IMO – LAB 6 Testy globalnej wypukłości

Jan Bróździak 141142

Opis zadania

Celem zadania było wygenerowanie dla każdej instancji 1000 losowych optimów lokalnych, tj. rozwiązań uzyskanych z losowych rozwiązań startowych po zastosowaniu lokalnego przeszukiwania w wersji zachłannej. Następnie dla każdego z tych rozwiązań należało policzyć podobieństwo do najlepszego rozwiązania (najlepszego spośród tego 1000 lub najlepszego wygenerowanego dowolną inną metodą) i średnie podobieństwo dla wszystkich pozostałych optimów lokalnych z tego zbioru.

Opis zaimplementowanych algorytmów w pseudokodzie

Generowanie lokalnych optimów przy użyciu lokalnego przeszukiwania w wersji zachłannej

```
function generate_random_local_optima(num_optima, distance_matrix):
    local_optima = []
    for _ in range(num_optima):
        cycle1, cycle2 = create_random_cycles(distance_matrix)
        best_cycle1, best_cycle2, best_score, _ =
greedy_local_search(cycle1, cycle2, distance_matrix,
generate_combined_moves_list_1)
        local_optima.append((best_cycle1, best_cycle2, best_score))
    return local_optima
```

Obliczanie wspólnych wierzchołków pomiędzy rozwiązaniami

```
function calculate_common_edges(cycle1, cycle2, other_cycle1,
other_cycle2):
    function get_edges(cycle):
        return set((min(cycle[i], cycle[i+1]), max(cycle[i],
cycle[i+1])) for i in range(len(cycle)-1))

    edges1 = get_edges(cycle1) | get_edges(cycle2)
    edges2 = get_edges(other_cycle1) | get_edges(other_cycle2)
    common_edges = len(edges1 & edges2)
    return common_edges
```

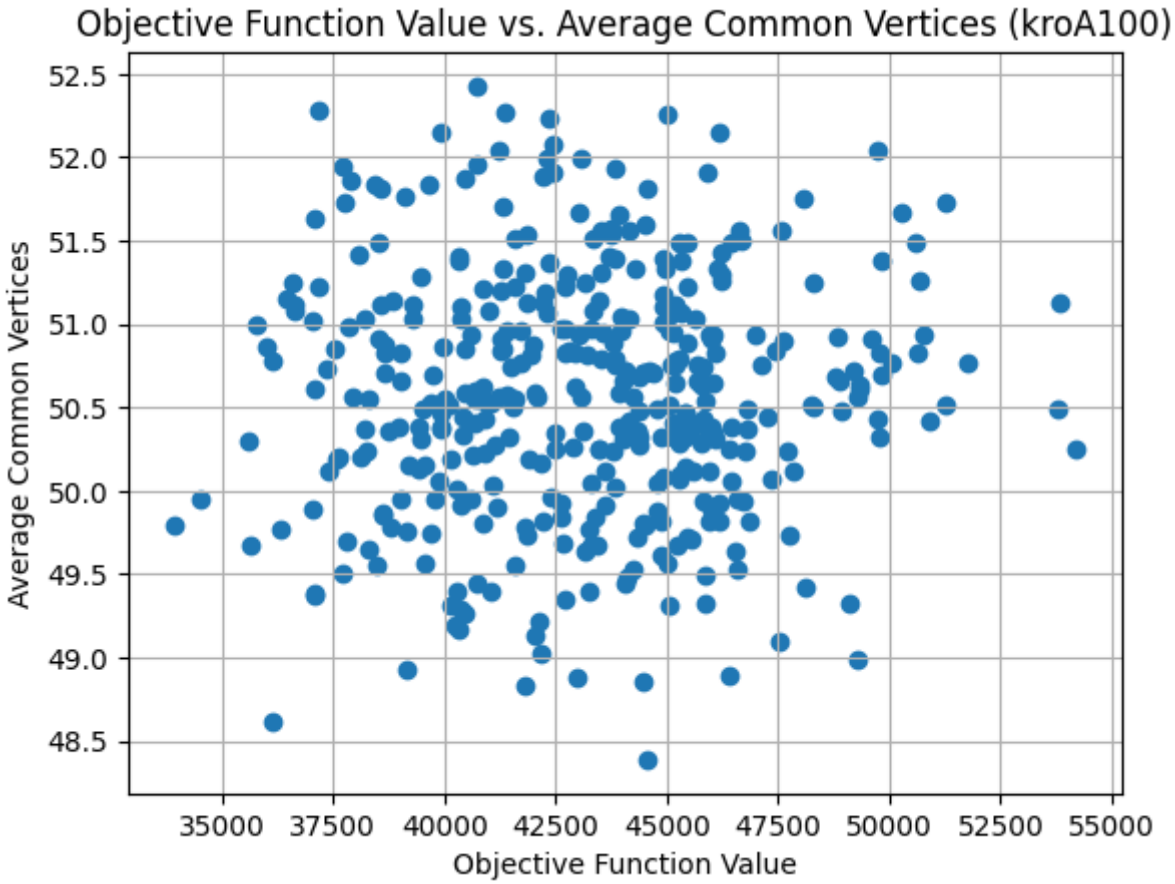
Obliczanie podobieństw pomiędzy lokalnymi optimami a najlepszym rozwiązaniem

