In [1]:
```
import matplotlib.pyplot as plt
from synutility.SynIO.data_type import load_from_pickle
import pandas as pd
```

In [2]:
```
import seaborn as sns

def plot_clusters_with_avg_size(clusters: dict, invariant_name, algorithm
    import matplotlib.pyplot as plt
    import pandas as pd

    # Extract group and cluster names
    group_names = set()
    cluster_names = set()

    for cluster_name in clusters.keys():
        group_name = cluster_name.split('_cluster_')[0]
        group_names.add(group_name)
        cluster_names.add(cluster_name)

    # Calculate the sizes of each cluster
    cluster_sizes = [len(cluster) for cluster in clusters.values()]

    # Find the minimum, maximum, and average size of the clusters
    min_size = min(cluster_sizes)
    max_size = max(cluster_sizes)

    # Calculate average cluster size per group
    group_cluster_sizes = {group: [] for group in group_names}
    for cluster_name, cluster in clusters.items():
        group_name = cluster_name.split('_cluster_')[0]
        group_cluster_sizes[group_name].append(len(cluster))

    average_cluster_size_per_group = {group: sum(sizes) / len(sizes) for

    # Create a dataframe for plotting
    group_cluster_counts = {group: 0 for group in group_names}
    for cluster_name in clusters.keys():
        group_name = cluster_name.split('_cluster_')[0]
        group_cluster_counts[group_name] += 1

    df_plot = pd.DataFrame({
        'Group': list(group_cluster_counts.keys()),
        'Cluster Count': list(group_cluster_counts.values()),
        'Min Size': [min_size] * len(group_cluster_counts),
        'Max Size': [max_size] * len(group_cluster_counts),
        'Average Size': [average_cluster_size_per_group[group] for group
    })

    # Order the dataframe by Group
    df_plot = df_plot.sort_values('Cluster Count', ascending=False)

    # Plotting
    fig, ax1 = plt.subplots(figsize=(10, 6))

    # Create a second y-axis for the average cluster size
    ax2 = ax1.twinx()
    sns.lineplot(x='Group', y='Average Size', data=df_plot, ax=ax2, color
    ax2.set_ylabel('Average Cluster Size', color='r')
```

```
    ax2.tick_params(axis='y', labelcolor='r')

    # Annotate each point on the line with its value
    for line in ax2.lines:
        for x, y in zip(line.get_xdata(), line.get_ydata()):
            ax2.annotate(f'{y:.2f}', xy=(x, y), xytext=(5, 5), textcoords

    # Bar plot for cluster counts with pastel colors
    sns.barplot(x='Group', y='Cluster Count', data=df_plot, ax=ax1, palet
    ax1.set_ylabel(f'# Clusters - Algorithm: {algorithm_name}', color='bl
    ax1.set_xlabel(f'Groups - Invariant: {invariant_name}', color='black'
    ax1.tick_params(axis='y', labelcolor='black')

    group_numbers = [str(i) for i in range(len(df_plot))]
    ax1.set_xticks(range(len(group_numbers)))  # Explicitly set tick posi
    ax1.set_xticklabels(group_numbers)        # Set tick labels

    # Move the legend to the right of the plot
    ax1.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

    # Overlay box for 3 number summary on the right
    row = df_plot.iloc[0]
    # Overlay box for 3-number summary
    # Adjust the figure to create space for the overlays on the right
    fig.subplots_adjust(right=0.9)  # Leave 25% of the figure for overlay

    # Define overlay content
    summary_text = (
        f"Min cluster size: {row['Min Size']}\n"
        f"Max cluster size: {row['Max Size']}\n"
        f"Average cluster size: {row['Average Size']:.2f}"
    )
    time_text = f"Time taken: {time_taken:.2f}s"

    # Add the summary text without a box
    fig.text(
        0.97, 0.6,  # x, y position in figure coordinates
        summary_text,
        ha='left', va='top', fontsize=10
    )

    # Add the time annotation without a box
    fig.text(
        0.97, 0.5,  # x, y position in figure coordinates
        time_text,
        ha='left', va='top', fontsize=10
    )

    plt.title(f'Invariant:{invariant_name} - Algorithm:{algorithm_name}')
    plt.show()
```

