

# evaluation

October 7, 2024

## 1 Evaluation

### 1.1 Machine specs

The evaluation was performed on a machine with the following specs: - 12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz - 16 GB of RAM - Windows 11

The evaluation of MAFIA was performed using GPUMAFIA, which was installed on a virtual machine using VirtualBox. The virtual machine was configured with 4 CPUs and 4 GB of RAM. The operating system was Ubuntu 24.04.1 LTS.

CLIQUE and SUBCLU were evaluated on the host machine using ELKI.

### 1.2 Scalability with data size

- Tested on a 20-dimensional data set containing 5 clusters in 5 different subspaces.
- 10% of the data was added as noise records.
- CLIQUE stopped working after 7mio records and SUBCLU stopped working after 200,000 records. MAFIA was able to handle at least 15mio records.

#### 1.2.1 Settings

CLIQUE: - xsi: 25 - tau: 0.08

SUBCLU: - epsilon 0.02 - minpts: 250 (for 10k points). Doubled for each step, so 500 minpts for 20k, 1000 minpts for 40k, etc.

MAFIA: -a 1.4 -b 0.35 -M 20 -n 1000 -V -timing

#### 1.2.2 Plot

```
[9]: import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Data
size = [10000, 20000, 30000, 40000, 50000, 100000, 200000, 1000000, 2000000,
        ↪3000000, 4000000, 5000000, 6000000, 7000000, 8000000, 9000000, 10000000,
        ↪15000000]
clique = [62, 119, 155, 203, 235, 533, 1255, 8795, 16290,
          ↪38581, 50392, 62348, 127730, 142005]
```

```

mafia = [15 , 25 , 17 , 22 , 27 , 31 , 45 , 155 , 265 ,
↪414 , 556 , 637 , 922 , 1016 , 1225 , 1215 , 1641]
subclu = [137 , 477 , 984 , 1751 , 3055 , 12490 , 67603]

# Figure with 3 subplots
fig, axs = plt.subplots(1, 3, figsize=(7.5, 2))

# CLIQUE (size in millions)
axs[0].plot([s / 1_000_000 for s in size[:len(clique)]], clique, 'ro-',
↪label='CLIQUE')
axs[0].set_xlabel('size (mio. of records)')
axs[0].set_ylabel('runtime (ms)')
axs[0].xaxis.set_major_locator(ticker.MaxNLocator(5))
axs[0].yaxis.set_major_locator(ticker.MaxNLocator(5))
axs[0].legend()

# MAFIA (size in millions)
axs[1].plot([s / 1_000_000 for s in size[:len(mafia)]], mafia, 'b^-',
↪label='MAFIA')
axs[1].set_xlabel('size (mio. of records)')
axs[1].xaxis.set_major_locator(ticker.MaxNLocator(4))
axs[1].yaxis.set_major_locator(ticker.MaxNLocator(4))
axs[1].legend()

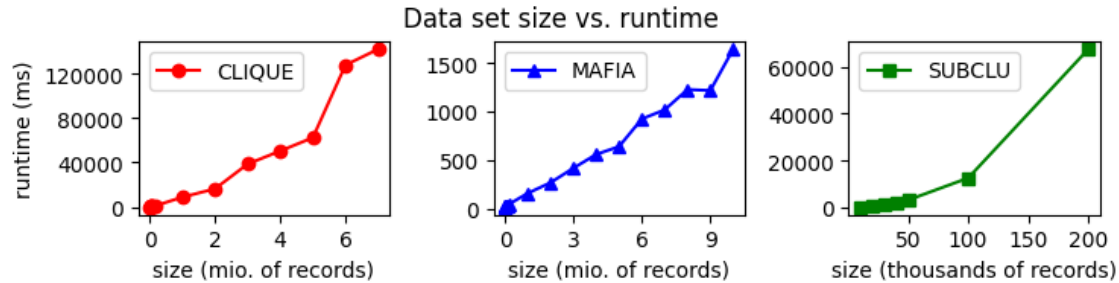
# SUBCLU (size in thousands)
axs[2].plot([s / 1000 for s in size[:len(subclu)]], subclu, 'gs-',
↪label='SUBCLU')
axs[2].set_xlabel('size (thousands of records)')
axs[2].xaxis.set_major_locator(ticker.MaxNLocator(5))
axs[2].yaxis.set_major_locator(ticker.MaxNLocator(4))

axs[2].legend()

# Show plots
plt.suptitle('Data set size vs. runtime')
plt.tight_layout(rect=[0, 0, 1, 1.1])
plt.show()

# Save plot
plt.savefig('figures/dataset_size_vs_runtime.png')

```



<Figure size 640x480 with 0 Axes>

Both CLIQUE and MAFIA scales linearly with the number of records. SUBCLU seems to grow more exponentially, which also explains why it stopped working after 200,000 records.