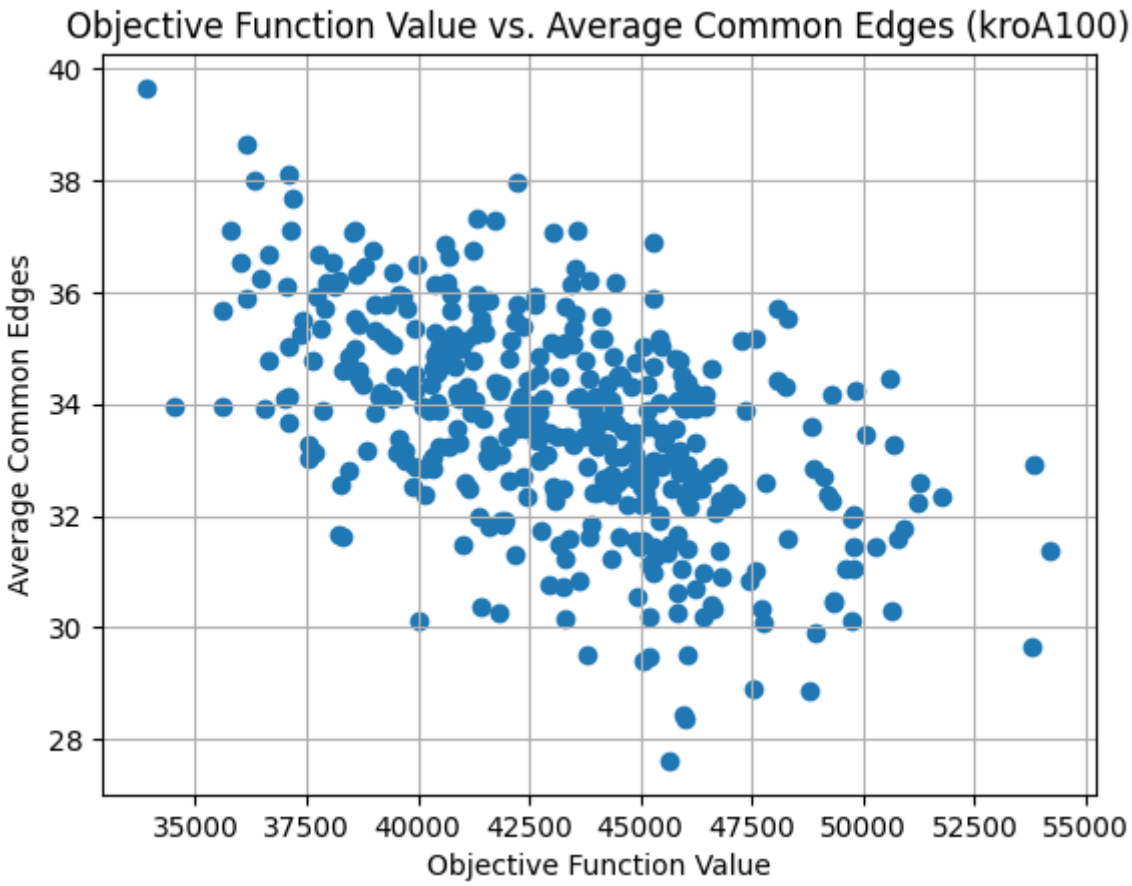
```
function calculate_similarities(local_optima, best_solution):
    best_cycle1, best_cycle2, _ = best_solution
    similarities = []
    for cycle1, cycle2, score in local_optima:
        common_vertices = calculate_common_vertices(cycle1, cycle2,
best_cycle1, best_cycle2)
        common_edges = calculate_common_edges(cycle1, cycle2,
best_cycle1, best_cycle2)

        avg_common_vertices = mean([calculate_common_vertices(cycle1,
cycle2, other_cycle1, other_cycle2)
                                    for other_cycle1, other_cycle2, _
in local_optima])
        avg_common_edges = mean([calculate_common_edges(cycle1,
cycle2, other_cycle1, other_cycle2)
                                 for other_cycle1, other_cycle2, _ in
local_optima])

        similarities.append((score, common_vertices,
avg_common_vertices, common_edges, avg_common_edges))
    return similarities
```