In [3]:

```
import pandas as pd
import seaborn as sns

def plot_clusters(clusters: dict, invariant_name, algorithm_name, time_ta
    import matplotlib.pyplot as plt

    # Extract cluster names
    cluster_names = list(clusters.keys())
```

```
    # Calculate the sizes of each cluster
    cluster_sizes = [len(cluster) for cluster in clusters.values()]

    # Find the minimum, maximum, and average size of the clusters
    min_size = min(cluster_sizes)
    max_size = max(cluster_sizes)
    avg_size = sum(cluster_sizes) / len(cluster_sizes)

    # Create a dataframe for plotting
    df_plot = pd.DataFrame({
        'Cluster': cluster_names,
        'Size': cluster_sizes
    })

    # Order the dataframe by Cluster Size
    df_plot = df_plot.sort_values('Size', ascending=False)

    # Plotting
    fig, ax1 = plt.subplots(figsize=(10, 6))

    # Bar plot for cluster sizes with pastel colors
    sns.barplot(x='Cluster', y='Size', data=df_plot, ax=ax1, palette='fla
    ax1.set_ylabel(f'Cluster Size - Algorithm: {algorithm_name}', color='
    ax1.set_xlabel(f'Clusters - Invariant: {invariant_name}', color='blac
    ax1.tick_params(axis='y', labelcolor='black')

    # Add height labels to each bar with smaller text size
    for p in ax1.patches:
        ax1.annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width()
                     ha='center', va='center', xytext=(0, 10), textcoords

    # Reduce the number of ticks on the x-axis
    num_ticks = 10  # Set the number of ticks you want
    tick_positions = range(0, len(df_plot), max(1, len(df_plot) // num_ti
    tick_labels = [str(i) for i in tick_positions]
    ax1.set_xticks(tick_positions)  # Explicitly set tick positions
    ax1.set_xticklabels(tick_labels)  # Set tick labels

    # Move the legend to the right of the plot
    ax1.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

    # Overlay box for 3 number summary on the right
    row = df_plot.iloc[0]
    # Overlay box for 3-number summary
    # Adjust the figure to create space for the overlays on the right
    fig.subplots_adjust(right=0.9)  # Leave 25% of the figure for overlay

    # Define overlay content
    summary_text = (
        f"Min cluster size: {min_size}\n"
        f"Max cluster size: {max_size}\n"
        f"Average cluster size: {avg_size:.2f}"
    )
    time_text = f"Time taken: {time_taken:.2f}s"

    # Add the summary text without a box
    fig.text(
        0.97, 0.6,  # x, y position in figure coordinates
        summary_text,
        ha='left', va='top', fontsize=10
```

```
    )

    # Add the time annotation without a box
    fig.text(
        0.97, 0.5,  # x, y position in figure coordinates
        time_text,
        ha='left', va='top', fontsize=10
    )

    plt.title(f'Invariant:{invariant_name} - Algorithm:{algorithm_name}')
    plt.show()
```

