IMO – LAB 2 Lokalne przeszukiwanie

Jan Bróździak 141142

Opis zadania

Celem zadania było zaimplementowanie lokalnego przeszukiwania w wersjach stromej (steepest) i zachłannej (greedy) z dwoma różnymi rodzajami sąsiedztwa (wymiana wierzchołków w cyklu, oraz wymiana krawędzi w cyklu), startując z rozwiązań losowych oraz rozwiązań uzyskanych za pomocą jednej z heurystyk opracowanych w ramach poprzedniego zadania. Dodając do tego ruchy międzytrasowe (wymiana wierzchołków pomiędzy cyklami) w sumie zostało zaimplementowanych 8 kombinacji lokalnego przeszukiwania. Jako punkt odniesienia zaimplementowano również algorytm losowego błądzenia.

Opis zaimplementowanych algorytmów w pseudokodzie

Pseudokod dla funkcji generujących ruchy

Generowanie wszystkich możliwych wymian wierzchołków pomiędzy cyklami:

```
Function generate_all_vertex_swaps_between_cycles(cycle1, cycle2,
distance_matrix)
    Initialize empty list possible_swaps

    For each i from 0 to length(cycle1) - 2
        For each j from 0 to length(cycle2) - 2
            Calculate delta from swapping vertices between cycles
            If delta < 0 Then
                Swap vertices between cycles
                Add new cycles and delta to possible_swaps
            End If
        End For
    End For
```

Generowanie wszystkich wymian wierzchołków wewnątrz cyklu

```
Function generate_all_vertex_swaps_within_cycles(cycle1, cycle2,
distance_matrix)

    Initialize empty list possible_swaps

    ' Swaps within the first cycle
    For each i from 0 to length(cycle1) - 2
        For each j from i + 1 to length(cycle1) - 2
            Calculate delta from swapping vertices within cycle1
            If delta < 0 Then
                Swap vertices within cycle1
                Add new cycle1, cycle2, and delta to possible_swaps
            End If
        End For
    End For

    ' Swaps within the second cycle
    For each i from 0 to length(cycle2) - 2
        For each j from i + 1 to length(cycle2) - 2
            Calculate delta from swapping vertices within cycle2
            If delta < 0 Then
                Swap vertices within cycle2
                Add cycle1, new cycle2, and delta to possible_swaps
            End If
        End For
    End For

    Return possible_swaps
End Function
```

Generowanie wszystkich wymian krawędzi wewnątrz cyklu

```
function generate_all_edge_swaps_within_cycles(cycle1, cycle2,
distance_matrix):
    possible_swaps = []

    // Wymiana krawędzi w pierwszym cyklu
    for i in range(len(cycle1) - 1):
        for j in range(i + 1, len(cycle1) - 1):
            new_cycle1 = swap_edges_in_cycle(cycle1, i, j)
            score = calculate_score(new_cycle1, cycle2,
distance_matrix)
            possible_swaps.append((new_cycle1, cycle2, score))

    // Wymiana krawędzi w drugim cyklu
    for i in range(len(cycle2) - 1):
        for j in range(i + 1, len(cycle2) - 1):
            new_cycle2 = swap_edges_in_cycle(cycle2, i, j)
            score = calculate_score(cycle1, new_cycle2,
distance_matrix)
            possible_swaps.append((cycle1, new_cycle2, score))

    return possible_swaps
```

Pseudokod dla funkcji lokalnego przeszukiwania

Lokalne przeszukiwanie w wersji zachłannej (greedy)

```
Function greedy_local_search(cycle1, cycle2, distance_matrix,
move_list_generator)
    Set best_cycle1, best_cycle2 to cycle1, cycle2
    Calculate best_score for current cycles

    While True
        Get possible_moves from move_list_generator
        If no moves available, break the loop

        Optionally reverse the order of possible_moves

        Set start_index to random index in possible_moves
        Set improved to False

        For each move starting from start_index
            If move's delta < 0 Then
                Apply the move
                Update best_score
                Set improved to True
                Break the loop
            End If
        End For

        If not improved Then break the loop
    End While

    Return best_cycle1, best_cycle2, best_score
End Function
```

Uwaga do kodu: wybrany został sposób randomizacji polegający na wybraniu losowego indeksu początkowego podczas przeglądania listy ruchów, oraz losowy kierunek przeglądania listy.

Lokalne przeszukiwanie w wersji stromej (steepest)

```
Function steepest_ascent_local_search(cycle1, cycle2,
distance_matrix, move_list_generator)

    Set best_cycle1, best_cycle2 to cycle1, cycle2
    Calculate best_score for current cycles

    While True
        Get possible_moves from move_list_generator
        Set best_delta to 0
        Set best_move to None

        For each move in possible_moves
            If move's delta < best_delta Then
                Set best_delta to move's delta
                Set best_move to move's new cycles
            End If
        End For

        If best_move is not None Then
            Apply the best move
            Update best_score with best_delta
        Else
            Break the loop
        End If
    End While

    Return best_cycle1, best_cycle2, best_score
End Function
```

Algorytm losowego błądzenia

```
Function randomized_optimization(cycle1, cycle2, distance_matrix,
time_limit)
    Set best_cycle1, best_cycle2 to cycle1, cycle2
    Calculate best_score for current cycles
    Set start_time to current time

    While current time is less than start_time + time_limit
        Get possible_moves from generator
        If no moves available, break the loop

        Select a random move from possible_moves
        Calculate score for new cycles from random move

        If score is better than best_score Then
            Update best_cycle1, best_cycle2, and best_score
        End If
    End While

    Return best_cycle1, best_cycle2, best_score
End Function
```

Przebieg eksperymentu

Dla każdej instancji została wygenerowana para rozwiązań startowych – jedno losowe, oraz jedno najlepsze po 100 przebiegach heurystyki 2 regret. Dla każdej kombinacji instancja-rozwiązanie startowe-sąsiedztwo-metoda + losowe błądzenie eksperyment został uruchomiony 100 razy

Wyniki eksperymentu obliczeniowego

```
df_results_A
```

	Starting cycles	Neighbourhood	Method	Min	Max	Mean	Min Time	Max Time	Mean Time
0	Random	Vertices	Greedy	34591	53212	42619.50	1.413060	2.544874	2.037524
1	Random	Vertices	Steep	35686	56783	44288.85	0.561871	1.091928	0.758007
2	Random	Edges	Greedy	25937	31794	28322.06	1.326293	1.893476	1.632402
3	Random	Edges	Steep	25441	31073	28134.38	0.452118	0.596820	0.522132
4	Random	-	Random search	153491	191572	171116.85	0.522174	0.563316	0.536456
5	2-regret	Vertices	Greedy	22733	28008	25558.69	0.010439	0.186292	0.081892
6	2-regret	Vertices	Steep	22733	27447	25500.45	0.011129	0.112948	0.059983
7	2-regret	Edges	Greedy	22278	27941	25060.38	0.018308	0.155076	0.073478
8	2-regret	Edges	Steep	22278	27380	24789.79	0.017611	0.099121	0.057288
9	2-regret	-	Random search	29527	63244	43775.48	0.060669	0.096404	0.072269

```
df_results_B ···
```

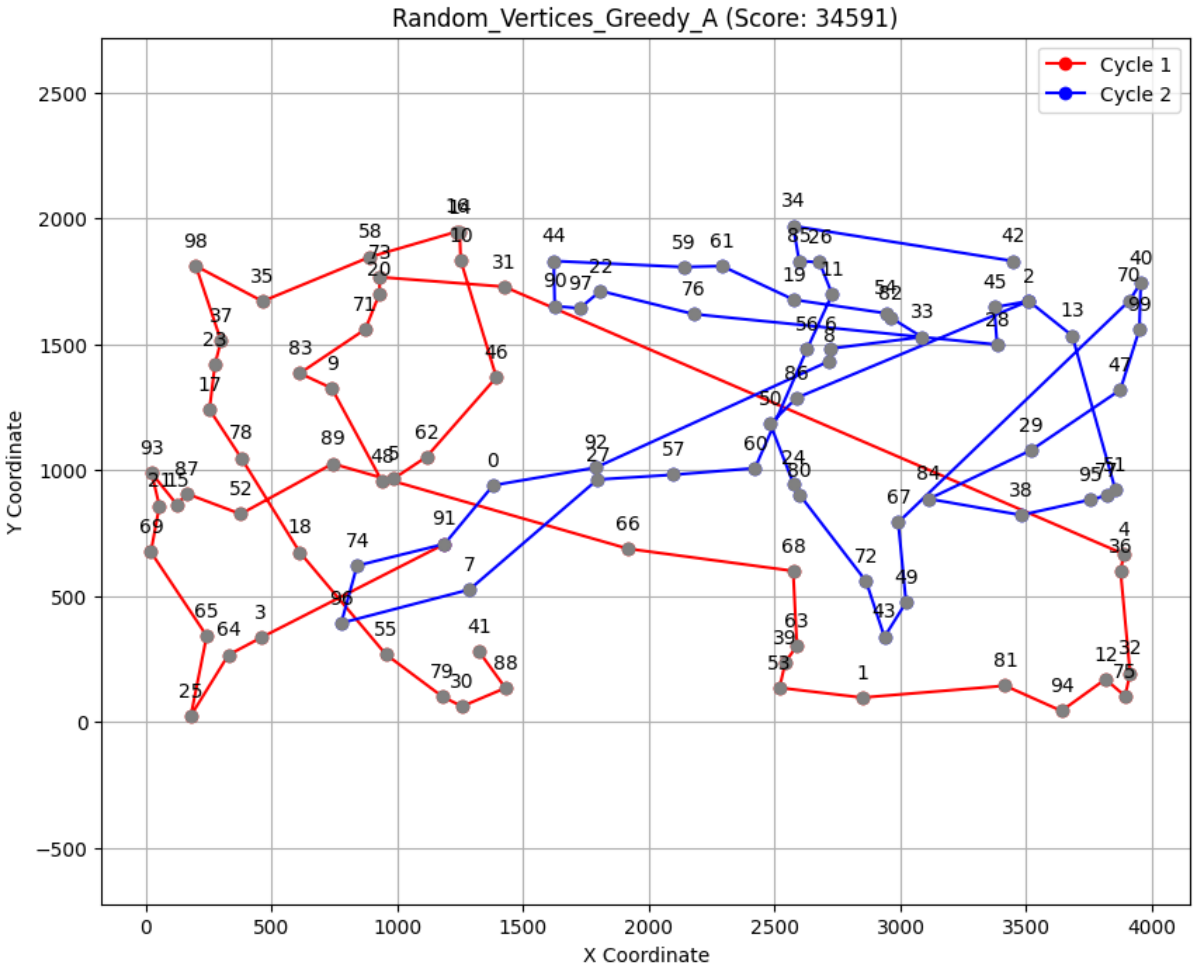
	Starting cycles	Neighbourhood	Method	Min	Max	Mean	Min Time	Max Time	Mean Time
0	Random	Vertices	Greedy	33547	51716	42799.95	1.511669	2.495910	1.968232
1	Random	Vertices	Steep	35397	53864	44434.21	0.512338	0.939039	0.713906
2	Random	Edges	Greedy	26086	32699	29073.44	1.275954	1.996860	1.596114
3	Random	Edges	Steep	26280	31653	28917.04	0.438531	0.602788	0.517051
4	Random	-	Random search	145824	184921	169572.32	0.517260	0.559660	0.531988
5	2-regret	Vertices	Greedy	22870	28569	25796.46	0.012510	0.200887	0.075673
6	2-regret	Vertices	Steep	23146	28270	25682.73	0.022533	0.107889	0.059849
7	2-regret	Edges	Greedy	22610	28419	25614.90	0.018214	0.164077	0.081776
8	2-regret	Edges	Steep	22741	28237	25380.55	0.017179	0.098478	0.064917
9	2-regret	-	Random search	27789	59344	44529.79	0.064933	0.095176	0.073936