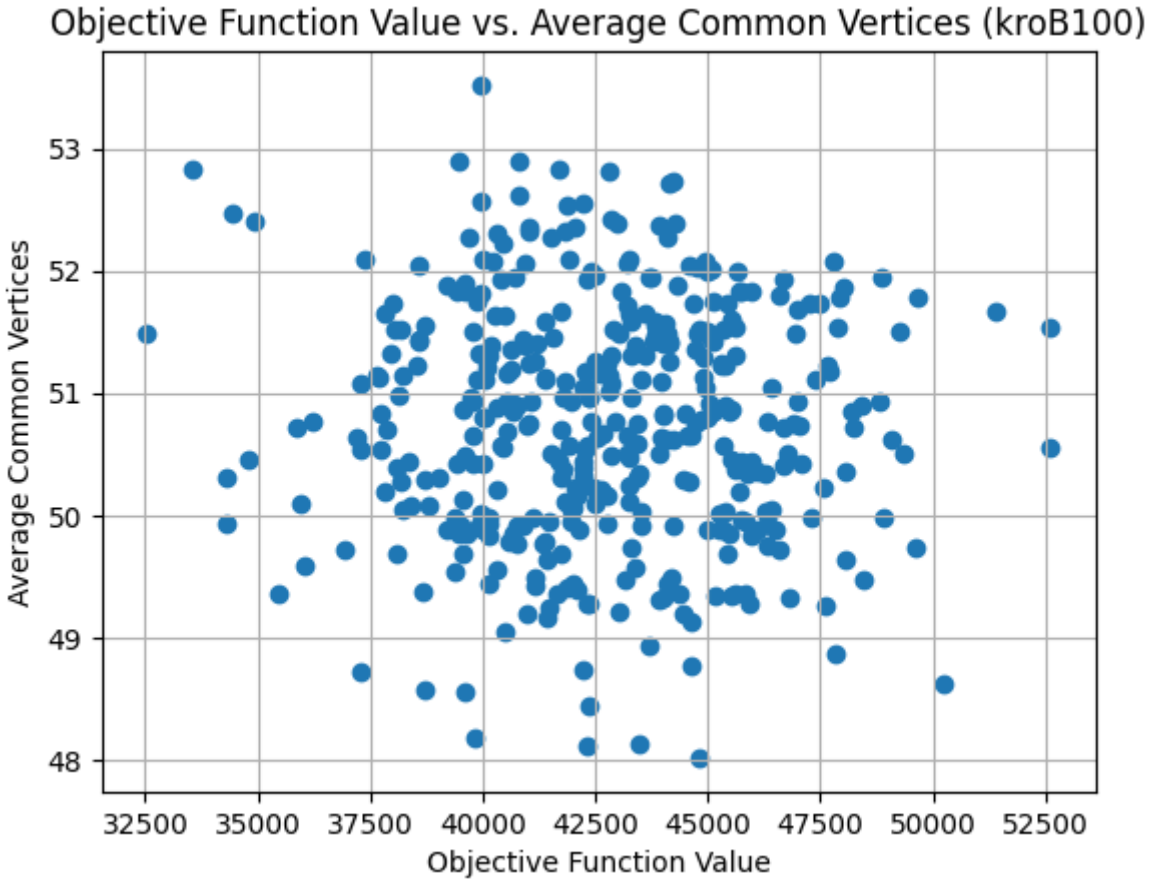
Wizualizacja i obliczanie współczynnika korelacji