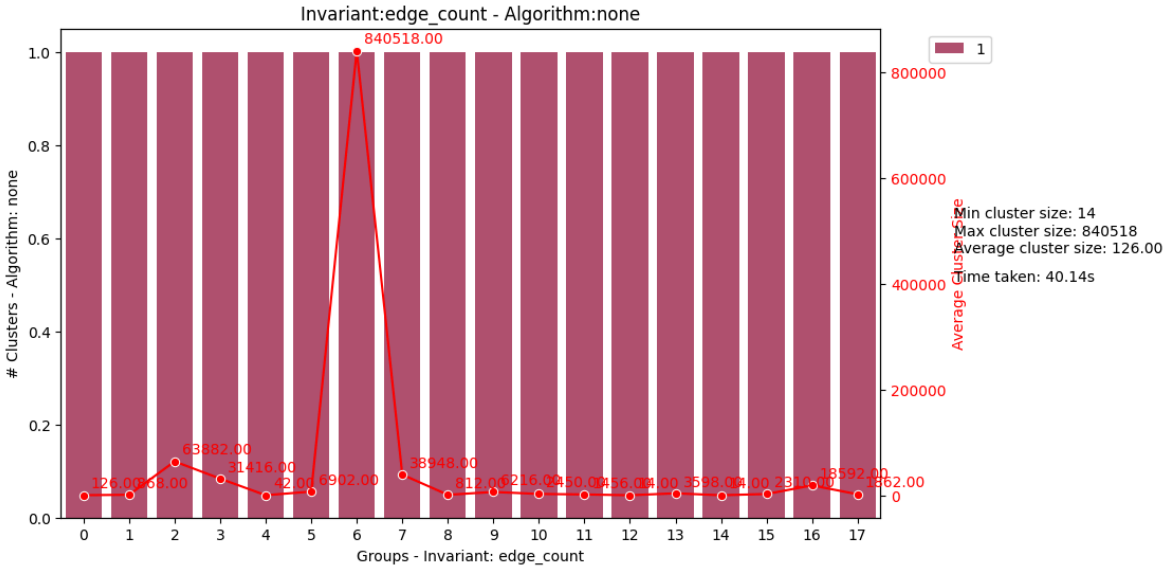
In [4]:
```
import os

def load_all_results():
    results_folder = 'data/results/'
    all_results = {}

    for file_name in os.listdir(results_folder):
        if file_name.endswith('.pkl.gz'):
            file_path = os.path.join(results_folder, file_name)
            result = load_from_pickle(file_path)
            all_results[file_name] = result

    return all_results
```

In [5]:
```
all_results = load_all_results()
```

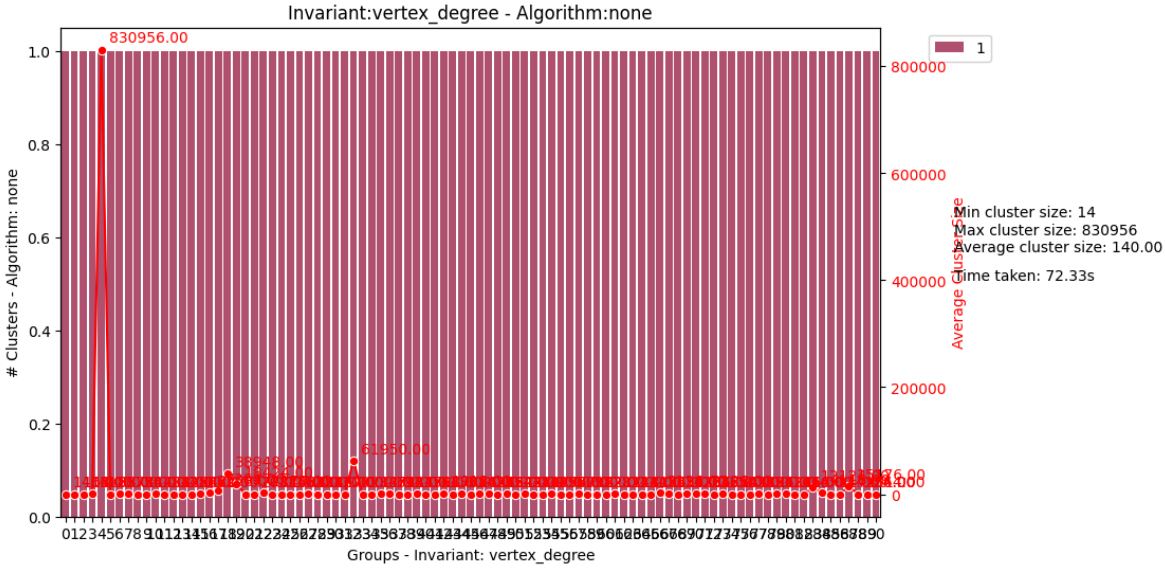
In [11]:
```
def plot_all_results(result_dict):
    for result_name, result in result_dict.items():
        clusters = result['clusters']
        invariant_name = result['configuration'].invariant
        algorithm_name = result['configuration'].algorithm
        num_clusters = result['cluster_count']
        time_taken = result['time']

        print(invariant_name, algorithm_name, time_taken, num_clusters)
        if invariant_name != "none":
            plot_clusters_with_avg_size(clusters, invariant_name, algorit
        else:
            plot_clusters(clusters, invariant_name, algorithm_name, time_
```

