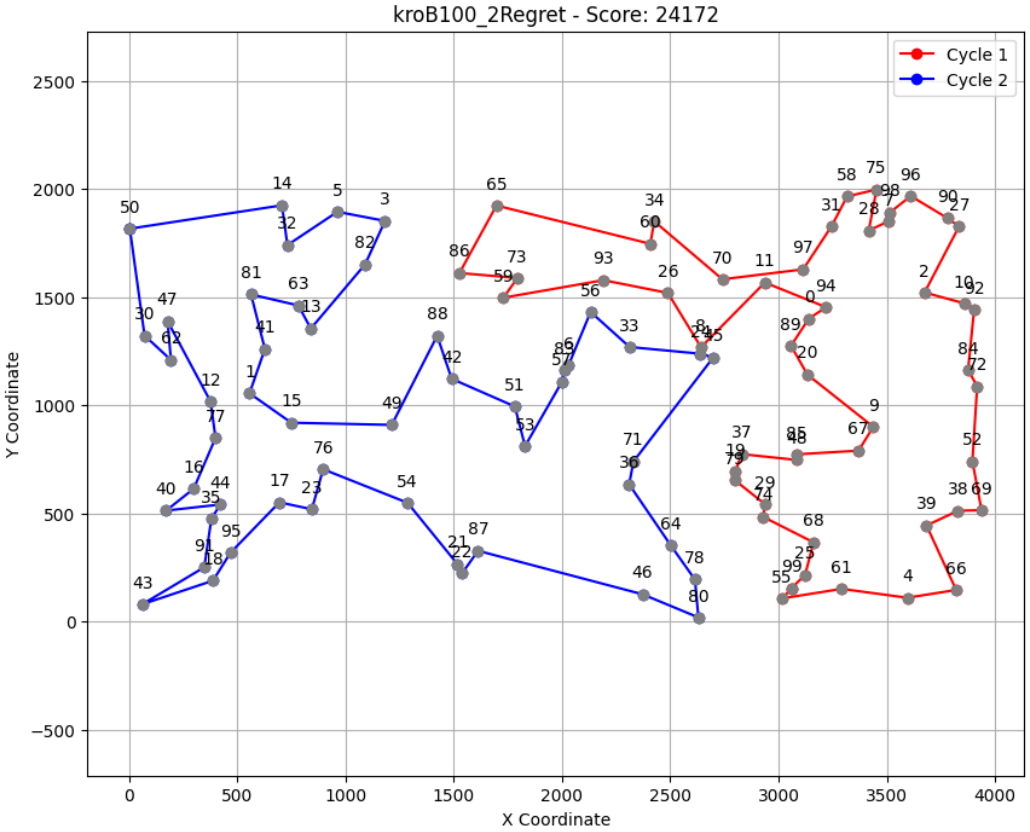





kroA100_2Regret - Score: 22914
Cycle 1
Cycle 2
Y Coordinate
X Coordinate


kroB100_NearestNeighbor - Score: 26298
Cycle 1
Cycle 2
Y Coordinate
X Coordinate





Kod programu:

Kod programu jest dostępny pod linkiem: https://github.com/Evarios/IMO