### 1.3 Accuracy

- Tested on a 10-dimensional data set containing 100,000 points for testing CLIQUE and MAFIA, however, SUBCLU was not able to handle this amount of data, so it was tested on a similar distribution of points with 20,000 points.
- 10% of the data was added as noise records.
- First case, has two clusters embedded in a different 4 dimensional subspace. Second case, has 4 clusters embedded in a different 4 dimensional subspace.

#### 1.3.1 2 clusters

**Settings** SUBCLU: - epsilon: 0.05 - minpts: 850

CLIQUE: - xsi: 20 - tau: 0.41 - prune: false

MAFIA: -a 3 -b 0.6 -M 20 -n 1000 -V -p -timing

#### Plot

```
[10]: import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load images
clique_img = mpimg.imread('datasets/mdcgen/accuracy/2clusters/100k/clique/
↪3d_plot.png')
mafia_img = mpimg.imread('datasets/mdcgen/accuracy/2clusters/100k/mafia/3d_plot.
↪png')
subclu_img = mpimg.imread('datasets/mdcgen/accuracy/2clusters/20k/subclu/
↪3d_plot.png')

# Figure with subplots
fig, axs = plt.subplots(1, 3, figsize=(11, 4))

# Plot images
```

```

axs[0].imshow(clique_img)
axs[0].axis('off')
axs[0].set_title('CLIQUE')

axs[1].imshow(mafia_img)
axs[1].axis('off')
axs[1].set_title('MAFIA')

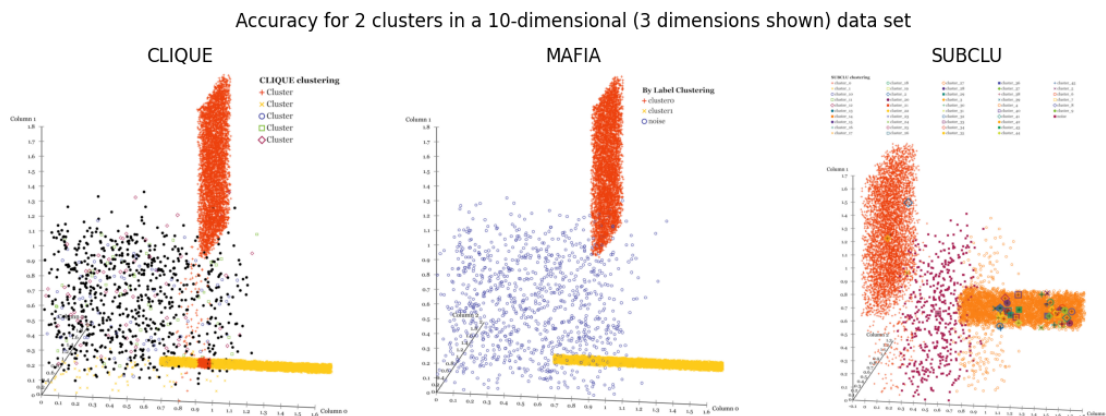
axs[2].imshow(subclu_img)
axs[2].axis('off')
axs[2].set_title('SUBCLU')

# Title
plt.suptitle('Accuracy for 2 clusters in a 10-dimensional (3 dimensions shown) ↵
↳data set')

# Show plot
plt.tight_layout(rect=[0, 0, 1, 1.02])
plt.show()

# Save plot
plt.savefig('figures/accuracy_2clusters.png')

```



<Figure size 640x480 with 0 Axes>

MAFIA reports the correct clusters in both cases and detects the noise points.

CLIQUE also reports the two clusters, however, it also finds some noise points as clusters.