In [12]:
```
plot_all_results(all_results)
```

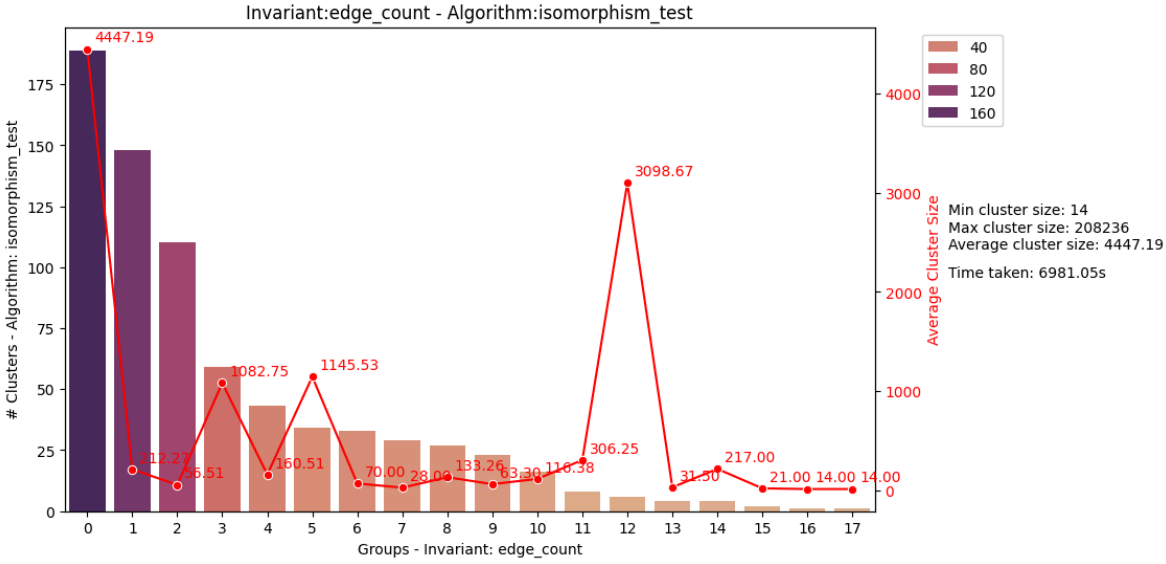
```
edge_count none 40.140152031 18
```