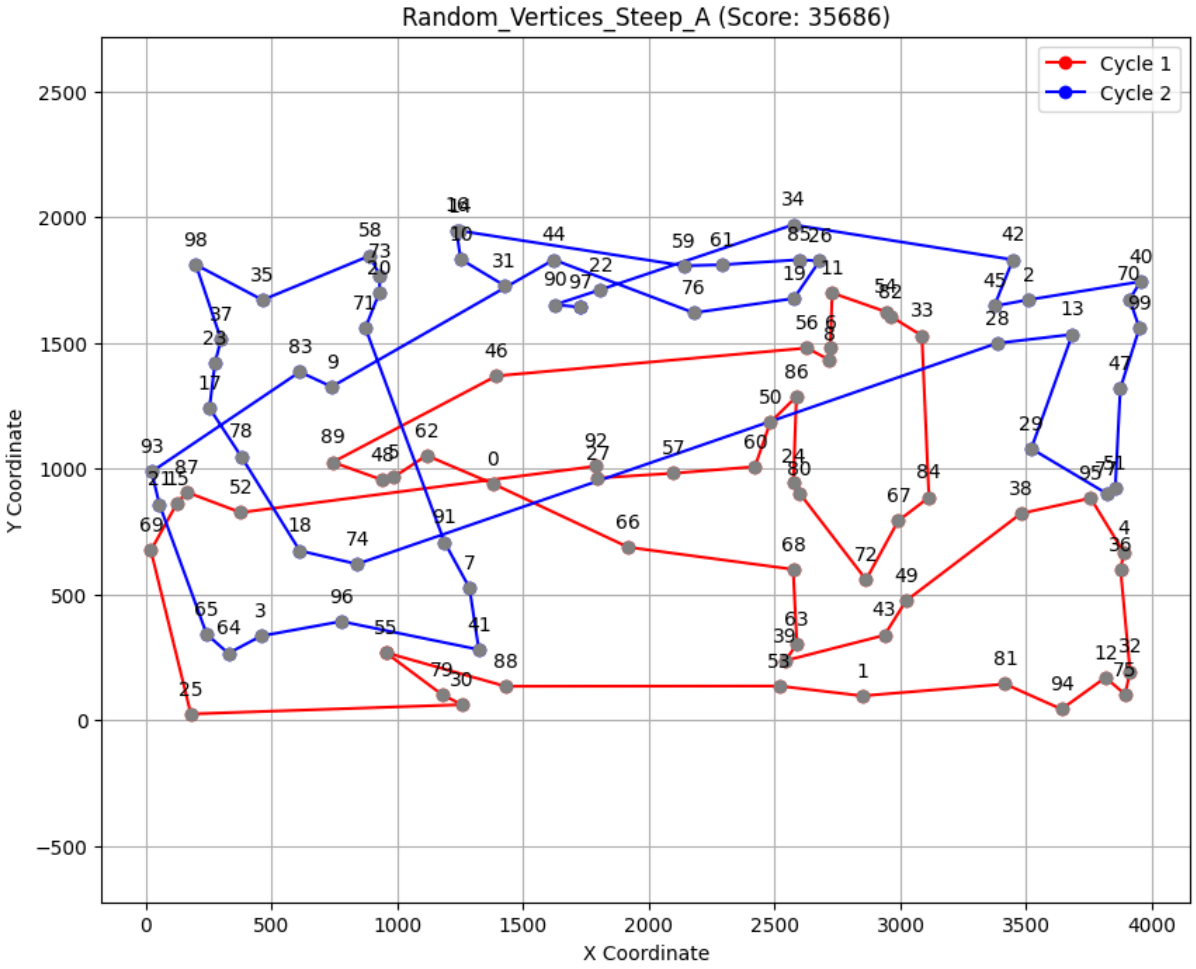
Uwaga do czasu greedy: powinien być krótszy, ale optymalizacja zastosowana wcześniej, tak naprawdę przegląda wszystkie ruchy jak steep. Zostawiłem to jako ciekawostkę – widać tutaj w drugą stronę, że lepiej zrobić kilka dużych kroków, niż wiele małych.