SUBCLU reports the correct clusters, but it also finds a lot extra incorrectly clusters.

### 1.3.2 4 clusters

**Settings** SUBCLU: - epsilon: 0.02 - minpts: 500

CLIQUE: - xsi: 35 - tau: 0.2

MAFIA: -a 3 -b 0.6 -M 100 -n 1000 -V -p -timing

### Plot

```
[11]: import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load images
clique_img = mpimg.imread('datasets/mdcgen/accuracy/4clusters/100k/clique/
↳3d_plot.png')
mafia_img = mpimg.imread('datasets/mdcgen/accuracy/4clusters/100k/mafia/3d_plot.
↳png')
subclu_img = mpimg.imread('datasets/mdcgen/accuracy/4clusters/20k/subclu/
↳3d_plot.png')

# Figure with subplots
fig, axs = plt.subplots(1, 3, figsize=(7, 4))

# Plot images
axs[0].imshow(clique_img)
axs[0].axis('off')
axs[0].set_title('CLIQUE')

axs[1].imshow(mafia_img)
axs[1].axis('off')
axs[1].set_title('MAFIA')

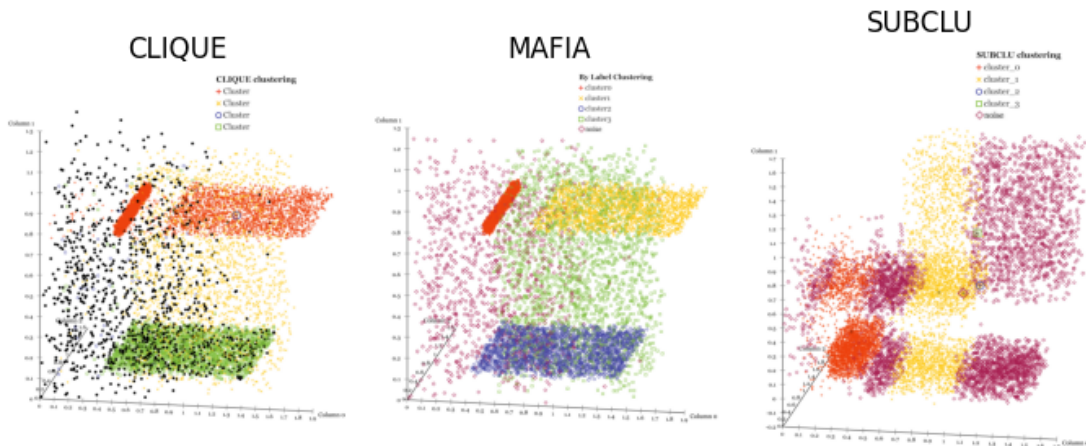
axs[2].imshow(subclu_img)
axs[2].axis('off')
axs[2].set_title('SUBCLU')

# Title
plt.suptitle('Accuracy for 4 clusters in a 10-dimensional (3 dimensions shown)
↳data set')

# Show plot
plt.tight_layout(rect=[0, 0, 1, 1.15])
plt.show()

# Save plot
plt.savefig('figures/accuracy_4clusters.png')
```

Accuracy for 4 clusters in a 10-dimensional (3 dimensions shown) data set



<Figure size 640x480 with 0 Axes>

Again, MAFIA reports the correct clusters in both cases and detects the noise points.

CLIQUE fails to detect one of the clusters.

SUBCLU fails completely to detect the clusters.

### 1.3.3 Plus shape

Settings CLIQUE: - xsi: 10 - tau: 0.2

#### Plot

```
[12]: import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load images
clique_img = mpimg.imread('datasets/home_made/accuracy/plus_shape/clique/plot.
    png')
mafia_img = mpimg.imread('datasets/home_made/accuracy/plus_shape/mafia/plot.
    png')

# Figure with subplots
fig, axs = plt.subplots(1, 2, figsize=(7, 4))

# Plot images
axs[0].imshow(clique_img)
axs[0].axis('off')
axs[0].set_title('CLIQUE')

axs[1].imshow(mafia_img)
```

```

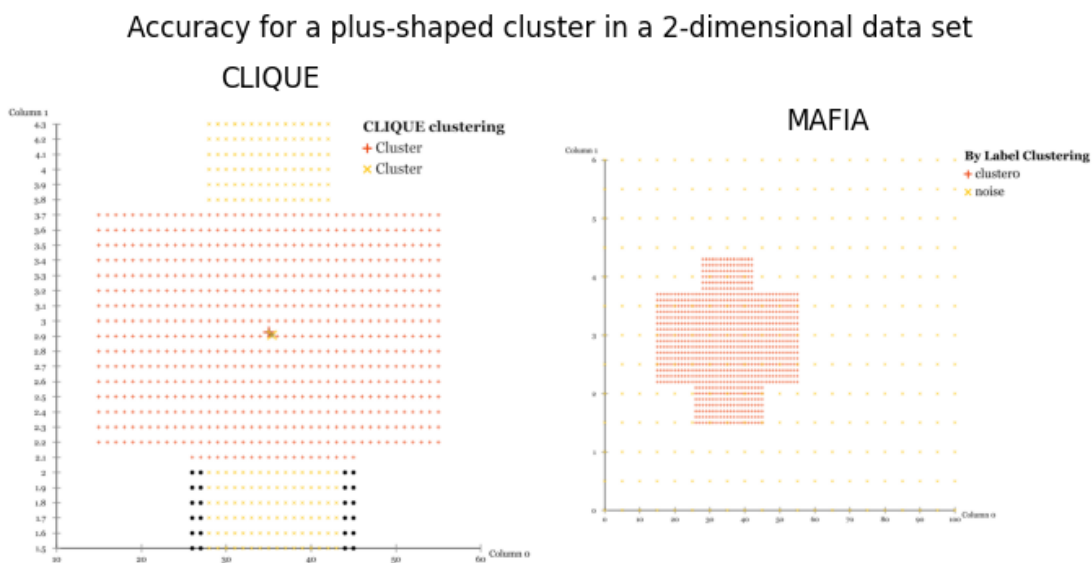
axs[1].axis('off')
axs[1].set_title('MAFIA')

# Title
plt.suptitle('Accuracy for a plus-shaped cluster in a 2-dimensional data set')

# Show plot
plt.tight_layout(rect=[0, 0, 1, 1.05])
plt.show()

# Save plot
plt.savefig('figures/accuracy_plus.png')

```



<Figure size 640x480 with 0 Axes>

MAFIA reports the correct cluster borders. CLIQUE fails to detect the borders correctly.

### 1.3.4 Bezier curve

Settings CLIQUE: - xsi: 20 - tau: 0.025

SUBCLU: - epsilon: 0.2 - minpts: 76 - mindim: 2

DBSCAN: - epsilon: 0.04 - minpts: 100

```

[13]: import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load images

```

```

clique_img = mpimg.imread('datasets/artificialCluster/accuracy/bezier/clique/
    ↳plot.png')
subclu_img = mpimg.imread('datasets/artificialCluster/accuracy/bezier/subclu/
    ↳plot.png')
dbscan_img = mpimg.imread('datasets/artificialCluster/accuracy/bezier/dbscan/
    ↳plot.png')

# Figure with subplots
fig, axs = plt.subplots(1, 3, figsize=(11, 4))

# Plot images
axs[0].imshow(clique_img)
axs[0].axis('off')
axs[0].set_title('CLIQUE')

axs[1].imshow(subclu_img)
axs[1].axis('off')
axs[1].set_title('SUBCLU')

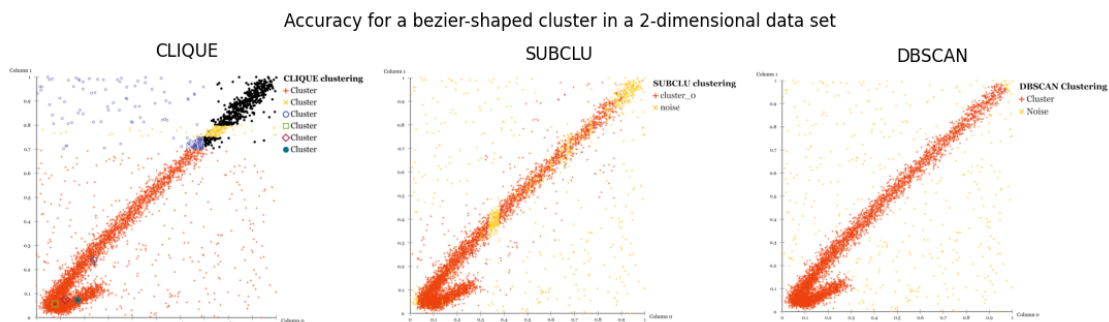
axs[2].imshow(dbscan_img)
axs[2].axis('off')
axs[2].set_title('DBSCAN')

# Title
plt.suptitle('Accuracy for a bezier-shaped cluster in a 2-dimensional data set')

# Show plot
plt.tight_layout(rect=[0, 0, 1, 1.15])
plt.show()

# Save plot
plt.savefig('figures/accuracyBezier.png')