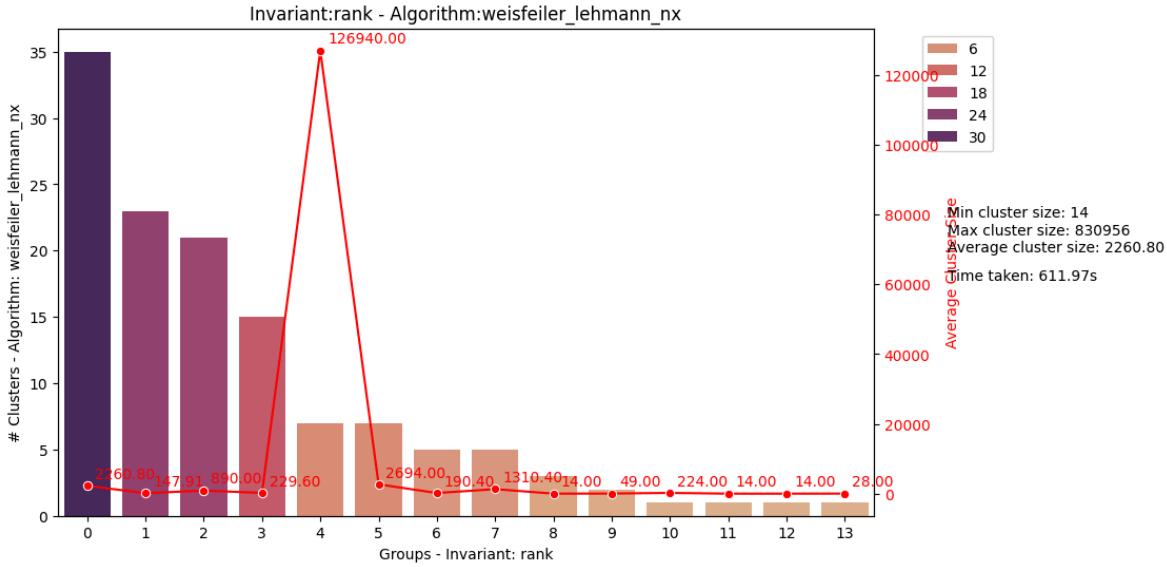
vertex_degree none 72.331255746 91



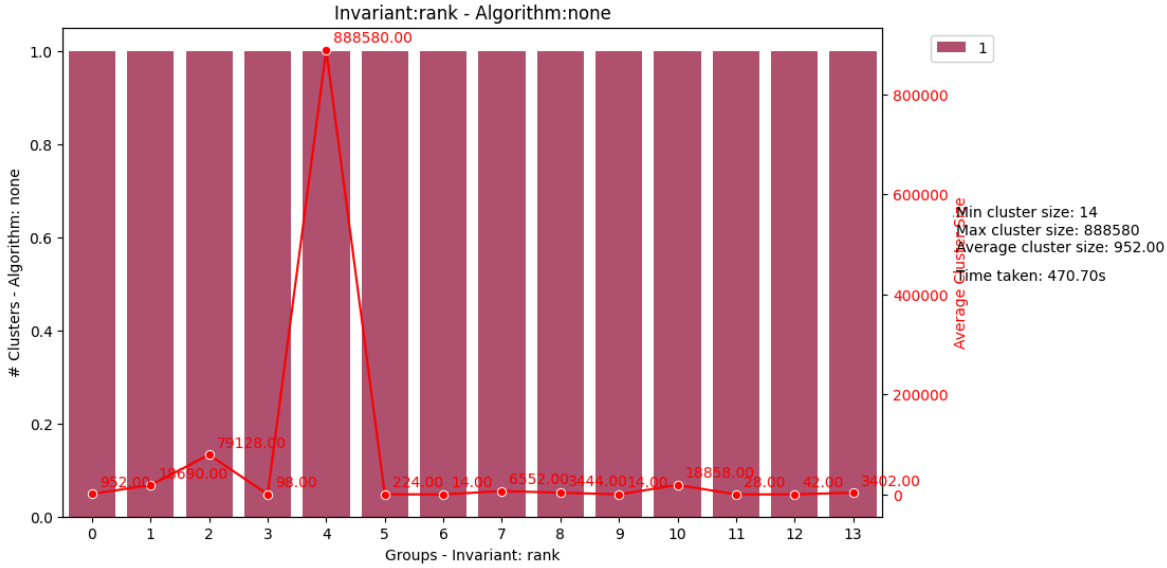
edge_count isomorphism_test 6981.045720730002 737