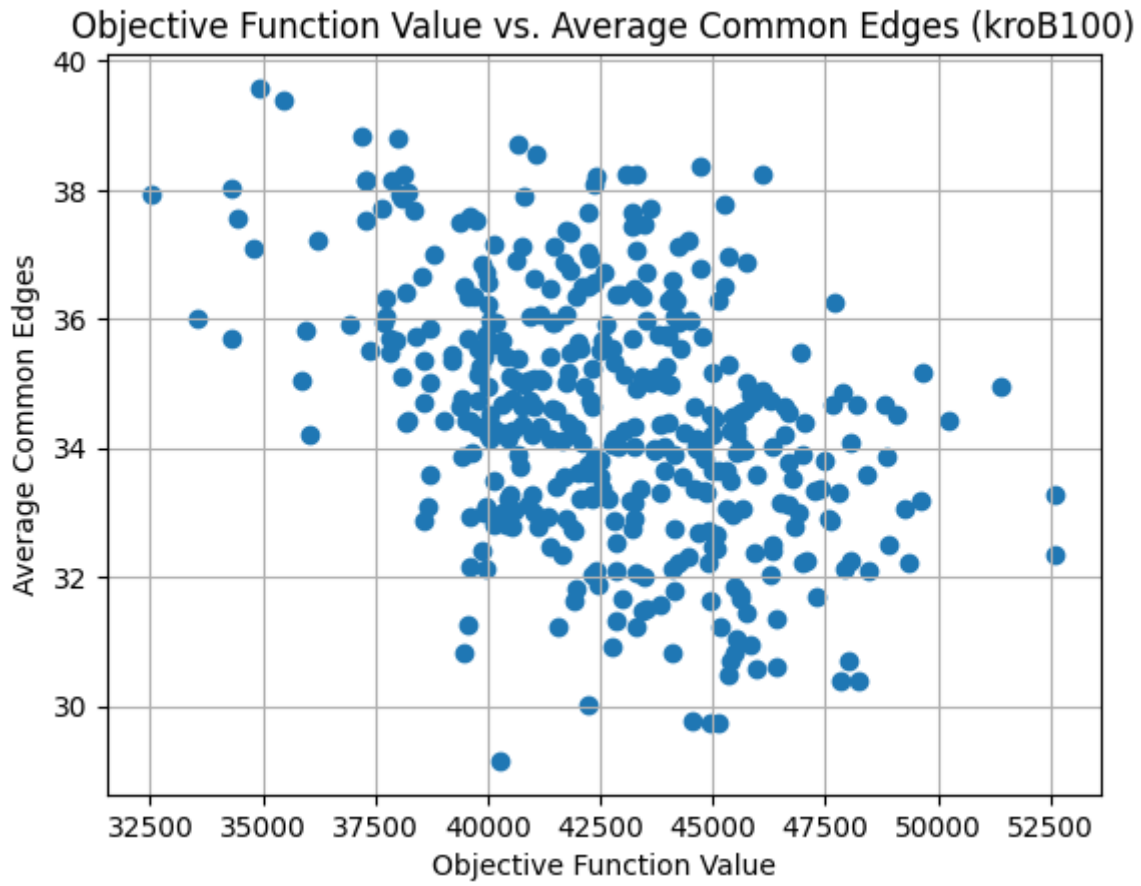
```
function plot_similarity(similarities, measure, instance_name):
    if measure == "vertices":
        x = [s[0] for s in similarities]
        y = [s[2] for s in similarities]
        ylabel = "Average Common Vertices"
    elif measure == "edges":
        x = [s[0] for s in similarities]
        y = [s[4] for s in similarities]
        ylabel = "Average Common Edges"

    scatter_plot(x, y)
    set_labels("Objective Function Value", ylabel)
    set_title(f"Objective Function Value vs. {ylabel}
({instance_name})")
    show_plot()

    correlation = correlation_coefficient(x, y)
    print(f"Correlation coefficient ({measure}, {instance_name}):
{correlation}")
```

Wyniki eksperymentu




Objective Function Value vs. Average Common Edges (kroA100)
Average Common Edges
Objective Function Value
40
38
36
34
32
30
28
35000
37500
40000
42500
45000
47500
50000
52500
55000


Objective Function Value vs. Average Common Vertices (kroB100)
Average Common Vertices
Objective Function Value
53
52
51
50
49
48
32500
35000
37500
40000
42500
45000
47500
50000
52500



Wnioski

1. W obu instancjach nie widać wyraźnej korelacji pomiędzy liczbą wspólnych wierzchołków a wartością funkcji celu
2. W obu instancjach widać umiarkowaną korelację pomiędzy liczbą wspólnych krawędzi a wartością funkcji celu
3. Podobny schemat jest zauważalny dla obu instancji, co świadczy o tym, że nie jest to związane z konkretną instancją problemu

Kod programu

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