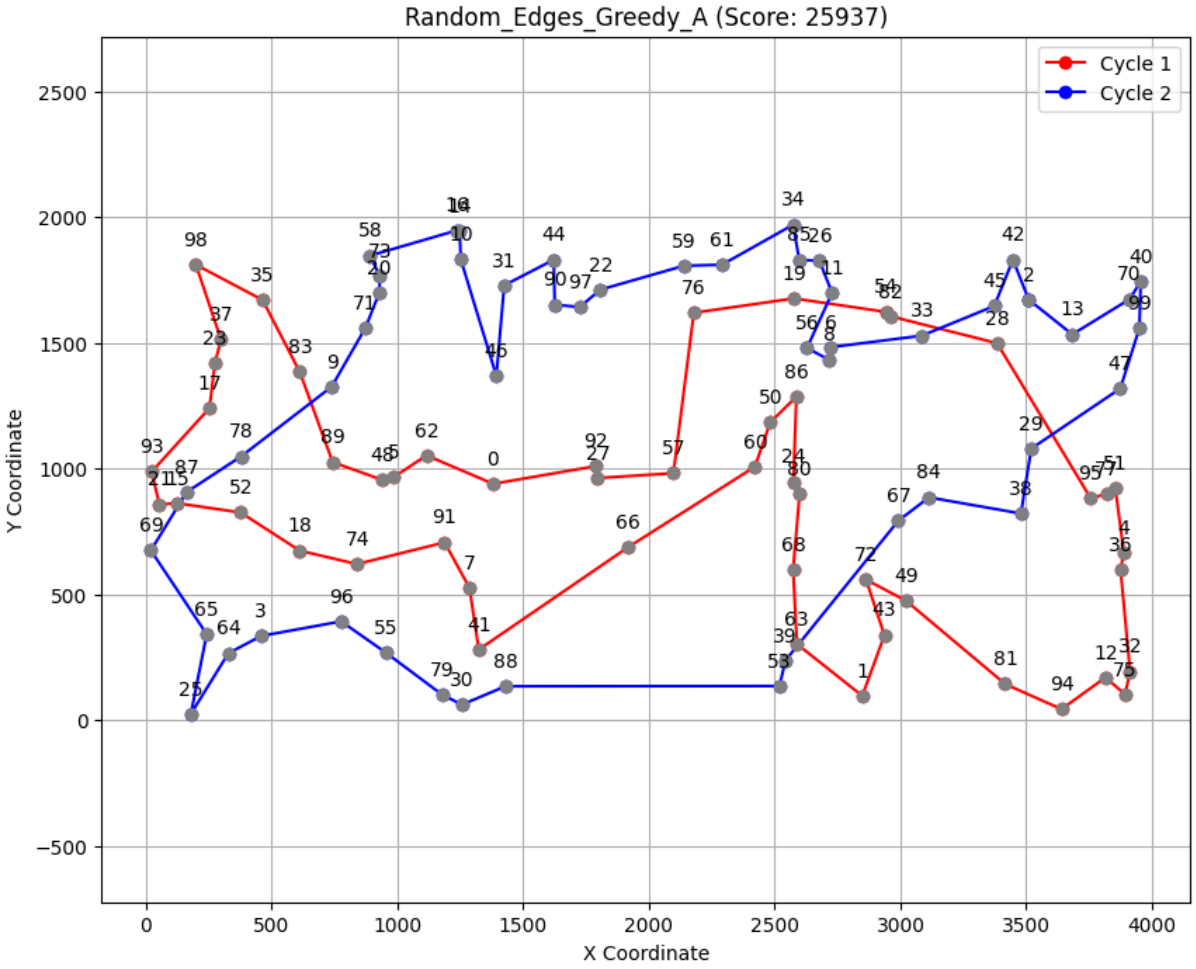
Wnioski

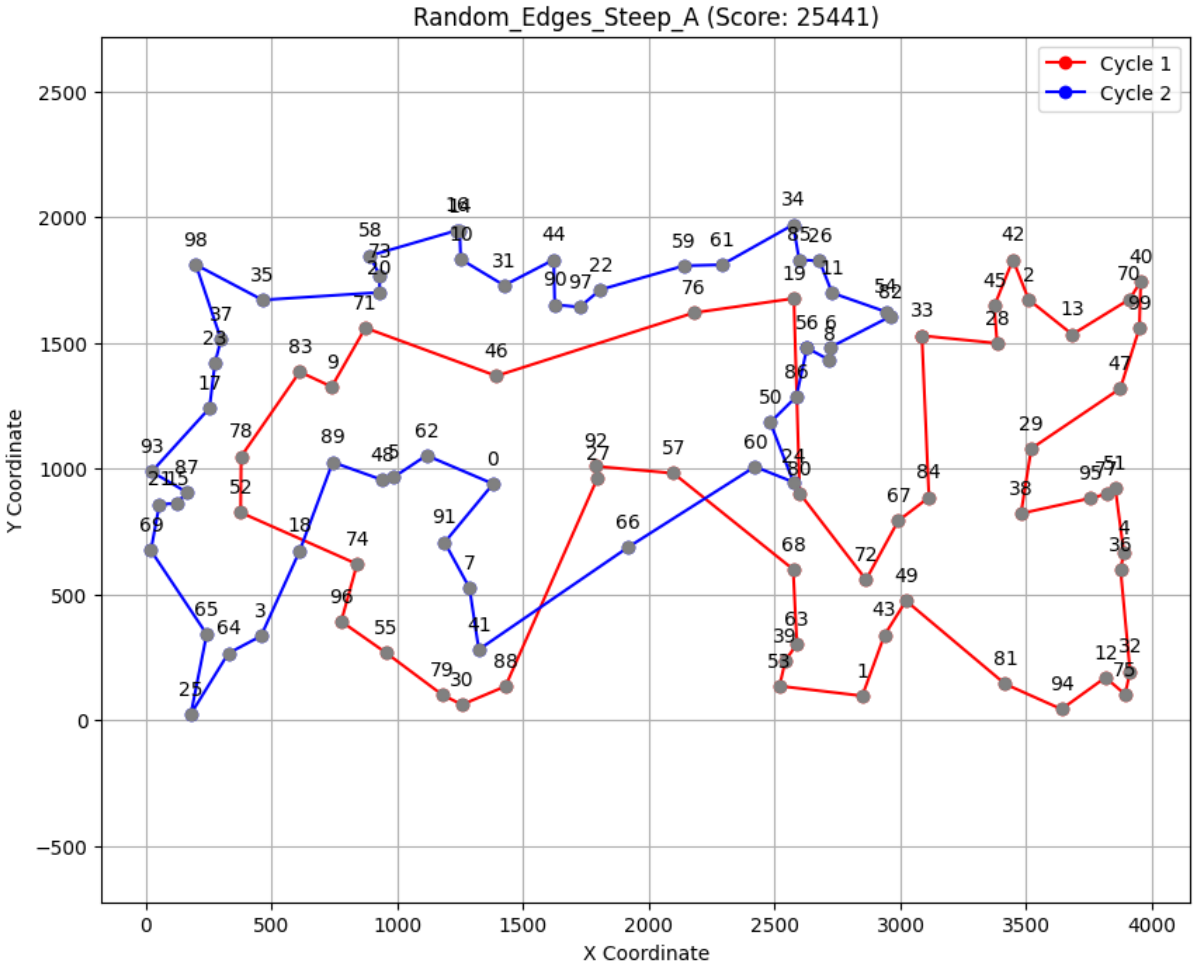
Najlepsze wyniki daje algorytm Steep dla sąsiedztwa z krawędziami. Wymiana krawędzi jest efektywniejsza zarówno dla algorytmu Steep i Greedy.

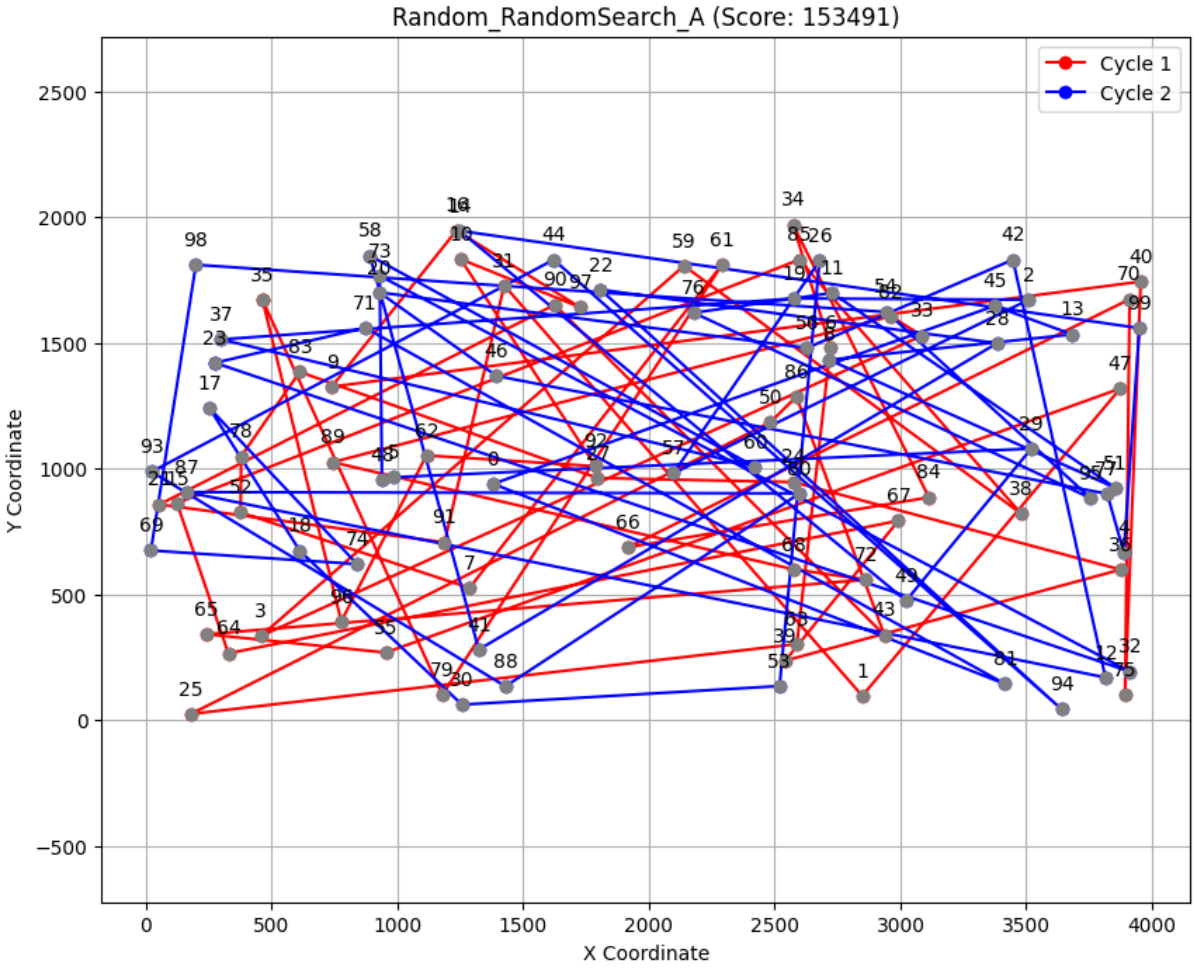
Wizualizacja najlepszych rozwiązań:

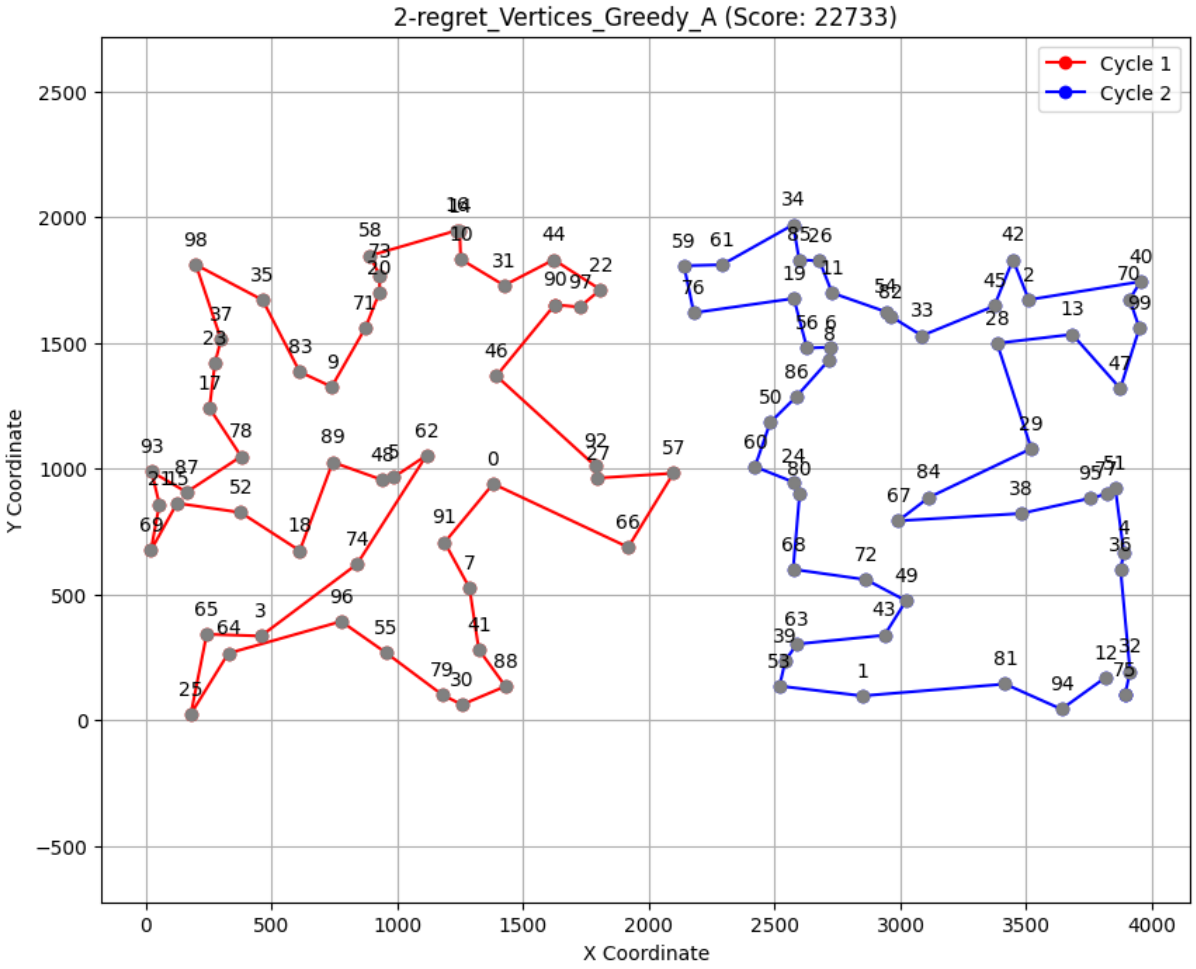


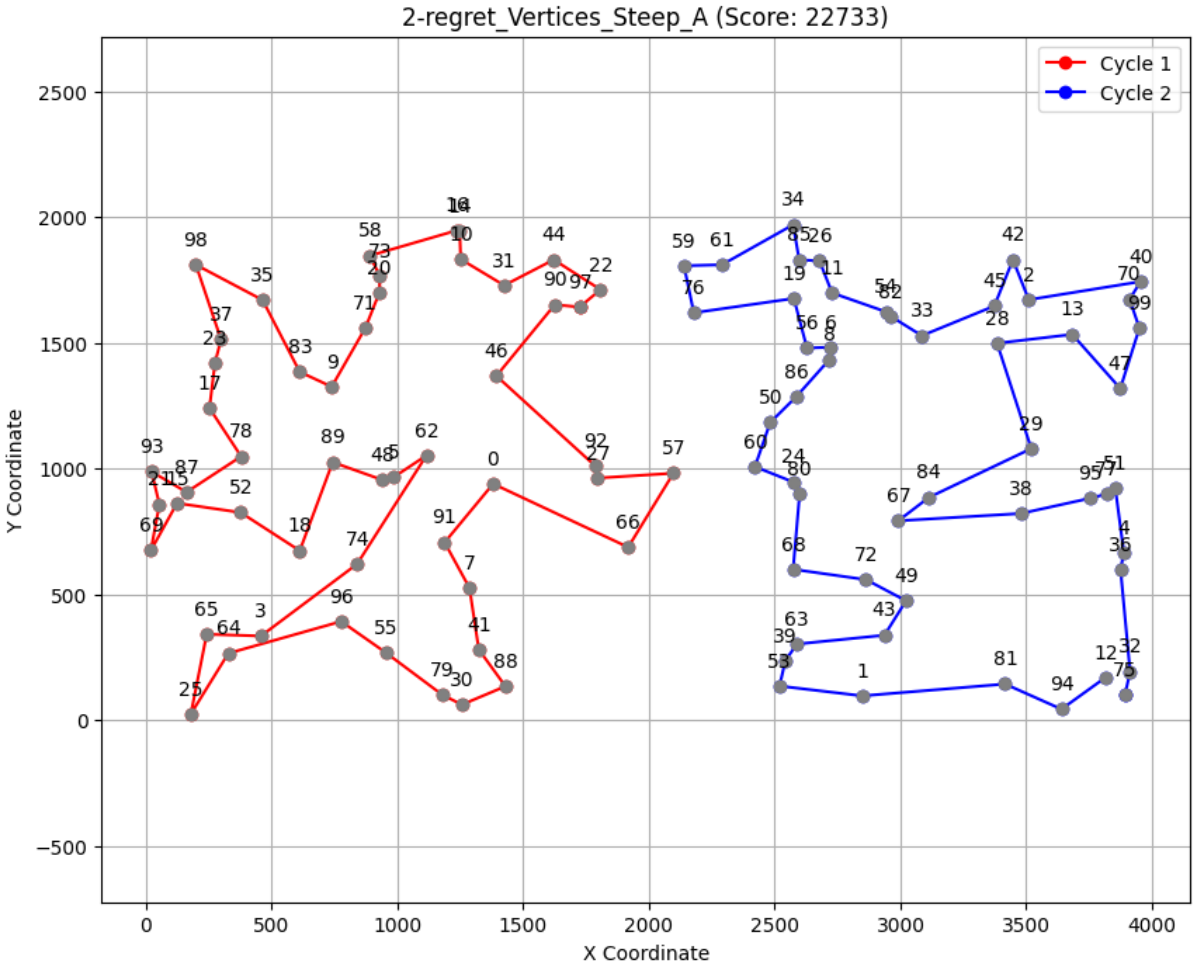

Random_Vertices_Steep_A (Score: 35686)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

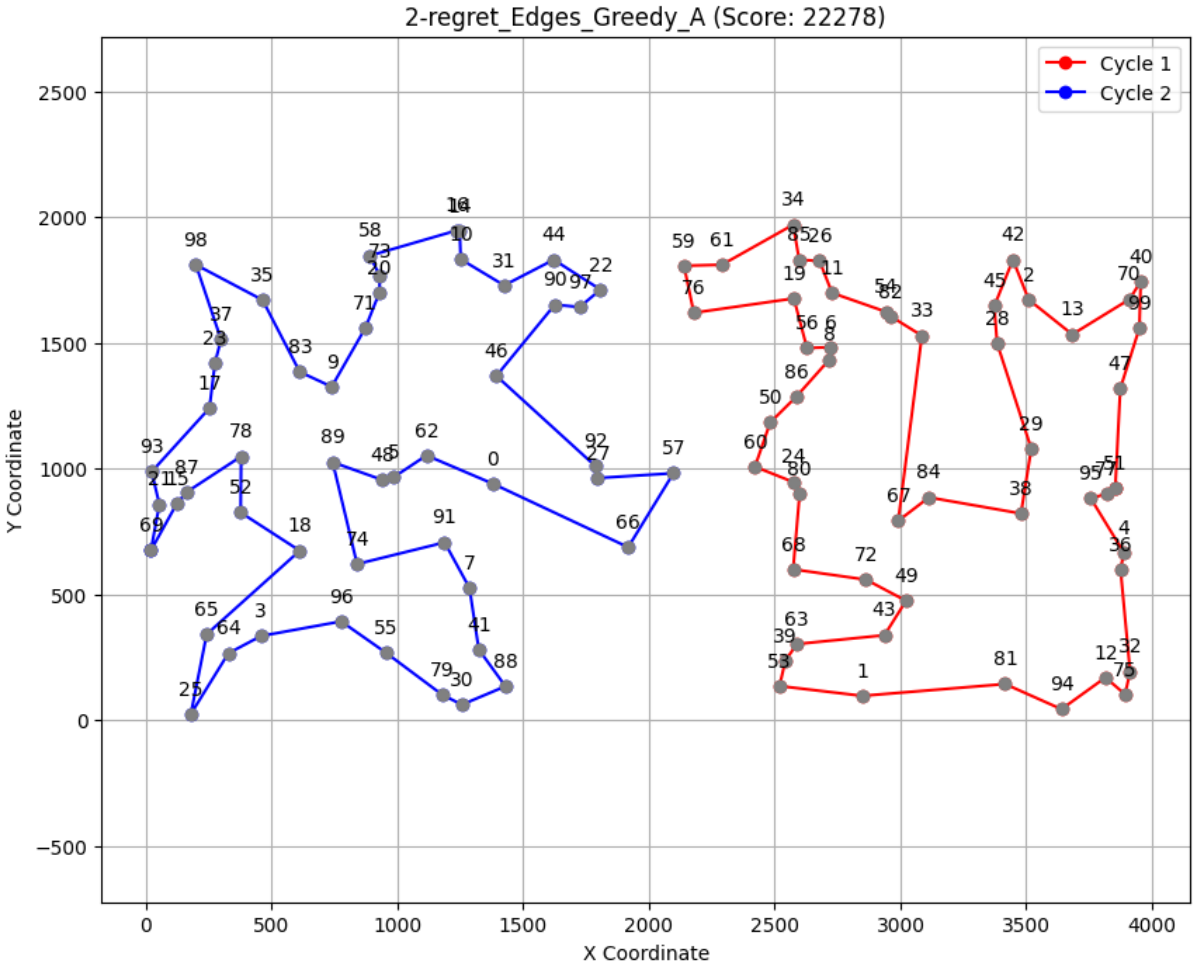

Random_Edges_Greedy_A (Score: 25937)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

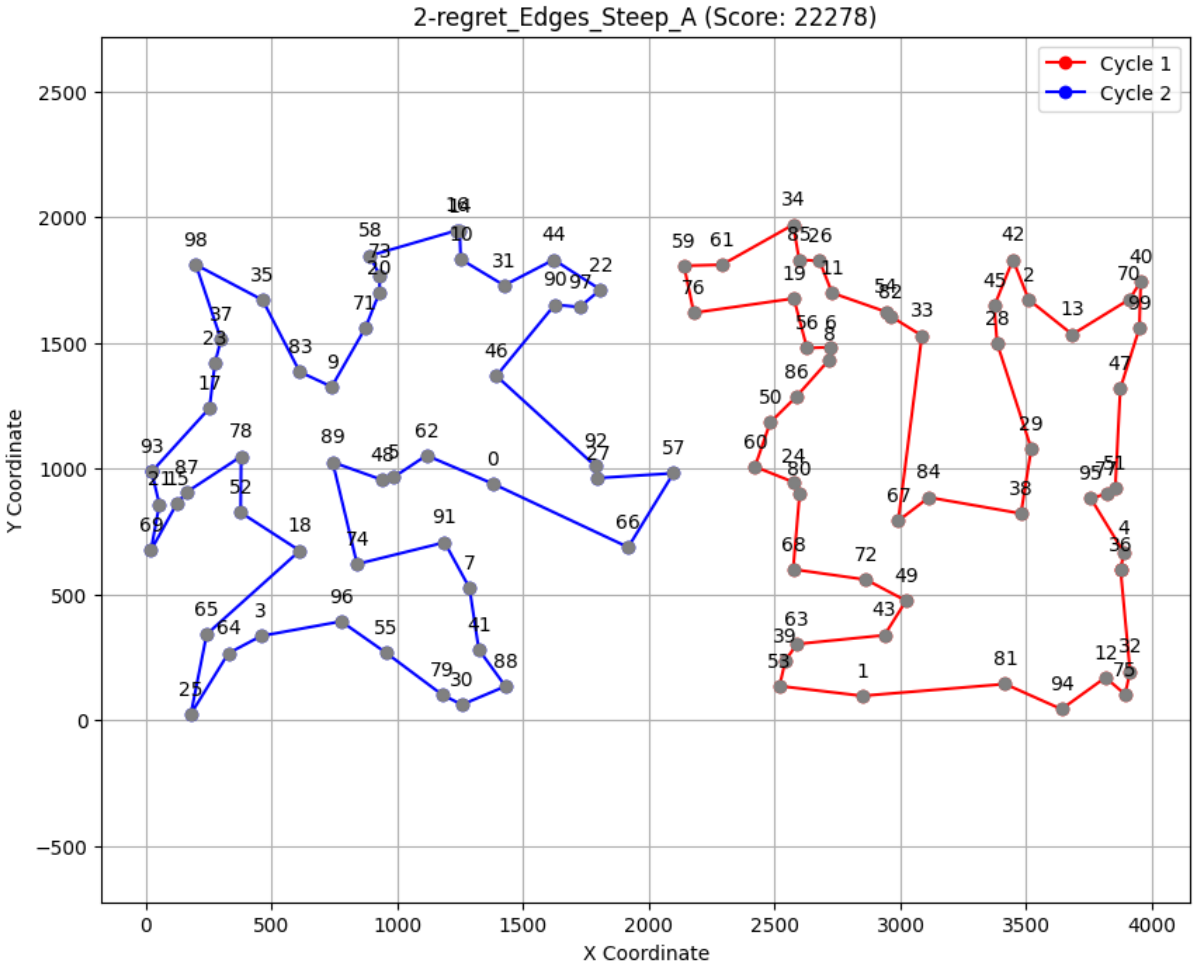

Random_Edges_Steep_A (Score: 25441)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

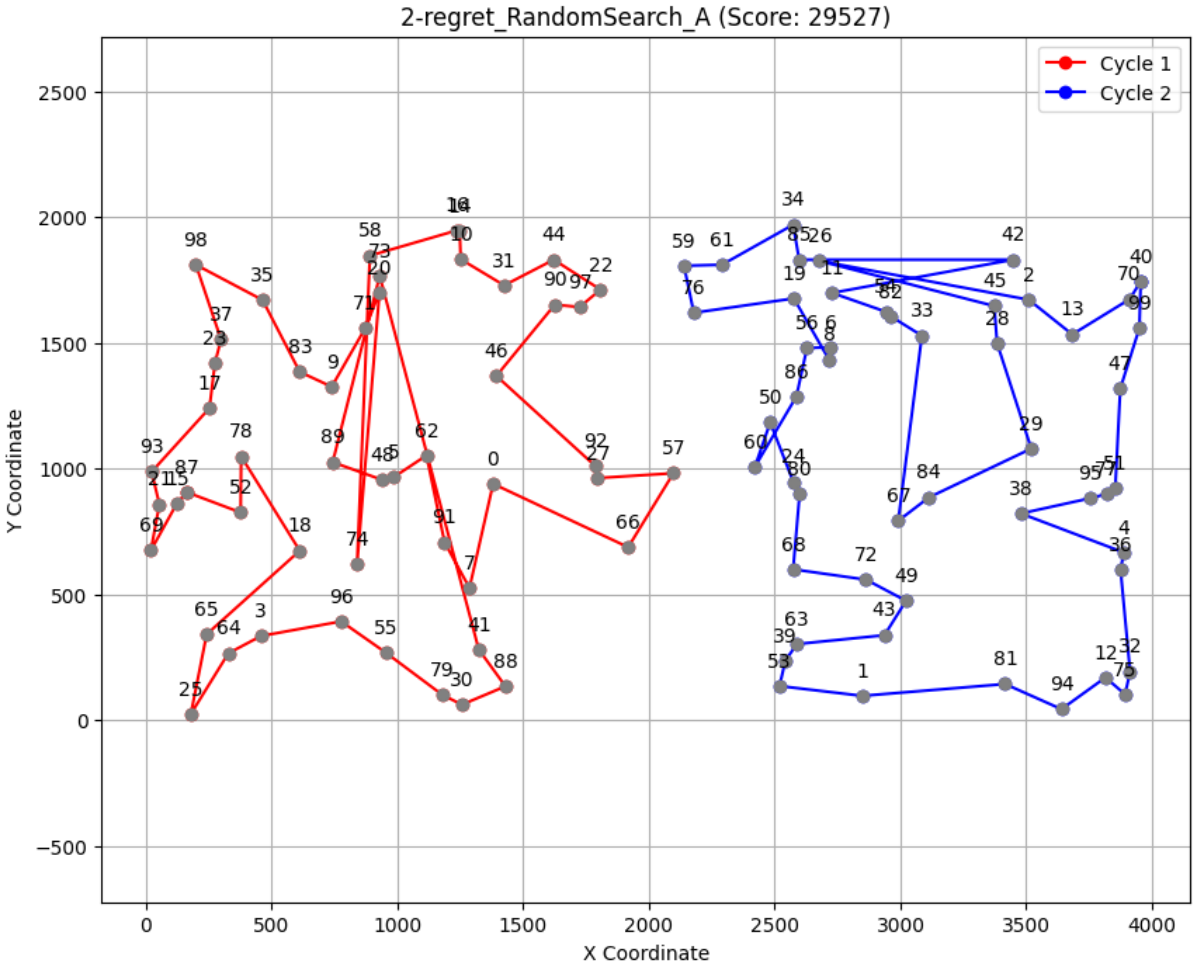

Random_RandomSearch_A (Score: 153491)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

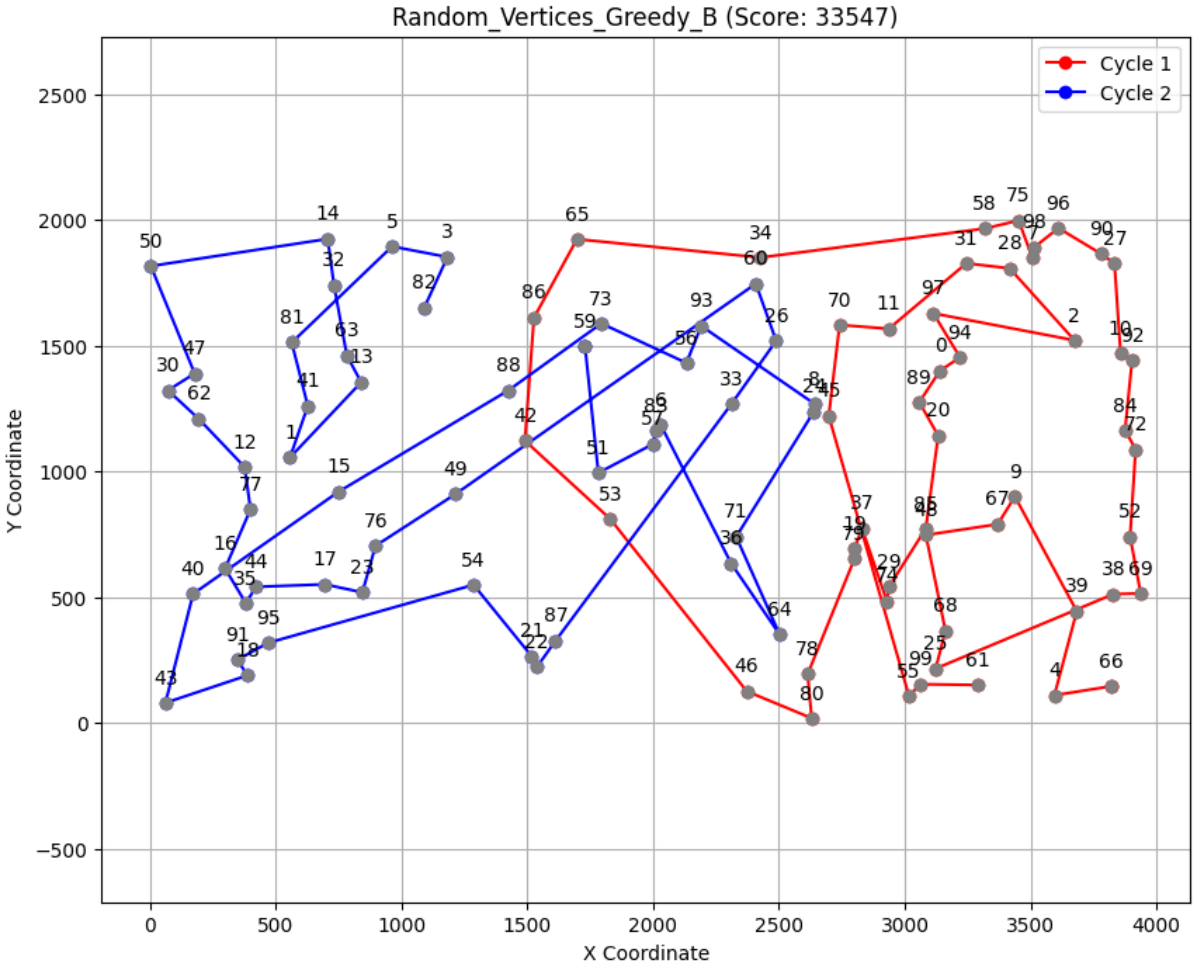

2-regret_Vertices_Greedy_A (Score: 22733)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

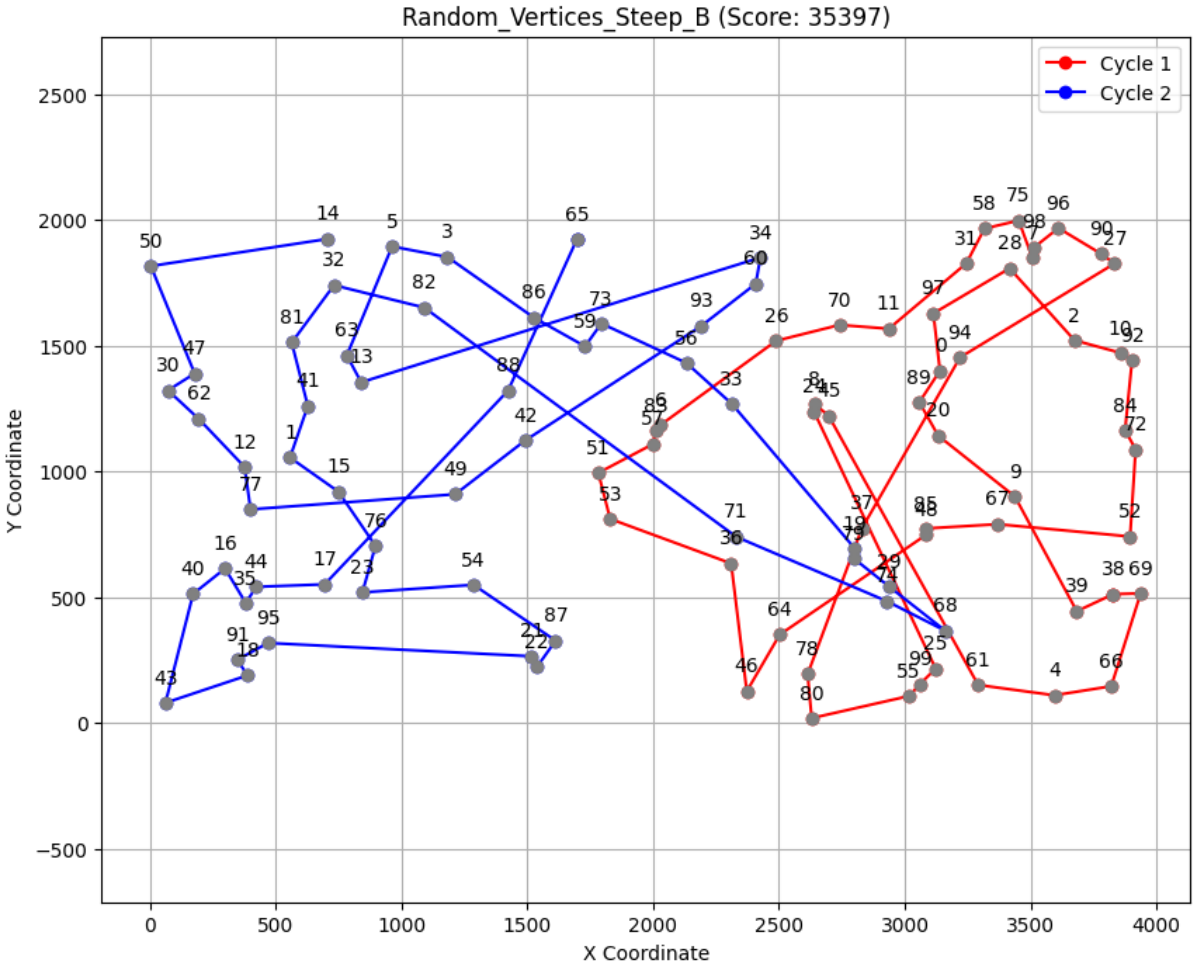

2-regret_Vertices_Steep_A (Score: 22733)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

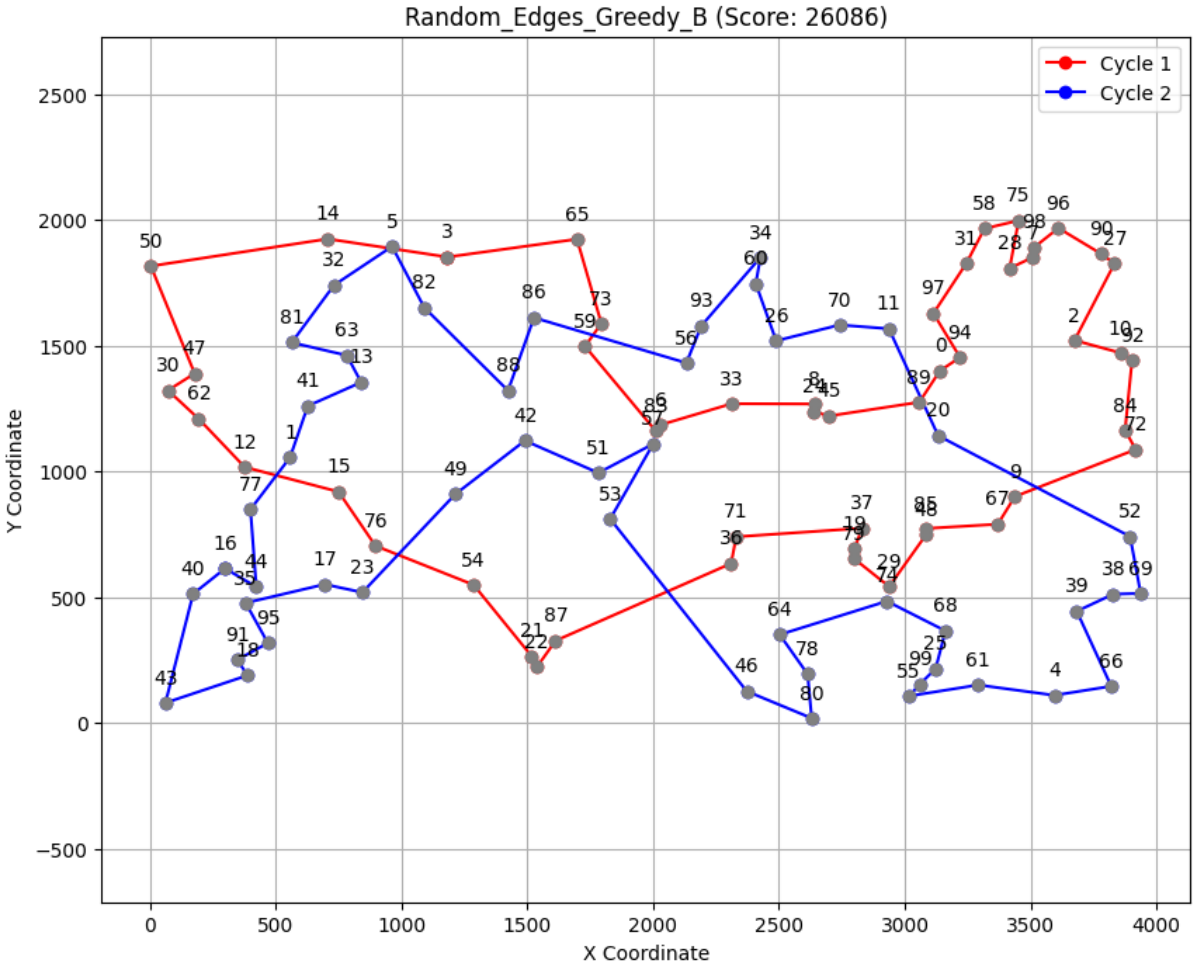

2-regret_Edges_Greedy_A (Score: 22278)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

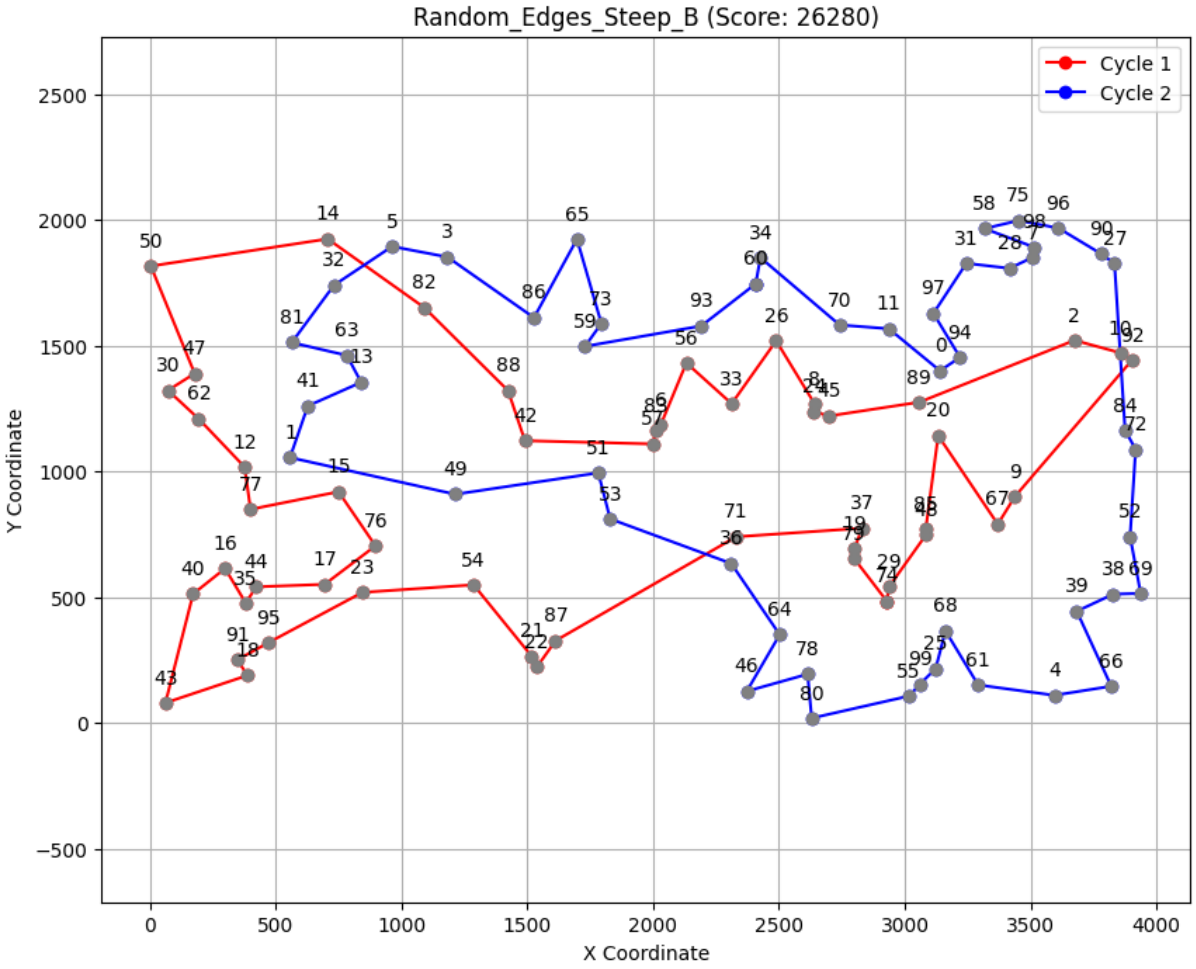

2-regret_Edges_Steep_A (Score: 22278)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

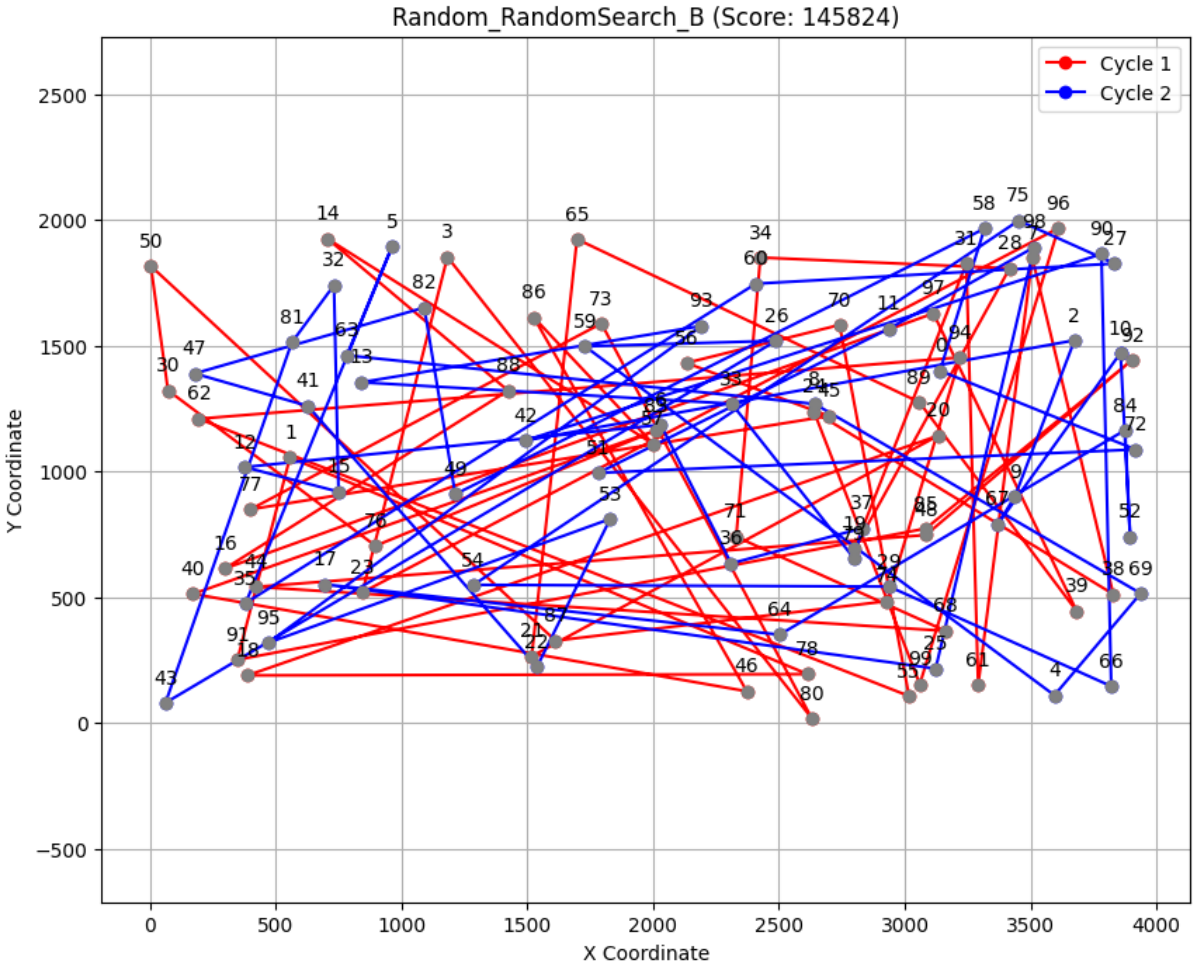

2-regret_RandomSearch_A (Score: 29527)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

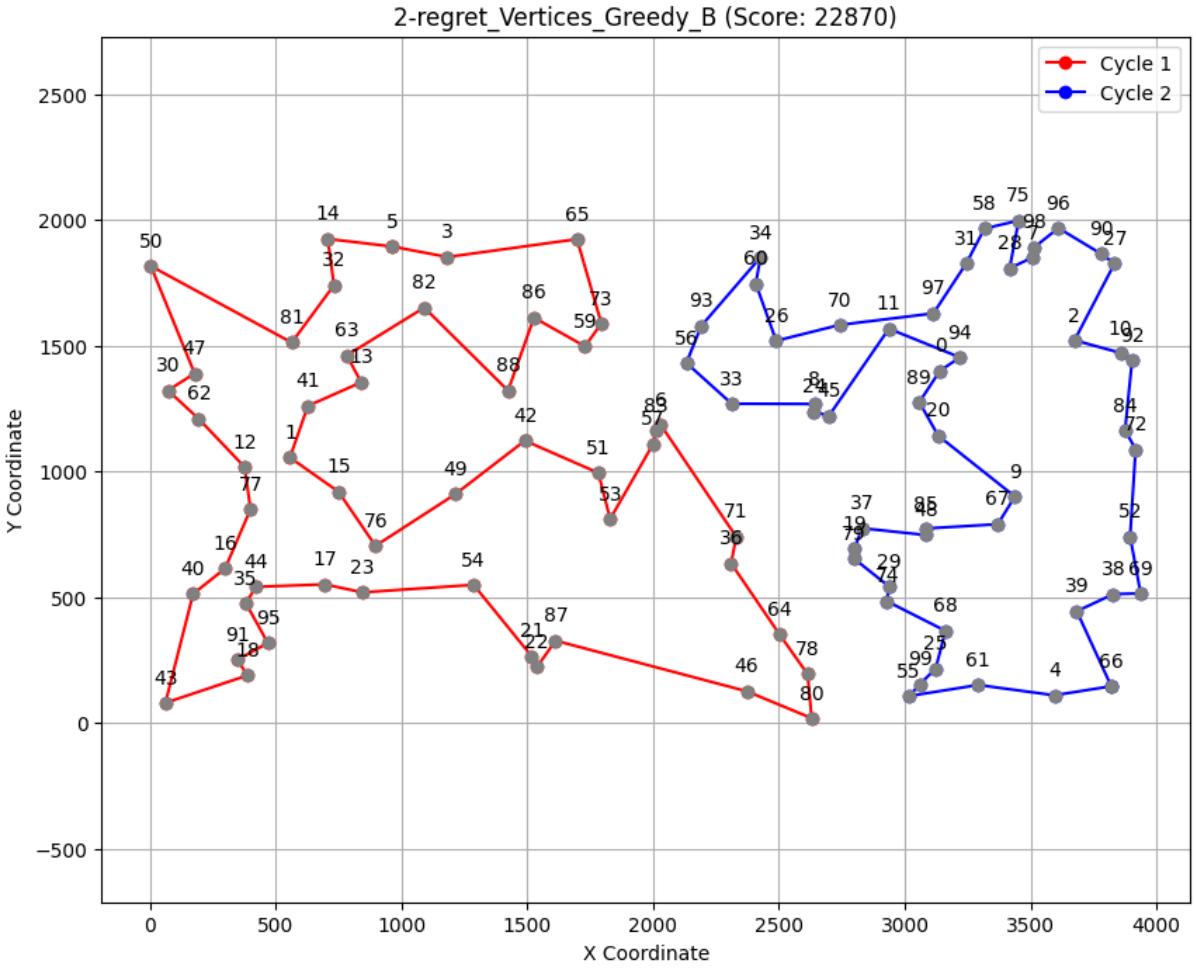

Random_Vertices_Greedy_B (Score: 33547)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

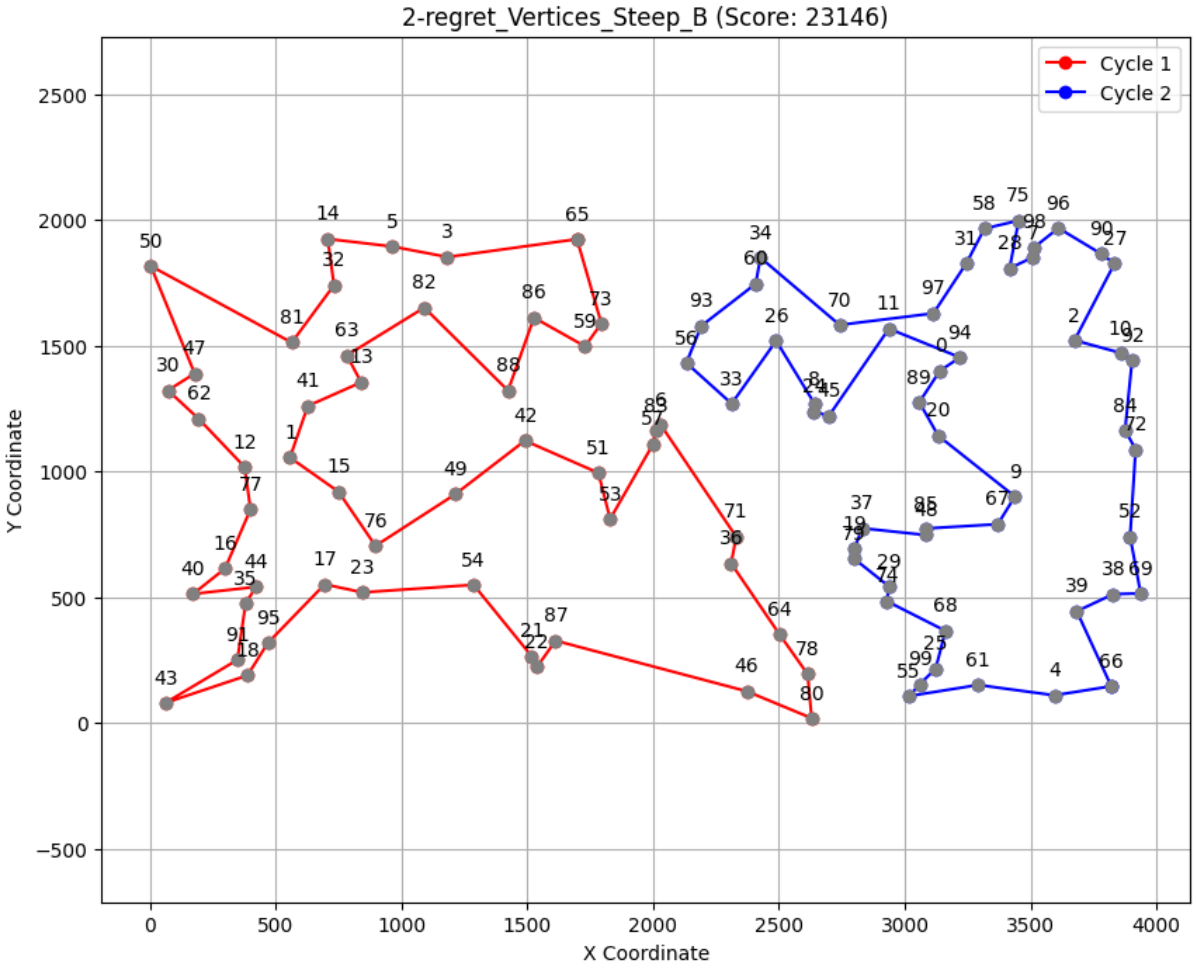

Random_Vertices_Steep_B (Score: 35397)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

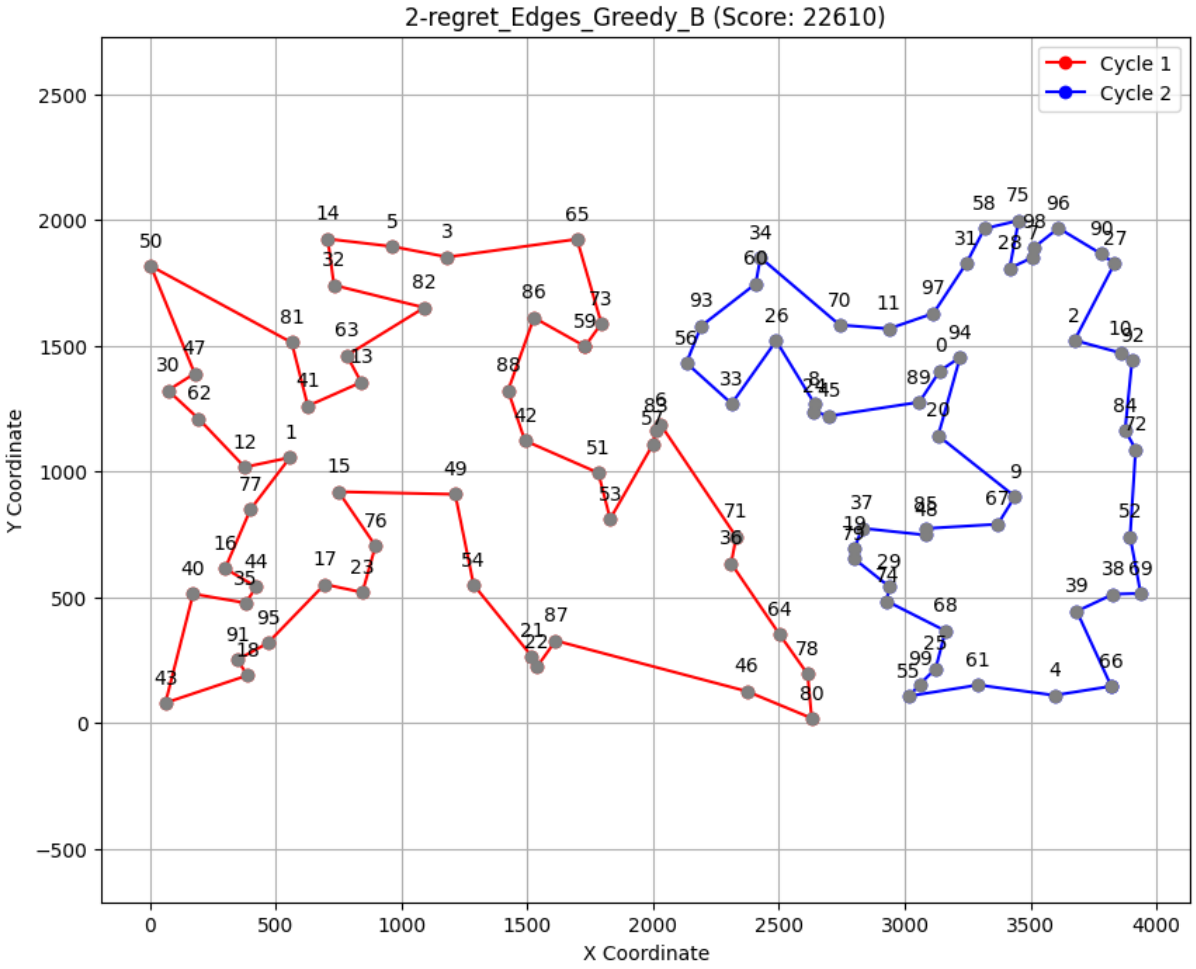

Random_Edges_Greedy_B (Score: 26086)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
0
500
1000
1500
2000
2500
3000
3500
4000
−500
0
500
1000
1500
2000
2500

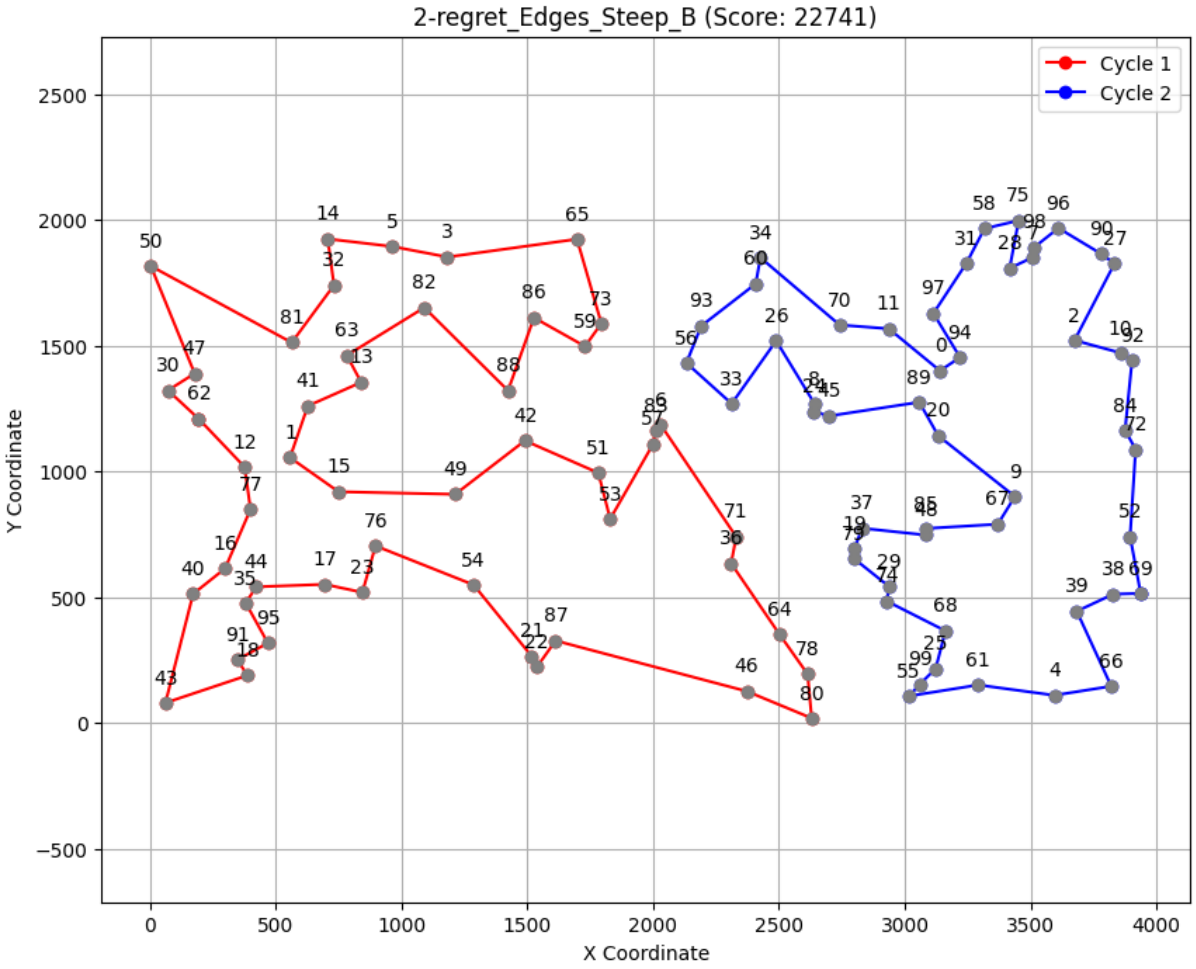

Random_Edges_Steep_B (Score: 26280)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000

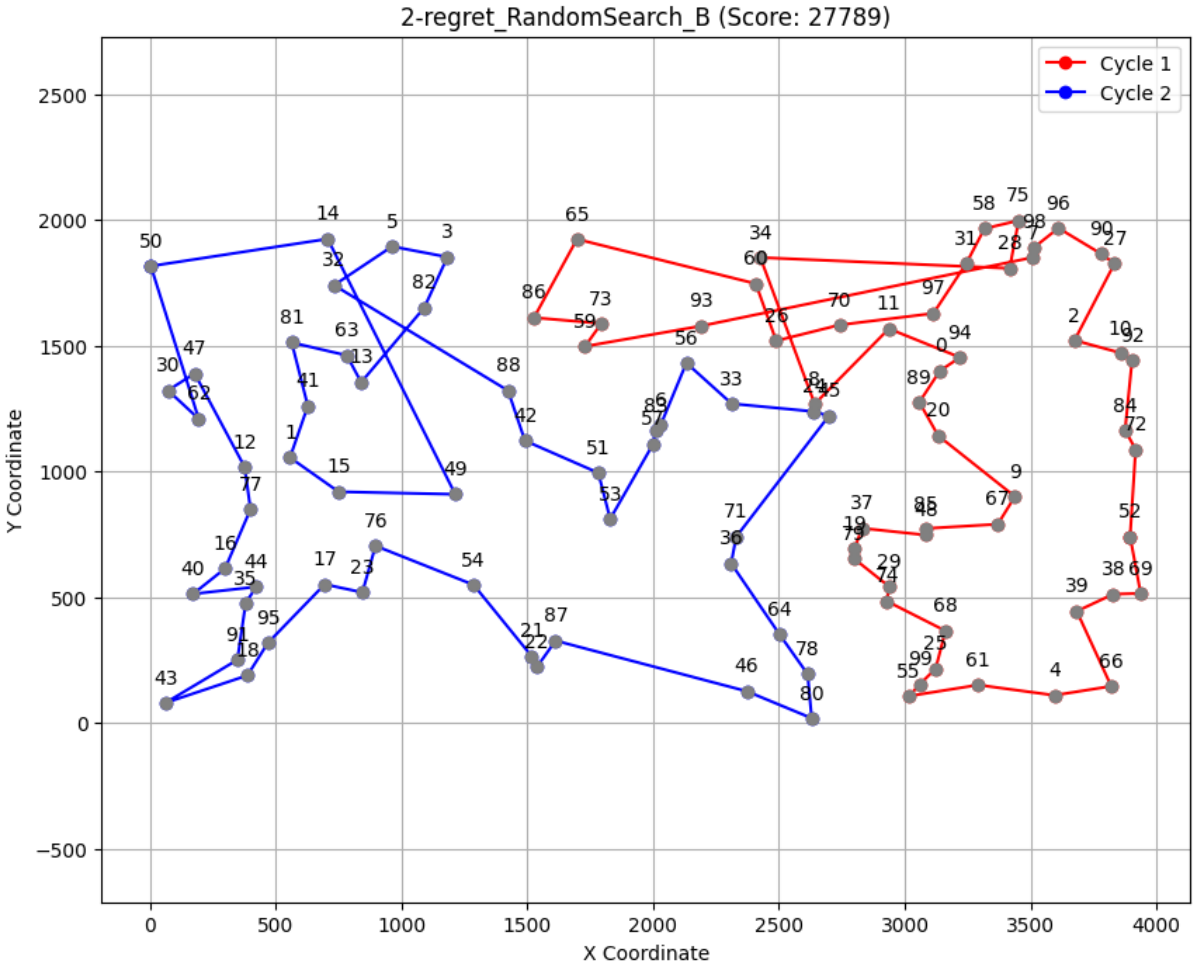

Random_RandomSearch_B (Score: 145824)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000


2-regret_Vertices_Greedy_B (Score: 22870)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000


2-regret_Vertices_Steep_B (Score: 23146)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000


2-regret_Edges_Greedy_B (Score: 22610)
Cycle 1
Cycle 2
X Coordinate
Y Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000


2-regret_Edges_Steep_B (Score: 22741)
Cycle 1
Cycle 2
Y Coordinate
X Coordinate
2500
2000
1500
1000
500
0
−500
0
500
1000
1500
2000
2500
3000
3500
4000



Kod programu

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