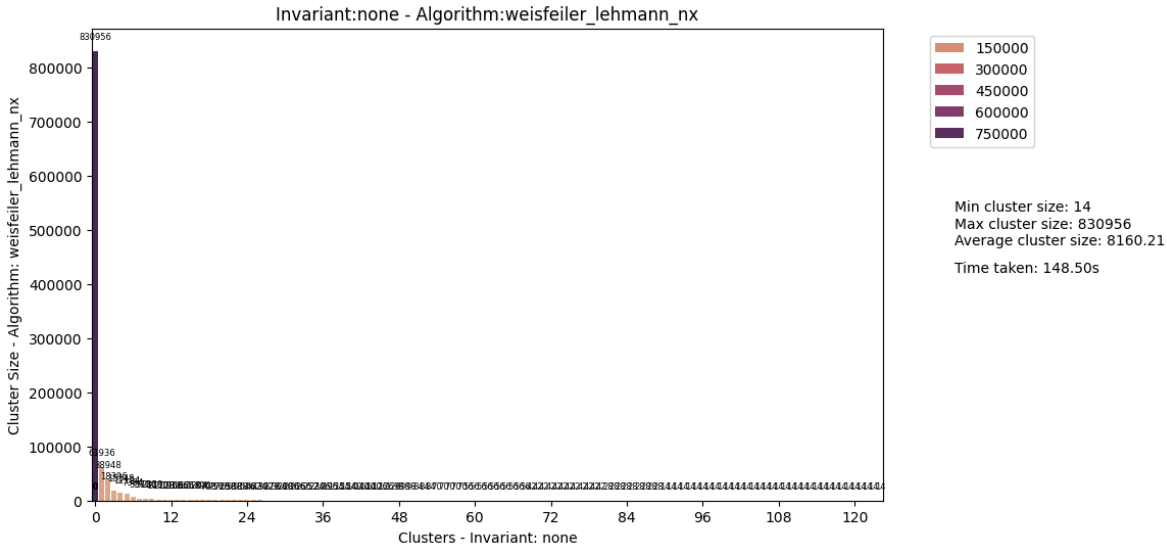
rank weisfeiler_lehmann_nx 611.9701464630001 127