```



<Figure size 640x480 with 0 Axes>



Even though one might expect SUBCLU to produce the same results as DBSCAN when run on a 2D dataset with  $\text{mindim}=2$ , the results will differ. This is because SUBCLU always uses a bottom-up approach to explore subspaces. It first examines 1D subspaces before moving on to higher-dimensional ones. When  $\text{mindim}$  is set to 2, SUBCLU disregards clusters that exist in subspaces with dimensionality less than 2. As a result, even though both algorithms are set to work in the full 2D space, the subspace exploration process and the focus on dimensionality lead SUBCLU to produce different clustering results than DBSCAN.

## 1.4 Cluster dimensionality

### 1.4.1 Settings

CLIQUE: - xsi: 30 - tau: 0.25

MAFIA: -a 2.2 -b 0.35 -M 40 -n 1000 -p -V -timing

### 1.4.2 Plot

```
[14]: import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Data
dims = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20]
mafia = [45, 63, 55, 45, 63, 55, 60, 79, 86, 1387, 62607]
clique = [4440, 4347, 6017, 5740, 8093, 12532, 26373]

# Figure size
plt.figure(figsize=(2.5, 2.1))

# Plot both in same figure
plt.plot(dims[:len(clique)], clique, 'ro-', label='CLIQUE')
plt.plot(dims, mafia, 'b^-', label='MAFIA')

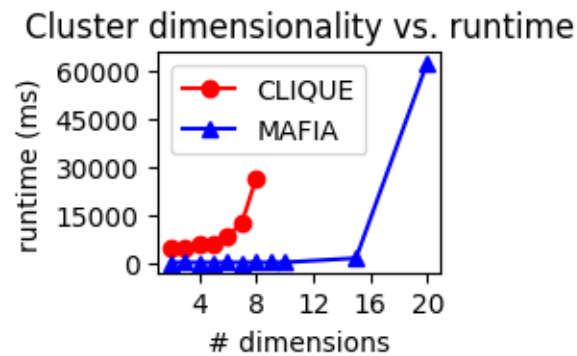
# Set number of x-ticks
plt.gca().xaxis.set_major_locator(ticker.MaxNLocator(6))
plt.gca().yaxis.set_major_locator(ticker.MaxNLocator(6))

# Add labels and title
plt.xlabel('# dimensions')
plt.ylabel('runtime (ms)')
plt.title('Cluster dimensionality vs. runtime')

# Add legend
plt.legend()

# Show plot
plt.tight_layout()
plt.show()
```

```
# Save plot
plt.savefig('figures/cluster_dimensionality_vs_runtime.png')
```



<Figure size 640x480 with 0 Axes>

MAFIA uses almost no time to find the clusters compared to CLIQUE of the cluster dimensions is not higher than 16. After that, the time seems to increase exponentially.

## 1.5 Data dimensionality

MAFIA and CLIQUE was on a 100 k dataset.