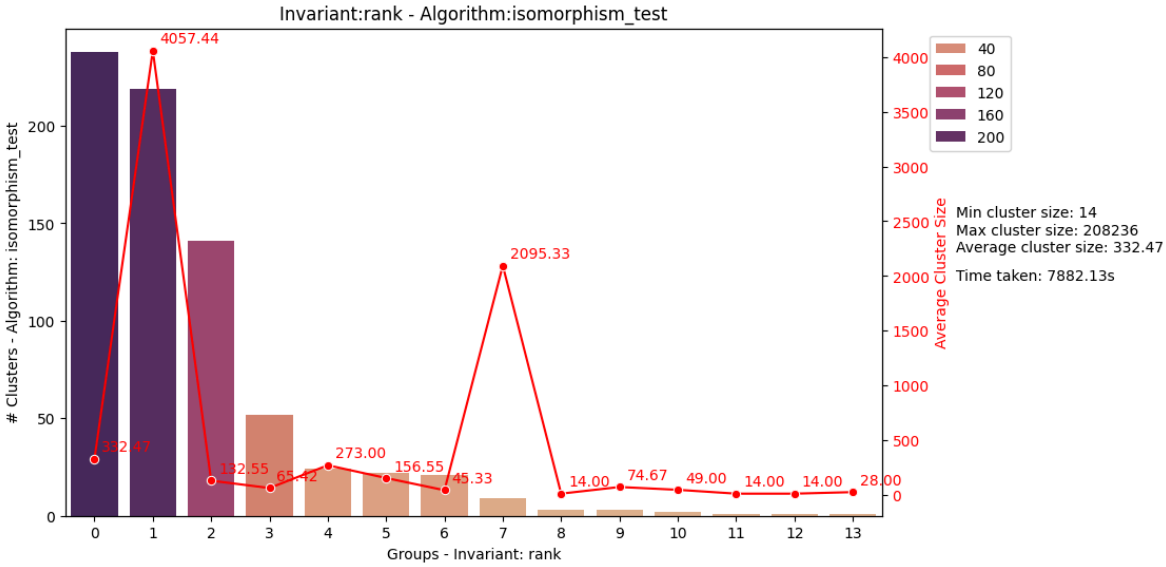

```
rank none 470.6967342010001 14
```



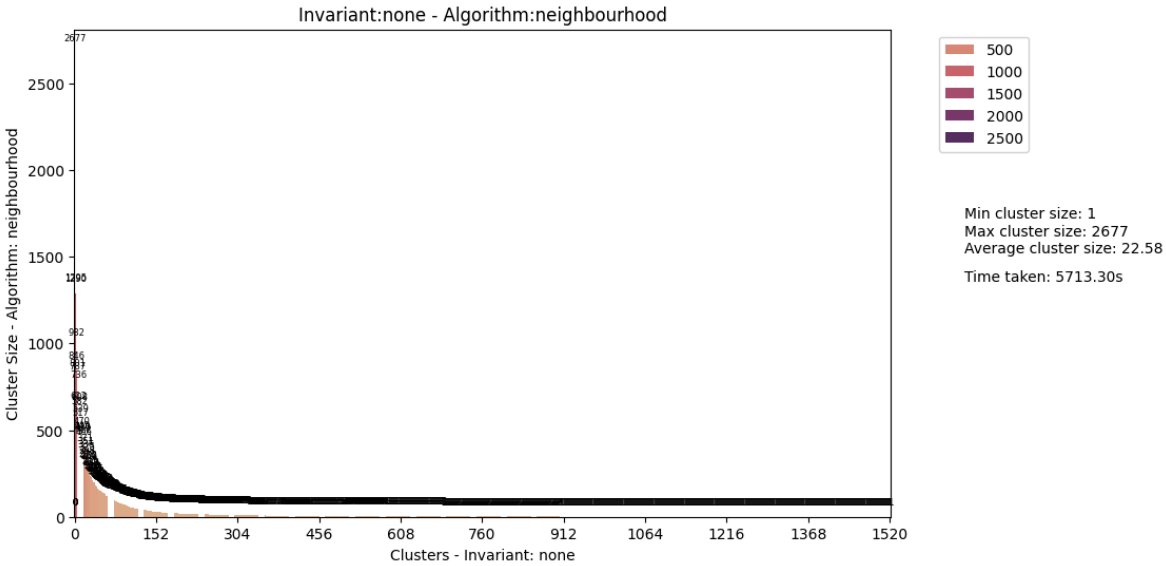

```
none weisfeiler_lehmann_nx 148.50112653300008 125
```



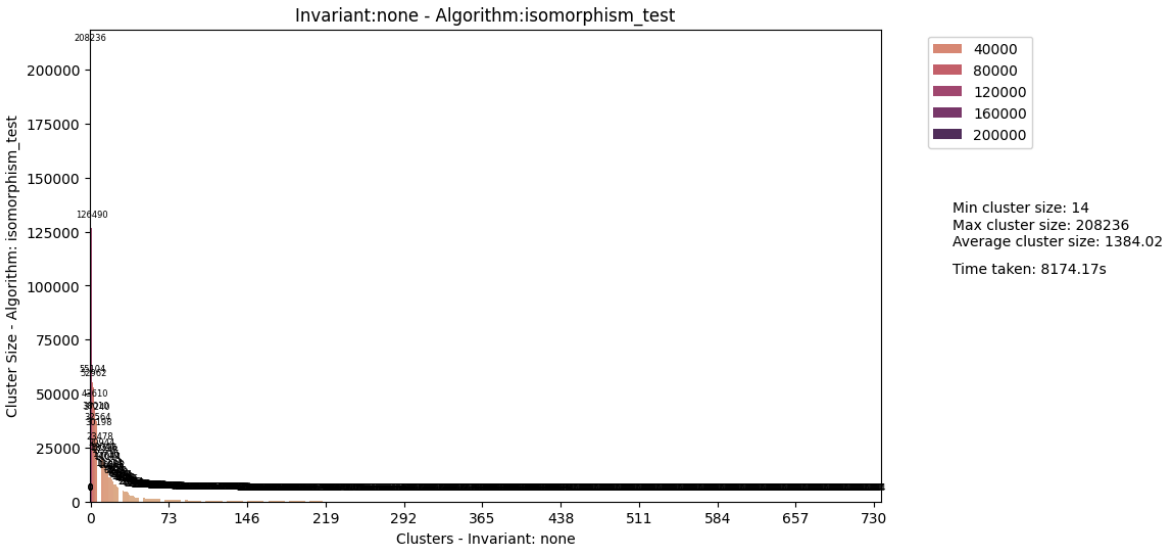

```
rank isomorphism_test 7882.125555398 737
```