### 1.5.1 Settings

CLIQUE: - xsi: 30 - tau: 0.3 - prune: false

MAFIA: -a 2.2 -b 0.35 -M 40 -n 1000 -p -V -timing

### 1.5.2 Plot

```
[15]: import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Data
dims = [10, 20, 30, 40, 50, 100]
mafia_runtime = [49, 72, 85, 104, 133, 317]
clique_runtime = [35688, 44180, 50275, 47682, 67772]

# CLIQUE:
# 10d = 66 clusters found
# 20d = 84
# 30d = 93
# 40d = 77
# 50d = 93
# 100d, not able to run.
```

```

# MAFIA:
# 10d = 3 clusters found
# 20d = 4 clusters found
# 30d = 3 clusters found
# 40d = 4 clusters found
# 50d = 3 clusters found
# 100d = 4 clusters found

# Figure with 2 subplots
fig, axs = plt.subplots(1, 2, figsize=(4, 2))

# CLIQUE
axs[0].plot(dims[:len(clique_runtime)], clique_runtime, 'ro-', label='CLIQUE')
axs[0].legend()
axs[0].xaxis.set_major_locator(ticker.MaxNLocator(5))

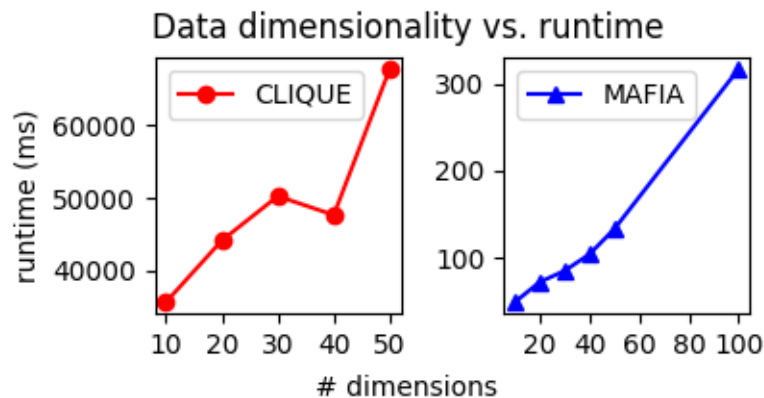
# MAFIA
axs[1].plot(dims, mafia_runtime, 'b^-', label='MAFIA')
axs[1].legend()
axs[1].xaxis.set_major_locator(ticker.MaxNLocator(5))

# Common x and y labels
fig.text(0.5, -0.02, '# dimensions', ha='center')
fig.text(-0.01, 0.5, 'runtime (ms)', va='center', rotation='vertical')

# Show plot
plt.suptitle('Data dimensionality vs. runtime')
plt.tight_layout(rect=[0, 0, 1, 1.1])
plt.show()

# Save plot
fig.savefig('figures/data_dimensionality_vs_runtime.png')

```



Both CLIQUE and MAFIA seems to scale linearly with the number of dimensions in the dataset.

## 1.6 Sensitivity of alpha

As observed in article. Alpha and beta controls the number of clusters and their quality.

- 1,000,000 data points
- 20 dims
- 5 clusters
- 10% outliers
- noiseMatrix: 1 3 5 7 8

### 1.6.1 Settings

-b 0.35 -M 40 -n 1000

-a 0.8 -> 5.2 (step size: 0.4)

### 1.6.2 Plot

```
[16]: import matplotlib.pyplot as plt
import numpy as np

# Data
alpha = np.arange(1.2, 9.6, 0.4)
clusters_found = [23, 11, 5, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 7, 7, 7, 22, 22, 22,
↪22, 22]

# Figure size
plt.figure(figsize=(2, 2))

# Plot
plt.plot(alpha, clusters_found, 'b^-')

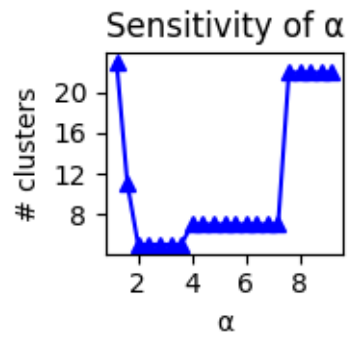
# Set number of x- and y-ticks
plt.gca().xaxis.set_major_locator(ticker.MaxNLocator(5))
plt.gca().yaxis.set_major_locator(ticker.MaxNLocator(5))

# Add labels and title
plt.xlabel(' ')
plt.ylabel('# clusters')
plt.title('Sensitivity of ')

# Show plot
plt.tight_layout()
plt.show()

# Save plot
```

```
plt.savefig('figures/sensitivity_alpha.png')
```



<Figure size 640x480 with 0 Axes>

MAFIA sensitivity to  $\alpha$  is not that high. The number of clusters found is almost the same if  $\alpha$  is somewhere between 2 and 7. After that, the number of clusters starts to increase.