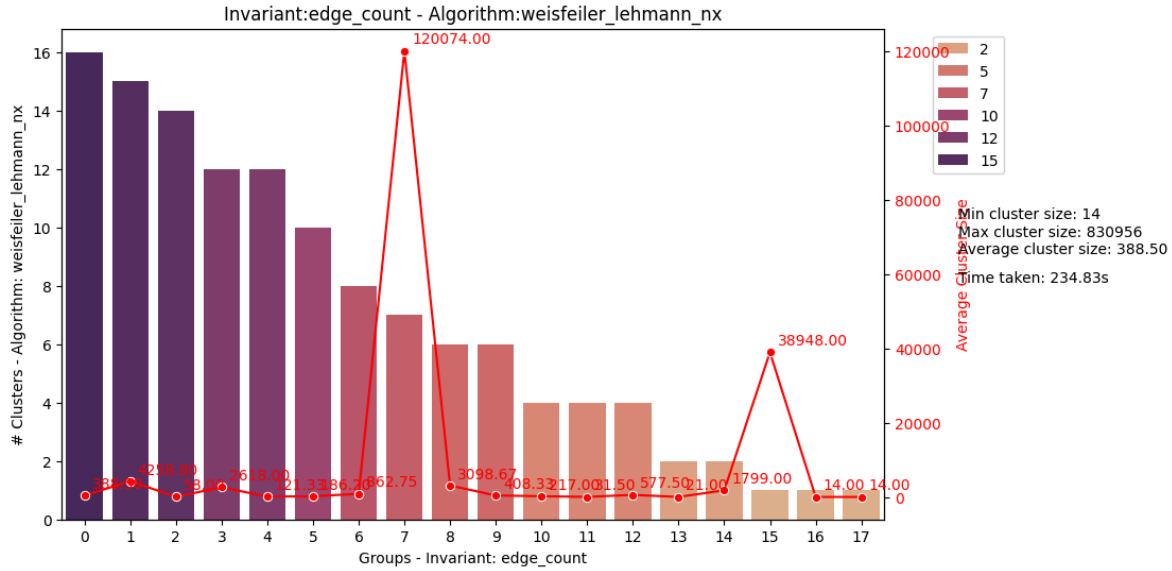
none neighbourhood 5713.295324134998 1523



none isomorphism_test 8174.170583116999 737



edge_count weisfeiler_lehmann_nx 234.83369949500002 125


Invariant:edge_count - Algorithm:weisfeiler_lehmann_nx
Clusters - Algorithm: weisfeiler_lehmann_nx
Average Cluster Size
Groups - Invariant: edge_count
120074.00
38948.00
3098.67
1799.00
14.00
14.00
2
5
7
10
12
15
Min cluster size: 14
Max cluster size: 830956
Average cluster size: 388.50
Time taken: 234.83s

In [ ]: