In [1]:
```
!pip install pandas
!pip install seaborn
!pip install numpy
!pip install scipy
!pip install matplotlib
```

```
Requirement already satisfied: pandas in c:\anaconda3\anaconda3\lib\site-packages
(2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\anaconda3\anaconda3\li
b\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\anaconda3\anaconda3\lib\site-pac
kages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\anaconda3\anaconda3\lib\site-p
ackages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\anaconda3\anaconda3\lib\site-pa
ckages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\anaconda3\anaconda3\lib\site-package
s (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: seaborn in c:\anaconda3\anaconda3\lib\site-packages
(0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\anaconda3\anaconda3\lib
\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\anaconda3\anaconda3\lib\site-pac
kages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\anaconda3\anaconda3\l
ib\site-packages (from seaborn) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\anaconda3\anaconda3\lib\site
-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\anaconda3\anaconda3\lib\site-pac
kages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\anaconda3\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\anaconda3\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\anaconda3\anaconda3\lib\site-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\anaconda3\anaconda3\lib\site-pa
ckages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\anaconda3\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\anaconda3\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\anaconda3\anaconda3\lib\site-pac
kages (from pandas>=0.25->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\anaconda3\anaconda3\lib\site-p
ackages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\anaconda3\anaconda3\lib\site-package
s (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
Requirement already satisfied: numpy in c:\anaconda3\anaconda3\lib\site-packages
(1.24.3)
Requirement already satisfied: scipy in c:\anaconda3\anaconda3\lib\site-packages
(1.11.1)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in c:\anaconda3\anaconda3\lib
\site-packages (from scipy) (1.24.3)
Requirement already satisfied: matplotlib in c:\anaconda3\anaconda3\lib\site-packa
ges (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\anaconda3\anaconda3\lib\site
-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\anaconda3\anaconda3\lib\site-pac
kages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\anaconda3\anaconda3\lib\sit
e-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\anaconda3\anaconda3\lib\sit
e-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\anaconda3\anaconda3\lib\site-pack
ages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\anaconda3\anaconda3\lib\site-
packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\anaconda3\anaconda3\lib\site-pa
ckages (from matplotlib) (9.4.0)
```

```
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\anaconda3\anaconda3\lib
\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\anaconda3\anaconda3\lib
\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\anaconda3\anaconda3\lib\site-package
s (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [2]:
```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from matplotlib import colormaps
np.random.seed(41)
#plt.switch_backend('TkAgg')
from scipy.stats import spearmanr
def calculate_dissimilarity(rating):
    dissimilarity = (9 - rating) / 8
    return dissimilarity

adjacency_matrix = np.zeros((33, 33))
materials = {}
p = 0
pcount = 0



for filename in os.listdir('./data'):
    p = p+1
    filename = "A ("+ str(p)+").txt"

    data = pd.read_csv("./data/"+filename)
    if(data.shape[0] == 528):
        duplicated_pairs = data.iloc[:, [1, 2]].duplicated()
        if any(duplicated_pairs):
            continue
        else:
            pcount += 1
            for index, row in data.iterrows():
                material1, material2, rating = row[1], row[2], row[0]
                material1 = material1.replace(" ","")
                material2 = material2.replace(" ","")
                # Calculate dissimilarity
                dissimilarity = calculate_dissimilarity(rating)

                key = tuple(sorted([material1.strip(), material2.strip()]))

                if (key not in materials.keys()):
                    materials[key] = []
                if (key in materials.keys()):
                    materials[key].append(dissimilarity)



materials2 = {}
for key in materials.keys():
    if (key not in materials2.keys()):
        materials2[key] = []
    materials2[key] = materials[key]
    for i in range(0,len(key)):
        materials[key][i] = abs( materials[key][i])
    materials[key] = np.mean(np.array(materials[key]))
```

In [ ]:

In [3]:
```
materials_index = {}
for key in materials.keys():
    mat1 = key[0]
```

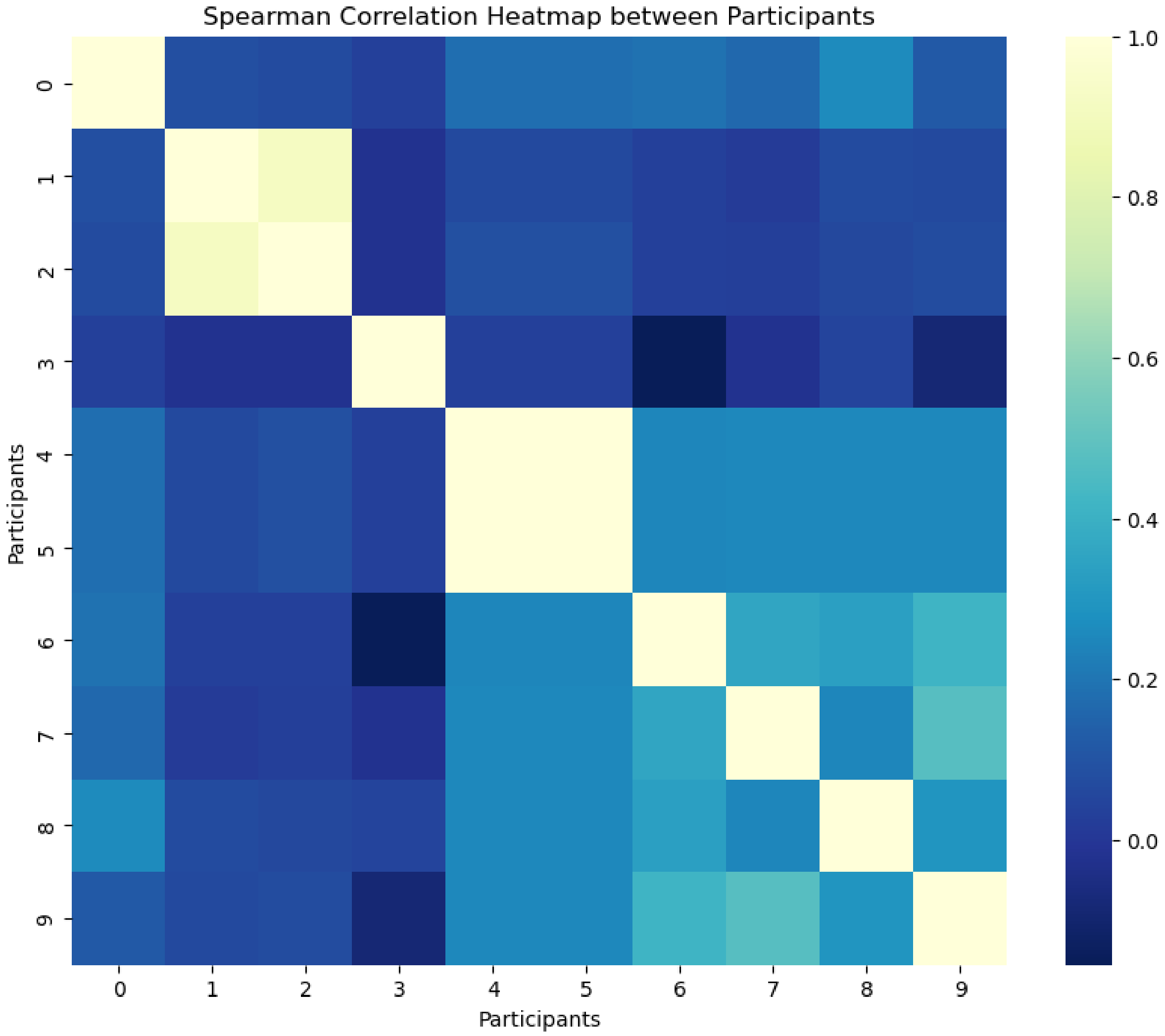
```
    mat2 = key[1]
    if(mat1 not in materials_index.keys()):
        materials_index[mat1]="0"
    if(mat2 not in materials_index.keys()):
        materials_index[mat2]="0"
```

In [4]:
```
i = 0
for key in materials_index.keys():
    materials_index[key] = i
    i = i+1
```

In [5]:
```
for key in materials.keys():
    index1 = materials_index[key[0]]
    index2 = materials_index[key[1]]
    adjacency_matrix[index1,index2] = materials[key]
    adjacency_matrix[index2,index1] = materials[key]
```

In [ ]:

In [6]:
```
participant_data = {}
for i in range(0,pcount):
    current_part = []
    for j in range(0,len(materials2.keys())):
        current_part.append(materials2[list(materials2.keys())[j]][i])
    if i not in participant_data.keys():
        participant_data[i] = []
    participant_data[i] = current_part

num_participants = len(participant_data)

# Calculate Spearman correlations for each participant across the surface pairs
spearman_corr = np.zeros((num_participants, num_participants))
for i, (pid1, data1) in enumerate(participant_data.items()):
    for j, (pid2, data2) in enumerate(participant_data.items()):
        spearman_corr[i, j] = spearmanr(data1, data2).correlation

# Plot heatmap for Spearman correlations between participants
plt.figure(figsize=(10, 8))
sns.heatmap(spearman_corr, cmap='YlGnBu_r', fmt='.2f')
plt.title('Spearman Correlation Heatmap between Participants')
plt.xlabel('Participants')
plt.ylabel('Participants')
plt.show()
```



In [7]:
```
mat_keys = {}
q = 0
for key in materials_index.keys():
    mat_keys[q] = key
    q = q+1
```

In [ ]:

In [8]:
```
def calculate_stress_with_disparity(X, disparities):
    n = len(X)
    stress = 0.0
    sum_squared_distances = 0.0
    sum_squared_disparities = 0.0

    for i in range(n):
        for j in range(i + 1, n):
            d_hat = np.linalg.norm(X[i] - X[j])  # Euclidean distance in reduced sp
            distance_ij = d_hat

            sum_squared_distances += distance_ij ** 2
            sum_squared_disparities += disparities[i, j] ** 2

            stress += (distance_ij - disparities[i, j]) ** 2

    stress = np.sqrt(stress / sum_squared_distances)
    return stress

def transform_to_disparity(dissimilarities):
    return np.square(dissimilarities)
```

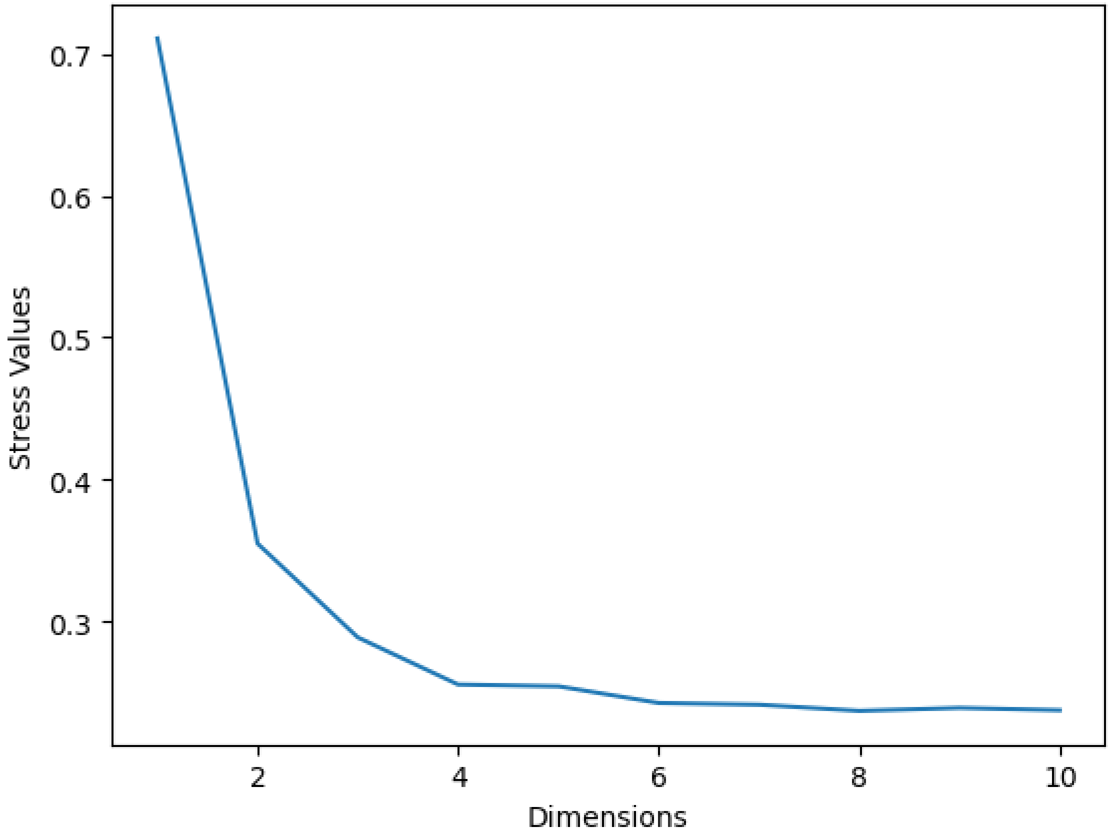
```
def nmds_with_disparity(dissimilarities, dimensions_range):
    n = len(dissimilarities)
    stress_values = []
    optimal_dimension = None
    optimal_stress = float('inf')
    optimal_X = None

    disparities = transform_to_disparity(dissimilarities)  # Convert dissimilaritie

    for dim in dimensions_range:
        X = np.random.rand(n, dim)

        # Gradient descent to minimize stress with disparities
        learning_rate = 0.01
        num_iterations = 50
        for _ in range(num_iterations):
            gradient = np.zeros((n, dim))

            for i in range(n):
                for j in range(i + 1, n):
                    d_hat = np.linalg.norm(X[i] - X[j])
                    delta_d = d_hat - disparities[i, j]

                    if d_hat != 0:
                        gradient[i] += 2 * delta_d * (X[i] - X[j]) / d_hat
                        gradient[j] += 2 * delta_d * (X[j] - X[i]) / d_hat

            # Update coordinates using gradient descent
            X -= learning_rate * gradient

        stress = calculate_stress_with_disparity(X, disparities)
        stress_values.append(stress)


    return stress_values

stress_val = nmds_with_disparity(adjacency_matrix,range(1,11))
```

In [9]:

```
plt.plot(range(1,11),stress_val)
plt.xlabel('Dimensions')
plt.ylabel('Stress Values')
```

Out[9]:

```
Text(0, 0.5, 'Stress Values')
```



In [10]:

```
import numpy as np
import matplotlib.pyplot as plt
diff = np.diff(np.array(stress_val))
curvature = np.diff(diff)
elbow_index = np.argmax(curvature) + 1
optim_stress_dim = elbow_index + 1
optimum_stress = stress_val[optim_stress_dim-1]
print("Optimum Stress: "+str(optimum_stress))
print("Optimum Dimension: "+str(optim_stress_dim))
```

```
Optimum Stress: 0.3548785836775515
Optimum Dimension: 2
```

In [ ]:

In [11]:

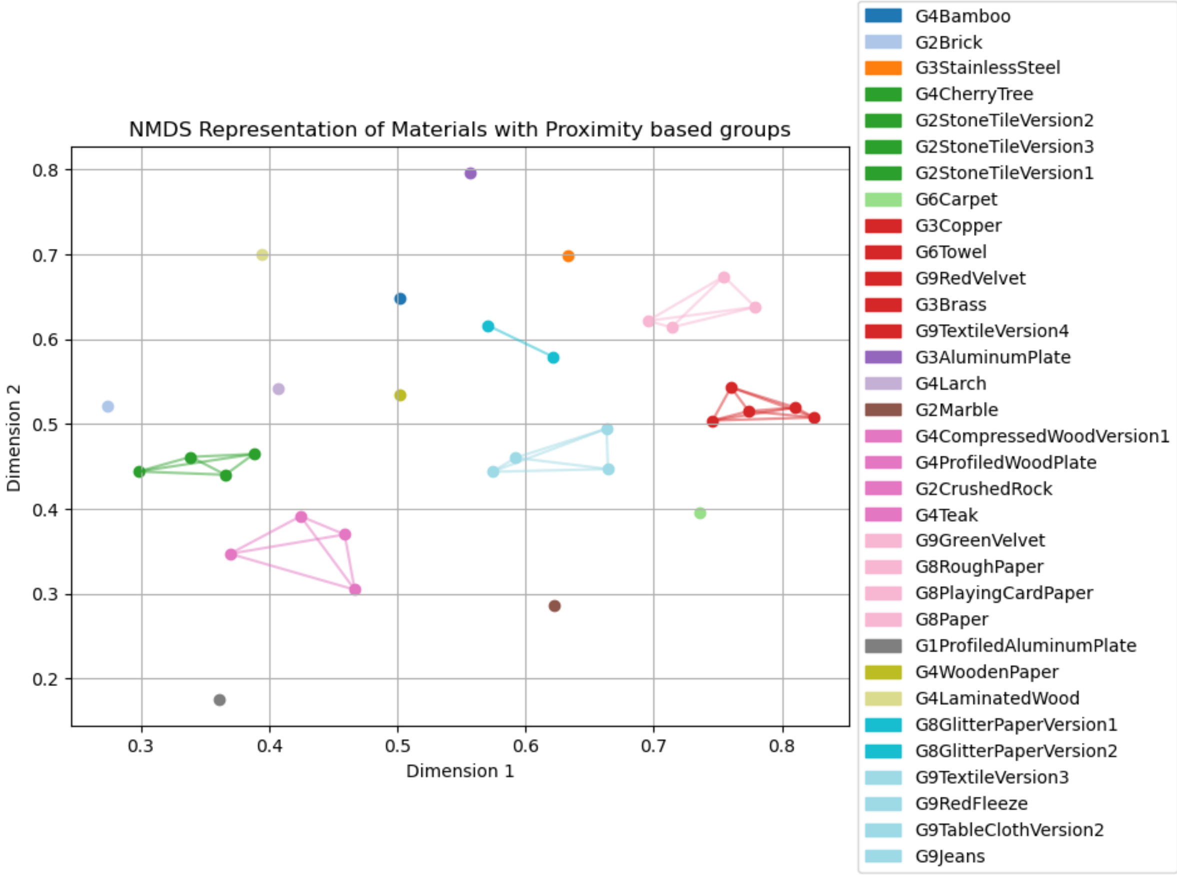
```
def get_optimal_X(dissimilarities, optimal_dimension):
    n = len(dissimilarities)
    disparities = transform_to_disparity(dissimilarities)  # Convert dissimilaritie
    X = np.random.rand(n, optimal_dimension)  # Random initialization for the giver

    # Gradient descent to minimize stress with disparities for the given dimension
    learning_rate = 0.01
    num_iterations = 50
    for _ in range(num_iterations):
        gradient = np.zeros((n, optimal_dimension))

        for i in range(n):
            for j in range(i + 1, n):
                d_hat = np.linalg.norm(X[i] - X[j])
                delta_d = d_hat - disparities[i, j]

                if d_hat != 0:
                    gradient[i] += 2 * delta_d * (X[i] - X[j]) / d_hat
                    gradient[j] += 2 * delta_d * (X[j] - X[i]) / d_hat

        # Update coordinates using gradient descent
```

```
        X -= learning_rate * gradient

    return X


optimal_X = get_optimal_X(adjacency_matrix, optim_stress_dim)
```

In [12]:

```
from matplotlib.patches import Patch
distances = np.linalg.norm(optimal_X[:, None] - optimal_X, axis=2)
distances_Y = distances
distance_matrix  = distances_Y

proximity_threshold = 0.075  # Set your desired proximity threshold

# Function to perform clustering based on proximity threshold
def cluster_points(distances, threshold):
    clusters = []
    n = len(distances)

    for i in range(n):
        assigned_to_clusters = []

        for cluster_index, cluster in enumerate(clusters):
            for point in cluster:
                if distances[i][point] < threshold:
                    assigned_to_clusters.append(cluster_index)
                    break

        if not assigned_to_clusters:
            new_cluster = [i]
            clusters.append(new_cluster)
        else:
            # Merge clusters if the point belongs to multiple clusters
            merged_cluster = [i]
            for index in assigned_to_clusters:
                merged_cluster += clusters[index]
                clusters[index] = []
            merged_cluster = list(set(merged_cluster))
            clusters = [c for c in clusters if c]
            clusters.append(merged_cluster)

    return [cluster for cluster in clusters if cluster]



clusters = cluster_points(distance_matrix, proximity_threshold)


colors = plt.cm.tab20(np.linspace(0, 1, len(clusters)))

plt.figure(figsize=(8, 6))

for k in range(len(clusters)):
    this_cluster = clusters[k]
    for points in this_cluster:
        plt.scatter(optimal_X[points][0],optimal_X[points][1], color=colors[k])


legend_elements = []
legend_labels = []

for i in range(len(optimal_X)):
    legend_elements.append(0)
    legend_labels.append(0)
```

```
for k in range(len(clusters)):
    this_cluster = clusters[k]
    for points in this_cluster:
        material_name = list(materials_index.keys())[points]
        legend_labels[points] = material_name
        legend_elements[points] = Patch(color=colors[k])


for k in range(len(clusters)):
    this_cluster = clusters[k]
    for indexi in range(len(this_cluster)):
            for indexij in range(indexi,len(this_cluster)):
                i = this_cluster[indexi]
                j = this_cluster[indexij]

                Xis = optimal_X[i]
                Xjs = optimal_X[j]
                points_X = [Xis[0],Xjs[0]]
                points_Y = [Xis[1],Xjs[1]]

                plt.plot(points_X, points_Y, color=colors[k], alpha=0.5)

new_legend = []
new_legend_labels = []

for k in range(len(clusters)):
    this_cluster = clusters[k]
    for points in this_cluster:
        new_legend.append(legend_elements[points])
        new_legend_labels.append(legend_labels[points])

final_matrix = np.zeros((len(optimal_X),len(optimal_X)),dtype="int")
for i in range(0,len(optimal_X)):
    for j in range(0, len(optimal_X)):
        if distances[i][j] < proximity_threshold:
            final_matrix[i][j] = 1

plt.legend(handles=new_legend,labels=new_legend_labels, loc='center left', bbox_to_
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('NMDS Representation of Materials with Proximity based groups')
plt.grid(True)
plt.show()
```



In [ ]:

In [13]:

```
num_rows = len(adjacency_matrix)
num_cols = len(adjacency_matrix[0])


existing_array = adjacency_matrix

index_row = np.arange(existing_array.shape[1], dtype="int").reshape(1, -1)

index_column = np.arange(existing_array.shape[0], dtype="int").reshape(-1, 1)

index_row_column = np.vstack((np.hstack((np.array([[-1]]), index_row)),
                              np.hstack((index_column, existing_array))))
adjacency_matrix_with_indices = index_row_column
num_rows = len(adjacency_matrix_with_indices)
num_cols = len(adjacency_matrix_with_indices[0])


float_array = adjacency_matrix_with_indices
format_func = lambda x: f'{x:.0f}' if x.is_integer() else f'{x:.2f}'

formatted_array = np.array([[format_func(x) for x in row] for row in float_array])
adjacency_matrix_with_indices = formatted_array


fig, axs = plt.subplots(1, 2, figsize=(11, 10), gridspec_kw={'width_ratios': [7, 4]


main_table = axs[0].table(cellText=adjacency_matrix_with_indices, loc='center', cel
                          )
axs[0].axis('off')  # Hide axis for the main table
axs[0].set_title('Dissimilarity Matrix')

legend = []
```

```
for i in range(0, len(materials_index.keys())):
    row = [str(i), list(materials_index.keys())[i]]
    legend.append(row)

sidebar_table = axs[1].table(cellText=legend, loc='center', cellLoc='center', edges
axs[1].axis('off')
axs[1].set_title('Materials')

for cell in main_table.get_celld().values():
    cell.set_fontsize(14)

for cell in sidebar_table.get_celld().values():
    cell.set_fontsize(16)

for row in range(0, num_rows):
    main_table[(row, 0)].set_facecolor('lightgray')  # Row colors
for col in range(0, num_cols):
    main_table[(0, col)].set_facecolor('lightgray')  # Index colors
main_table[(0,0)].set_facecolor("gray")
plt.subplots_adjust(wspace=0.1)
plt.subplots_adjust(left=0.05, right=0.95, bottom=0.1, top=0.9, wspace=0.4)

plt.tight_layout()

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

Dissimilarity Matrix

-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	0.39	0.57	0.45	0.57	0.45	0.53	0.49	0.34	0.46	0.36	0.57	0.71	0.62	0.56	0.62	0.51	0.70	0.70	0.31	0.35	0.70	0.64	0.42	0.40	0.55	0.36	0.49	0.59	0.36	0.54	0.54	0.61
1	0.39	0	0.69	0.41	0.66	0.45	0.39	0.46	0.29	0.55	0.41	0.46	0.50	0.65	0.46	0.55	0.54	0.69	0.54	0.49	0.50	0.66	0.64	0.61	0.45	0.72	0.39	0.62	0.61	0.44	0.45	0.54	0.41
2	0.57	0.69	0	0.49	0.53	0.55	0.59	0.57	0.69	0.57	0.62	0.49	0.34	0.34	0.49	0.53	0.45	0.45	0.56	0.49	0.60	0.33	0.23	0.49	0.60	0.31	0.47	0.28	0.79	0.38	0.65	0.46	0.39
3	0.45	0.41	0.49	0	0.44	0.55	0.61	0.44	0.56	0.55	0.69	0.44	0.34	0.34	0.54	0.42	0.45	0.24	0.59	0.34	0.57	0.20	0.29	0.47	0.55	0.31	0.45	0.41	0.68	0.53	0.50	0.44	0.25
4	0.57	0.66	0.53	0.44	0	0.44	0.76	0.47	0.57	0.76	0.72	0.31	0.31	0.49	0.60	0.17	0.53	0.53	0.29	0.54	0.61	0.59	0.54	0.59	0.53	0.36	0.55	0.36	0.75	0.49	0.46	0.36	0.54
5	0.45	0.45	0.55	0.55	0.44	0	0.64	0.44	0.46	0.69	0.55	0.40	0.62	0.54	0.57	0.46	0.44	0.60	0.49	0.47	0.35	0.59	0.54	0.45	0.55	0.51	0.60	0.42	0.64	0.39	0.54	0.54	0.61
6	0.53	0.39	0.59	0.61	0.76	0.64	0	0.24	0.51	0.51	0.45	0.60	0.62	0.60	0.44	0.69	0.57	0.64	0.57	0.45	0.42	0.68	0.65	0.65	0.34	0.66	0.49	0.76	0.57	0.60	0.51	0.60	0.56
7	0.49	0.46	0.57	0.44	0.47	0.44	0.24	0	0.41	0.39	0.40	0.68	0.70	0.69	0.28	0.59	0.59	0.64	0.54	0.54	0.51	0.47	0.70	0.53	0.44	0.75	0.44	0.75	0.60	0.47	0.54	0.69	0.46
8	0.34	0.29	0.69	0.56	0.57	0.46	0.51	0.41	0	0.59	0.39	0.61	0.71	0.61	0.40	0.75	0.70	0.54	0.71	0.47	0.36	0.41	0.56	0.56	0.80	0.79	0.36	0.72	0.44	0.51	0.60	0.62	0.35
9	0.46	0.55	0.57	0.55	0.76	0.69	0.51	0.39	0.59	0	0.54	0.79	0.80	0.66	0.53	0.74	0.65	0.70	0.71	0.47	0.35	0.39	0.71	0.53	0.65	0.72	0.21	0.53	0.57	0.44	0.62	0.49	0.49
10	0.36	0.41	0.62	0.69	0.72	0.55	0.45	0.40	0.39	0.54	0	0.62	0.56	0.62	0.35	0.51	0.65	0.64	0.57	0.50	0.56	0.54	0.66	0.50	0.57	0.61	0.60	0.68	0.60	0.46	0.39	0.51	0.42
11	0.57	0.46	0.49	0.44	0.31	0.40	0.60	0.68	0.61	0.79	0.62	0	0.62	0.62	0.47	0.25	0.55	0.50	0.21	0.66	0.71	0.60	0.40	0.39	0.36	0.50	0.60	0.34	0.64	0.36	0.65	0.45	0.56
12	0.71	0.50	0.34	0.34	0.31	0.62	0.62	0.70	0.71	0.80	0.56	0.62	0	0.24	0.76	0.55	0.41	0.30	0.46	0.47	0.59	0.36	0.38	0.53	0.46	0.42	0.59	0.29	0.74	0.57	0.72	0.44	0.30
13	0.62	0.65	0.34	0.34	0.49	0.54	0.60	0.69	0.61	0.66	0.62	0.62	0.24	0	0.64	0.47	0.30	0.30	0.71	0.42	0.70	0.35	0.45	0.59	0.70	0.44	0.64	0.33	0.79	0.54	0.69	0.40	0.51
14	0.56	0.46	0.49	0.54	0.60	0.57	0.44	0.28	0.40	0.53	0.35	0.47	0.76	0.64	0	0.68	0.51	0.71	0.70	0.40	0.56	0.50	0.61	0.50	0.49	0.70	0.49	0.59	0.53	0.38	0.45	0.44	0.47
15	0.62	0.55	0.53	0.42	0.17	0.46	0.69	0.59	0.75	0.74	0.51	0.25	0.55	0.47	0.68	0	0.39	0.35	0.42	0.41	0.56	0.59	0.36	0.47	0.55	0.26	0.71	0.64	0.66	0.51	0.47	0.54	0.44
16	0.51	0.54	0.45	0.45	0.53	0.44	0.57	0.59	0.70	0.65	0.65	0.55	0.41	0.30	0.51	0.39	0	0.44	0.76	0.53	0.59	0.34	0.35	0.38	0.53	0.39	0.53	0.51	0.65	0.49	0.74	0.55	0.50
17	0.70	0.69	0.45	0.24	0.53	0.60	0.64	0.64	0.54	0.70	0.64	0.50	0.30	0.30	0.71	0.35	0.44	0	0.60	0.33	0.50	0.12	0.44	0.47	0.56	0.38	0.66	0.34	0.76	0.53	0.60	0.46	0.29
18	0.70	0.54	0.56	0.59	0.29	0.49	0.57	0.54	0.71	0.71	0.57	0.21	0.46	0.71	0.70	0.42	0.76	0.60	0	0.64	0.55	0.54	0.56	0.49	0.51	0.51	0.81	0.60	0.64	0.54	0.50	0.71	0.64
19	0.31	0.49	0.49	0.34	0.54	0.47	0.45	0.54	0.47	0.47	0.50	0.66	0.47	0.42	0.40	0.41	0.53	0.33	0.64	0	0.53	0.39	0.40	0.44	0.59	0.47	0.45	0.65	0.69	0.40	0.62	0.38	0.23
20	0.35	0.50	0.60	0.57	0.61	0.35	0.42	0.51	0.36	0.35	0.56	0.71	0.59	0.70	0.56	0.56	0.59	0.50	0.55	0.53	0	0.35	0.46	0.54	0.54	0.61	0.44	0.57	0.46	0.38	0.54	0.62	0.44
21	0.70	0.66	0.33	0.20	0.59	0.59	0.68	0.47	0.41	0.39	0.54	0.60	0.36	0.35	0.50	0.59	0.34	0.12	0.54	0.39	0.35	0	0.28	0.42	0.57	0.51	0.39	0.40	0.54	0.61	0.74	0.53	0.31
22	0.64	0.64	0.23	0.29	0.54	0.54	0.65	0.70	0.56	0.71	0.66	0.40	0.38	0.45	0.61	0.36	0.35	0.44	0.56	0.40	0.46	0.28	0	0.33	0.70	0.35	0.66	0.35	0.68	0.55	0.41	0.36	0.40
23	0.42	0.61	0.49	0.47	0.59	0.45	0.65	0.53	0.56	0.53	0.50	0.39	0.53	0.59	0.50	0.47	0.38	0.47	0.49	0.44	0.54	0.42	0.33	0	0.68	0.46	0.56	0.59	0.72	0.36	0.55	0.14	0.29
24	0.40	0.45	0.60	0.55	0.53	0.55	0.34	0.44	0.80	0.65	0.57	0.36	0.46	0.70	0.49	0.55	0.53	0.56	0.51	0.59	0.54	0.57	0.70	0.68	0	0.68	0.60	0.59	0.78	0.44	0.64	0.51	0.59
25	0.55	0.72	0.31	0.31	0.36	0.51	0.66	0.75	0.79	0.72	0.61	0.50	0.42	0.44	0.70	0.26	0.39	0.38	0.51	0.47	0.61	0.51	0.35	0.46	0.68	0	0.78	0.46	0.68	0.51	0.35	0.46	0.64
26	0.36	0.39	0.47	0.45	0.55	0.60	0.49	0.44	0.36	0.21	0.60	0.60	0.59	0.64	0.49	0.71	0.53	0.66	0.81	0.45	0.44	0.39	0.66	0.56	0.60	0.78	0	0.62	0.51	0.46	0.68	0.46	0.49
27	0.49	0.62	0.28	0.41	0.36	0.42	0.76	0.75	0.72	0.53	0.68	0.34	0.29	0.33	0.59	0.64	0.51	0.34	0.60	0.65	0.57	0.40	0.35	0.59	0.59	0.46	0.62	0	0.79	0.64	0.49	0.51	0.57
28	0.59	0.61	0.79	0.68	0.75	0.64	0.57	0.60	0.44	0.57	0.60	0.64	0.74	0.79	0.53	0.66	0.65	0.76	0.64	0.69	0.46	0.54	0.68	0.72	0.78	0.68	0.51	0.79	0	0.60	0.50	0.78	0.72
29	0.36	0.44	0.38	0.53	0.49	0.39	0.60	0.47	0.51	0.44	0.46	0.36	0.57	0.54	0.38	0.51	0.49	0.53	0.54	0.40	0.38	0.61	0.55	0.36	0.44	0.51	0.46	0.64	0.60	0	0.56	0.33	0.41
30	0.54	0.45	0.65	0.50	0.46	0.54	0.51	0.54	0.60	0.62	0.39	0.65	0.72	0.69	0.45	0.47	0.74	0.60	0.50	0.62	0.54	0.74	0.41	0.55	0.64	0.35	0.68	0.49	0.50	0.56	0	0.62	0.60
31	0.54	0.54	0.46	0.44	0.36	0.54	0.60	0.69	0.62	0.49	0.51	0.45	0.44	0.40	0.44	0.54	0.55	0.46	0.71	0.38	0.62	0.53	0.36	0.14	0.51	0.46	0.46	0.51	0.78	0.33	0.62	0	0.53
32	0.61	0.41	0.39	0.25	0.54	0.61	0.56	0.46	0.35	0.49	0.42	0.56	0.30	0.51	0.47	0.44	0.50	0.29	0.64	0.23	0.44	0.31	0.40	0.29	0.59	0.64	0.49	0.57	0.72	0.41	0.60	0.53	0

Materials

0	G4ProfiledWoodPlate
1	G4Teak
2	G8RoughPaper
3	G9RedFleeze
4	G3Copper
5	G4Bamboo
6	G2StoneTileVersion1
7	G2StoneTileVersion2
8	G4CompressedWoodVersion1
9	G2Brick
10	G4CherryTree
11	G3StainlessSteel
12	G6Towel
13	G9RedVelvet
14	G2StoneTileVersion3
15	G3Brass
16	G6Carpet
17	G9TextileVersion4
18	G3AluminumPlate
19	G9TableClothVersion2
20	G4Larch
21	G9Jeans
22	G8Paper
23	G8GlitterPaperVersion1
24	G2Marble
25	G9GreenVelvet
26	G2CrushedRock
27	G8PlayingCardPaper
28	G1ProfiledAluminumPlate
29	G4WoodenPaper
30	G4LaminatedWood
31	G8GlitterPaperVersion2
32	G9TextileVersion3

In [14]:

```
distances_Y = distances

num_rows = len(distances_Y)
```

```
num_cols = len(distances_Y[0])

dist_trunc = np.round(distances_Y, 2)
num_rows = len(dist_trunc)
num_cols = len(dist_trunc[0])

existing_array = dist_trunc

index_row = np.arange(existing_array.shape[1], dtype="int").reshape(1, -1)

index_column = np.arange(existing_array.shape[0], dtype="int").reshape(-1, 1)

index_row_column = np.vstack((np.hstack((np.array([[-1]]), index_row)),
                              np.hstack((index_column, existing_array))))
dist_trunc = index_row_column
num_rows = len(dist_trunc)
num_cols = len(dist_trunc[0])
# Colors for the table cells (modify as needed)
cell_colors = [['lightgreen' if i == j else 'yellow' if dist_trunc[i][j] < proximit
                for j in range(num_cols)] for i in range(num_rows)]
float_array = dist_trunc
format_func = lambda x: f'{x:.0f}' if x.is_integer() else f'{x:.2f}'

formatted_array = np.array([[format_func(x) for x in row] for row in float_array])
dist_trunc = formatted_array


fig, axs = plt.subplots(1, 2, figsize=(11, 10), gridspec_kw={'width_ratios': [7, 4]


main_table = axs[0].table(cellText=dist_trunc, loc='center', cellLoc='center', edge
                          cellColours=cell_colors)
axs[0].axis('off')  # Hide axis for the main table
axs[0].set_title('Distance Matrix')

legend = []
for i in range(0, len(materials_index.keys())):
    row = [str(i), list(materials_index.keys())[i]]
    legend.append(row)

sidebar_table = axs[1].table(cellText=legend, loc='center', cellLoc='center', edges
axs[1].axis('off')
axs[1].set_title('Materials')

for cell in main_table.get_celld().values():
    cell.set_fontsize(14)

for cell in sidebar_table.get_celld().values():
    cell.set_fontsize(16)

for row in range(0, num_rows):
    main_table[(row, 0)].set_facecolor('lightgray')  # Row colors
for col in range(0, num_cols):
    main_table[(0, col)].set_facecolor('lightgray')  # Index colors
main_table[(0,0)].set_facecolor("gray")
plt.subplots_adjust(wspace=0.1)
plt.subplots_adjust(left=0.05, right=0.95, bottom=0.1, top=0.9, wspace=0.4)

plt.tight_layout()

plt.show()
```

Distance Matrix

-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0.04	0.35	0.24	0.35	0.28	0.18	0.15	0.09	0.24	0.12	0.37	0.38	0.39	0.12	0.32	0.28
1	0.04	0	0.36	0.26	0.37	0.27	0.14	0.11	0.07	0.20	0.08	0.37	0.41	0.42	0.08	0.34	0.31
2	0.35	0.36	0	0.13	0.12	0.21	0.45	0.40	0.44	0.45	0.39	0.12	0.14	0.15	0.36	0.11	0.22
3	0.24	0.26	0.13	0	0.11	0.22	0.37	0.33	0.33	0.39	0.30	0.21	0.15	0.16	0.28	0.08	0.12
4	0.35	0.37	0.12	0.11	0	0.30	0.48	0.44	0.44	0.50	0.41	0.23	0.04	0.05	0.39	0.03	0.13
5	0.28	0.27	0.21	0.22	0.30	0	0.29	0.25	0.33	0.26	0.25	0.14	0.33	0.35	0.22	0.28	0.34
6	0.18	0.14	0.45	0.37	0.48	0.29	0	0.04	0.12	0.08	0.07	0.42	0.52	0.53	0.09	0.45	0.44
7	0.15	0.11	0.40	0.33	0.44	0.25	0.04	0	0.12	0.09	0.03	0.38	0.48	0.49	0.05	0.41	0.40
8	0.09	0.07	0.44	0.33	0.44	0.33	0.12	0.12	0	0.20	0.09	0.44	0.47	0.48	0.12	0.41	0.37
9	0.24	0.20	0.45	0.39	0.50	0.26	0.08	0.09	0.20	0	0.12	0.40	0.54	0.55	0.13	0.47	0.48
10	0.12	0.08	0.39	0.30	0.41	0.25	0.07	0.03	0.09	0.12	0	0.37	0.45	0.46	0.03	0.39	0.37
11	0.37	0.37	0.12	0.21	0.23	0.14	0.42	0.38	0.44	0.40	0.37	0	0.25	0.27	0.34	0.23	0.32
12	0.38	0.41	0.14	0.15	0.04	0.33	0.52	0.48	0.47	0.54	0.45	0.25	0	0.02	0.43	0.07	0.14
13	0.39	0.42	0.15	0.16	0.05	0.35	0.53	0.49	0.48	0.55	0.46	0.27	0.02	0	0.44	0.08	0.14
14	0.12	0.08	0.36	0.28	0.39	0.22	0.09	0.05	0.12	0.13	0.03	0.34	0.43	0.44	0	0.36	0.35
15	0.32	0.34	0.11	0.08	0.03	0.28	0.45	0.41	0.41	0.47	0.39	0.23	0.07	0.08	0.36	0	0.11
16	0.28	0.31	0.22	0.12	0.13	0.34	0.44	0.40	0.37	0.48	0.37	0.32	0.14	0.14	0.35	0.11	0
17	0.35	0.37	0.08	0.11	0.03	0.28	0.47	0.43	0.44	0.49	0.41	0.20	0.06	0.07	0.38	0.04	0.15
18	0.44	0.43	0.24	0.32	0.35	0.16	0.44	0.40	0.49	0.39	0.40	0.12	0.38	0.39	0.37	0.35	0.44
19	0.14	0.16	0.22	0.10	0.21	0.22	0.28	0.24	0.23	0.31	0.21	0.26	0.25	0.26	0.19	0.18	0.17
20	0.18	0.15	0.31	0.26	0.37	0.14	0.15	0.11	0.20	0.14	0.11	0.28	0.40	0.42	0.08	0.34	0.36
21	0.22	0.25	0.17	0.05	0.13	0.26	0.37	0.33	0.31	0.40	0.30	0.25	0.16	0.17	0.28	0.10	0.09
22	0.35	0.36	0.02	0.13	0.13	0.20	0.44	0.39	0.43	0.43	0.38	0.10	0.15	0.17	0.35	0.13	0.23
23	0.27	0.27	0.14	0.15	0.23	0.08	0.32	0.28	0.34	0.31	0.27	0.10	0.26	0.28	0.24	0.21	0.28
24	0.18	0.22	0.34	0.21	0.27	0.38	0.36	0.33	0.26	0.42	0.30	0.41	0.30	0.30	0.29	0.25	0.16
25	0.42	0.43	0.07	0.20	0.16	0.25	0.51	0.47	0.51	0.50	0.45	0.12	0.16	0.18	0.42	0.17	0.28
26	0.07	0.10	0.40	0.27	0.37	0.34	0.22	0.20	0.11	0.29	0.17	0.43	0.41	0.41	0.18	0.34	0.28
27	0.42	0.43	0.07	0.18	0.12	0.28	0.52	0.47	0.50	0.52	0.46	0.16	0.12	0.14	0.43	0.14	0.25
28	0.22	0.23	0.56	0.44	0.53	0.49	0.28	0.29	0.17	0.36	0.27	0.59	0.57	0.57	0.29	0.51	0.43
29	0.17	0.16	0.23	0.17	0.27	0.11	0.22	0.18	0.23	0.23	0.17	0.21	0.31	0.32	0.13	0.25	0.27
30	0.34	0.31	0.33	0.34	0.42	0.12	0.27	0.25	0.35	0.22	0.26	0.24	0.45	0.47	0.24	0.40	0.46
31	0.27	0.27	0.10	0.09	0.17	0.14	0.35	0.31	0.34	0.35	0.29	0.12	0.20	0.22	0.26	0.15	0.22
32	0.16	0.18	0.20	0.08	0.19	0.21	0.29	0.25	0.25	0.32	0.23	0.24	0.23	0.24	0.20	0.16	0.16

-1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0.35	0.44	0.14	0.18	0.22	0.35	0.27	0.18	0.42	0.07	0.42	0.22	0.17	0.34	0.27	0.16
1	0.37	0.43	0.16	0.15	0.25	0.36	0.27	0.22	0.43	0.10	0.43	0.23	0.16	0.31	0.27	0.18
2	0.08	0.24	0.22	0.31	0.17	0.02	0.14	0.34	0.07	0.40	0.07	0.56	0.23	0.33	0.10	0.20
3	0.11	0.32	0.10	0.26	0.05	0.13	0.15	0.21	0.20	0.27	0.18	0.44	0.17	0.34	0.09	0.08
4	0.03	0.35	0.21	0.37	0.13	0.13	0.23	0.27	0.16	0.37	0.12	0.53	0.27	0.42	0.17	0.19
5	0.28	0.16	0.22	0.14	0.26	0.20	0.08	0.38	0.25	0.34	0.28	0.49	0.11	0.12	0.14	0.21
6	0.47	0.44	0.28	0.15	0.37	0.44	0.32	0.36	0.51	0.22	0.52	0.28	0.22	0.27	0.35	0.29
7	0.43	0.40	0.24	0.11	0.33	0.39	0.28	0.33	0.47	0.20	0.47	0.29	0.18	0.25	0.31	0.25
8	0.44	0.49	0.23	0.20	0.31	0.43	0.34	0.26	0.51	0.11	0.50	0.17	0.23	0.35	0.34	0.25
9	0.49	0.39	0.31	0.14	0.40	0.43	0.31	0.42	0.50	0.29	0.52	0.36	0.23	0.22	0.35	0.32
10	0.41	0.40	0.21	0.11	0.30	0.38	0.27	0.30	0.45	0.17	0.46	0.27	0.17	0.26	0.29	0.23
11	0.20	0.12	0.26	0.28	0.25	0.10	0.10	0.41	0.12	0.43	0.16	0.59	0.21	0.24	0.12	0.24
12	0.06	0.38	0.25	0.40	0.16	0.15	0.26	0.30	0.16	0.41	0.12	0.57	0.31	0.45	0.20	0.23
13	0.07	0.39	0.26	0.42	0.17	0.17	0.28	0.30	0.18	0.41	0.14	0.57	0.32	0.47	0.22	0.24
14	0.38	0.37	0.19	0.08	0.28	0.35	0.24	0.29	0.42	0.18	0.43	0.29	0.13	0.24	0.26	0.20
15	0.04	0.35	0.18	0.34	0.10	0.13	0.21	0.25	0.17	0.34	0.14	0.51	0.25	0.40	0.15	0.16
16	0.15	0.44	0.17	0.36	0.09	0.23	0.28	0.16	0.28	0.28	0.25	0.43	0.27	0.46	0.22	0.16
17	0	0.32	0.21	0.35	0.14	0.10	0.20	0.29	0.13	0.38	0.10	0.54	0.26	0.40	0.14	0.19
18	0.32	0	0.35	0.29	0.36	0.22	0.18	0.51	0.23	0.50	0.27	0.65	0.27	0.19	0.23	0.34
19	0.21	0.35	0	0.19	0.09	0.22	0.17	0.17	0.29	0.18	0.28	0.34	0.12	0.31	0.14	0.02
20	0.35	0.29	0.19	0	0.27	0.30	0.18	0.33	0.37	0.24	0.38	0.37	0.09	0.16	0.22	0.20
21	0.14	0.36	0.09	0.27	0	0.18	0.19	0.17	0.24	0.24	0.22	0.41	0.18	0.37	0.14	0.07
22	0.10	0.22	0.22	0.30	0.18	0	0.13	0.34	0.08	0.39	0.09	0.56	0.21	0.31	0.09	0.19
23	0.20	0.18	0.17	0.18	0.19	0.13	0	0.33	0.19	0.33	0.21	0.49	0.11	0.19	0.06	0.16
24	0.29	0.51	0.17	0.33	0.17	0.34	0.33	0	0.41	0.16	0.39	0.28	0.28	0.47	0.29	0.18
25	0.13	0.23	0.29	0.37	0.24	0.08	0.19	0.41	0	0.47	0.04	0.64	0.29	0.36	0.16	0.27
26	0.38	0.50	0.18	0.24	0.24	0.39	0.33	0.16	0.47	0	0.46	0.17	0.23	0.40	0.32	0.20
27	0.10	0.27	0.28	0.38	0.22	0.09	0.21	0.39	0.04	0.46	0	0.62	0.30	0.39	0.17	0.26
28	0.54	0.65	0.34	0.37	0.41	0.56	0.49	0.28	0.64	0.17	0.62	0	0.39	0.53	0.48	0.37
29	0.26	0.27	0.12	0.09	0.18	0.21	0.11	0.28	0.29	0.23	0.30	0.39	0	0.20	0.13	0.12
30	0.40	0.19	0.31	0.16	0.37	0.31	0.19	0.47	0.36	0.40	0.39	0.53	0.20	0	0.26	0.31
31	0.14	0.23	0.14	0.22	0.14	0.09	0.06	0.29	0.16	0.32	0.17	0.48	0.13	0.26	0	0.12
32	0.19	0.34	0.02	0.20	0.07	0.19	0.16	0.18	0.27	0.20	0.26	0.37	0.12	0.31	0.12	0

Materials

0	G4ProfiledWoodPlate
1	G4Teak
2	G8RoughPaper
3	G9RedFleeze
4	G3Copper
5	G4Bamboo
6	G2StoneTileVersion1
7	G2StoneTileVersion2
8	G4CompressedWoodVersion1
9	G2Brick
10	G4CherryTree
11	G3StainlessSteel
12	G6Towel
13	G9RedVelvet
14	G2StoneTileVersion3
15	G3Brass
16	G6Carpet
17	G9TextileVersion4
18	G3AluminumPlate
19	G9TableClothVersion2
20	G4Larch
21	G9Jeans
22	G8Paper
23	G8GlitterPaperVersion1
24	G2Marble
25	G9GreenVelvet
26	G2CrushedRock
27	G8PlayingCardPaper
28	G1ProfiledAluminumPlate
29	G4WoodenPaper
30	G4LaminatedWood
31	G8GlitterPaperVersion2
32	G9TextileVersion3

In [ ]:

In [15]:

```
num_rows = len(final_matrix)
num_cols = len(final_matrix)


existing_array = final_matrix

index_row = np.arange(existing_array.shape[1],dtype="int").reshape(1, -1)

index_column = np.arange(existing_array.shape[0],dtype="int").reshape(-1, 1)

index_row_column = np.vstack((np.hstack((np.array([[-1]]), index_row)),
                              np.hstack((index_column, existing_array))))


final_matrix_with_indices = index_row_column

float_array = final_matrix_with_indices
format_func = lambda x: f'{x:.0f}' if x.is_integer() else f'{x:.2f}'

formatted_array = np.array([[format_func(x) for x in row] for row in float_array])

final_matrix_with_indices = formatted_array




fig, axs = plt.subplots(1, 2, figsize=(11, 10), gridspec_kw={'width_ratios': [7, 4]
```

```
main_table = axs[0].table(cellText=final_matrix_with_indices, loc='center', cellLoc
                          cellColours=cell_colors)
axs[0].axis('off')
axs[0].set_title('Adjacency Matrix based on edges between points of same cluster')

legend = []
for i in range(0, len(materials_index.keys())):
    row = [str(i), list(materials_index.keys())[i]]
    legend.append(row)

sidebar_table = axs[1].table(cellText=legend, loc='center', cellLoc='center', edges
axs[1].axis('off')
axs[1].set_title('Materials')

for cell in main_table.get_celld().values():
    cell.set_fontsize(14)

for cell in sidebar_table.get_celld().values():
    cell.set_fontsize(16)

for row in range(1, num_rows + 1):
    main_table[(row, 0)].set_facecolor('lightgray')  # Row colors
for col in range(1, num_cols + 1):
    main_table[(0, col)].set_facecolor('lightgray')  # Index colors
main_table[(0,0)].set_facecolor('gray')  # Index colors
plt.subplots_adjust(wspace=0.1)
plt.subplots_adjust(left=0.05, right=0.95, bottom=0.1, top=0.9, wspace=0.4)

plt.tight_layout()

plt.show()
```

Adjacency Matrix based on edges between points of same cluster

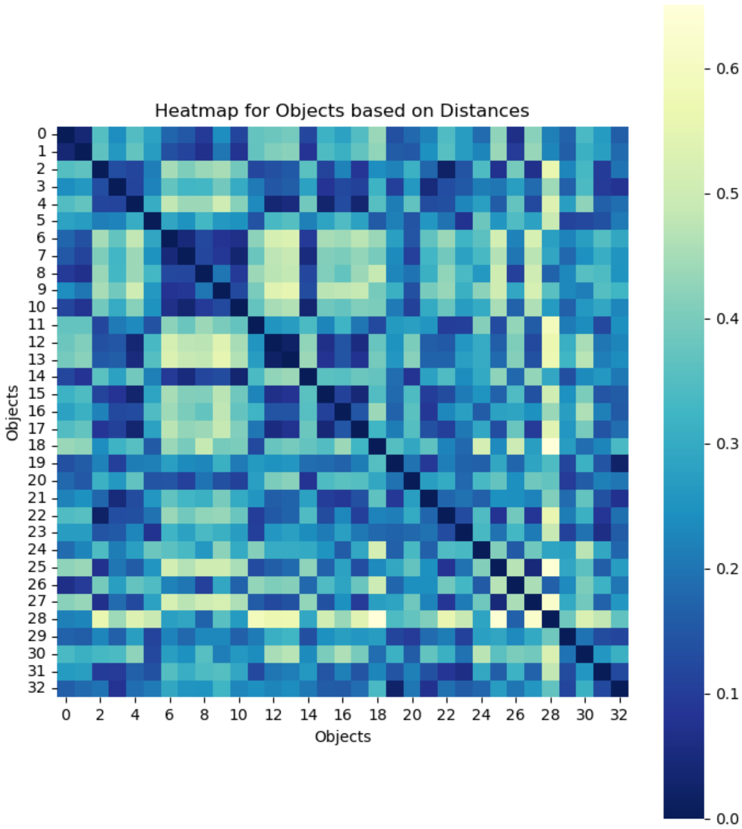
-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
22	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
25	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
26	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
27	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1

Materials

0	G4ProfiledWoodPlate
1	G4Teak
2	G8RoughPaper
3	G9RedFleeze
4	G3Copper
5	G4Bamboo
6	G2StoneTileVersion1
7	G2StoneTileVersion2
8	G4CompressedWoodVersion1
9	G2Brick
10	G4CherryTree
11	G3StainlessSteel
12	G6Towel
13	G9RedVelvet
14	G2StoneTileVersion3
15	G3Brass
16	G6Carpet
17	G9TextileVersion4
18	G3AluminumPlate
19	G9TableClothVersion2
20	G4Larch
21	G9Jeans
22	G8Paper
23	G8GlitterPaperVersion1
24	G2Marble
25	G9GreenVelvet
26	G2CrushedRock
27	G8PlayingCardPaper
28	G1ProfiledAluminumPlate
29	G4WoodenPaper
30	G4LaminatedWood
31	G8GlitterPaperVersion2
32	G9TextileVersion3

In [16]:

```
fig, axs = plt.subplots(1, 2, figsize=(11, 8),gridspec_kw={'width_ratios': [7, 4]})

sns.heatmap(distances_Y, cmap='YlGnBu_r', fmt='.2f', ax=axs[0], square = True)  # S
axs[0].set_title('Heatmap for Objects based on Distances')
axs[0].set_xlabel('Objects')
axs[0].set_ylabel('Objects')
sidebar_table = axs[1].table(cellText=legend, loc='center', cellLoc='center', edges
axs[1].axis('off')
axs[1].set_title('Materials')

for cell in main_table.get_celld().values():
    cell.set_fontsize(14)  # Change font size for main table text

for cell in sidebar_table.get_celld().values():
    cell.set_fontsize(16)  #

plt.subplots_adjust(wspace=0.05)
plt.subplots_adjust(left=0.05, right=0.95, bottom=0.1, top=0.9, wspace=0.4)

plt.tight_layout()

plt.show()
```

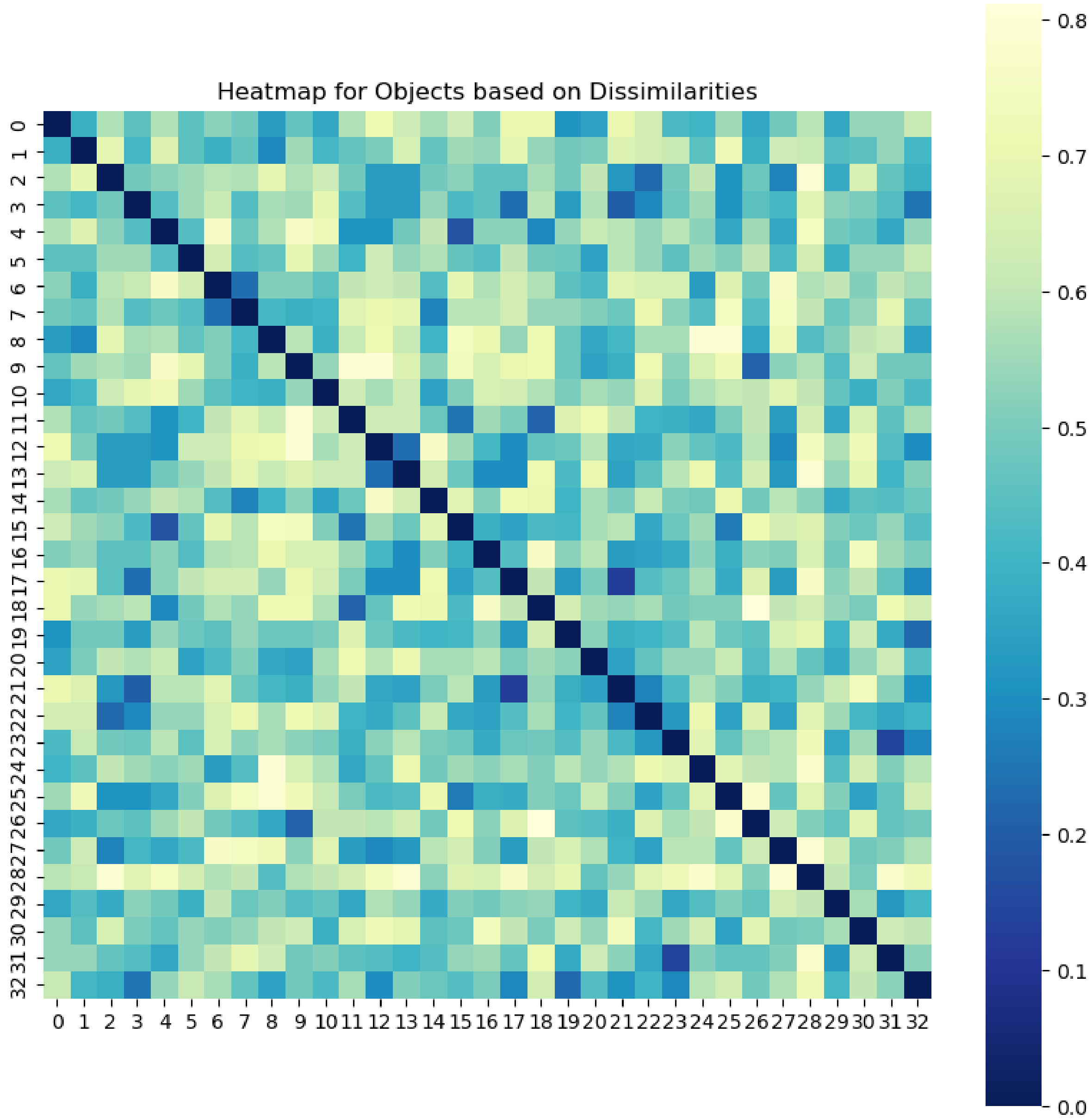


Materials

0	G4ProfiledWoodPlate
1	G4Teak
2	G8RoughPaper
3	G9RedFleeze
4	G3Copper
5	G4Bamboo
6	G2StoneTileVersion1
7	G2StoneTileVersion2
8	G4CompressedWoodVersion1
9	G2Brick
10	G4CherryTree
11	G3StainlessSteel
12	G6Towel
13	G9RedVelvet
14	G2StoneTileVersion3
15	G3Brass
16	G6Carpet
17	G9TextileVersion4
18	G3AluminumPlate
19	G9TableClothVersion2
20	G4Larch
21	G9Jeans
22	G8Paper
23	G8GlitterPaperVersion1
24	G2Marble
25	G9GreenVelvet
26	G2CrushedRock
27	G8PlayingCardPaper
28	G1ProfiledAluminumPlate
29	G4WoodenPaper
30	G4LaminatedWood
31	G8GlitterPaperVersion2
32	G9TextileVersion3

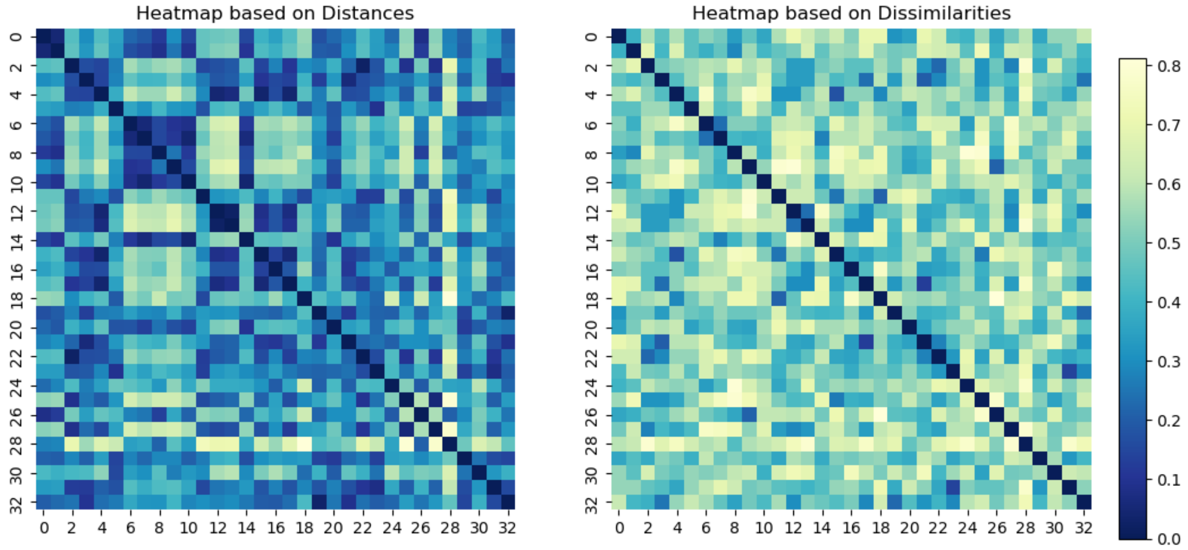
In [17]:

```
plt.figure(figsize=(10, 10))
sns.heatmap(adjacency_matrix, cmap='YlGnBu_r', fmt='.2f', square = True)
plt.title('Heatmap for Objects based on Dissimilarities')

plt.show()
```



In [18]:

```
fig, axs = plt.subplots(1, 2, figsize=(12, 6), gridspec_kw={'width_ratios': [5, 5]}

# Plot the first heatmap on the left subplot (axs[0])
heatmap1 = sns.heatmap(distances_Y, cmap='YlGnBu_r', fmt='.2f', square=True, cbar=F
axs[0].set_title('Heatmap based on Distances')

# Plot the second heatmap on the right subplot (axs[1])
heatmap2 = sns.heatmap(adjacency_matrix, cmap='YlGnBu_r', fmt='.2f', square=True, c
axs[1].set_title('Heatmap based on Dissimilarities')

# Get the maximum height of the two subplots
max_height = max(axs[0].get_position().height, axs[1].get_position().height)

# Create a color bar based on the maximum height of subplots
cbar_ax = fig.add_axes([0.92, 0.1, 0.02, max_height])  # Position for color bar
fig.colorbar(heatmap2.collections[0], cax=cbar_ax)  # Use the collections attribute


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



In [19]:

```
distance_Adj_matrix = np.round(distances_Y,2)
distance_dataframe = pd.DataFrame(distance_Adj_matrix)
distance_dataframe.to_csv("./distance_adjacency_matrix.csv", header=False, index=Fa
diss_dataframe = pd.DataFrame(adjacency_matrix)
diss_dataframe.to_csv("./dissimilarity_adjacency_matrix.csv", header=False, index=F
final_matrix_csv = pd.DataFrame(final_matrix)
final_matrix_csv.to_csv("./final_adjacency_matrix_edges.csv", header=False, index=F

mymaterials_with_OUT_INDEX = list(materials_index.keys())

distance_dataframe = pd.read_csv("./distance_adjacency_matrix.csv", header=None)
diss_dataframe = pd.read_csv("./dissimilarity_adjacency_matrix.csv", header=None)
final_matrix_csv = pd.read_csv("./final_adjacency_matrix_edges.csv", header=None)

distance_dataframe.columns = mymaterials_with_OUT_INDEX
distance_dataframe.index = mymaterials_with_OUT_INDEX

diss_dataframe.columns = mymaterials_with_OUT_INDEX
diss_dataframe.index = mymaterials_with_OUT_INDEX

final_matrix_csv.columns = mymaterials_with_OUT_INDEX
final_matrix_csv.index = mymaterials_with_OUT_INDEX

# Save modified DataFrames back to CSV files
distance_dataframe.to_csv("./distance_adjacency_matrix.csv")
diss_dataframe.to_csv("./dissimilarity_adjacency_matrix.csv")
final_matrix_csv.to_csv("./final_adjacency_matrix_edges.csv")
```

In [20]:

```
disparity_matrix = transform_to_disparity(adjacency_matrix)
flatten_dissimilarity = []
flatten_disparity = []
flatten_Y = []
for j in range(0, adjacency_matrix.shape[0]):
    for i in range(0,j):
        flatten_dissimilarity.append(adjacency_matrix[i][j])
        flatten_disparity.append(disparity_matrix[i][j])
        flatten_Y.append(distances_Y[i][j])
pairs = []

for i in range(0, len(flatten_disparity)):
    pairs.append((flatten_dissimilarity[i],flatten_disparity[i]))

plotx = []
ploty = []
sorted_pairs = sorted(pairs, key=lambda x: x[0])
```

```
for k in range(0,len(sorted_pairs)):
    plotx.append(sorted_pairs[k][0])
    ploty.append(sorted_pairs[k][1])
```

In [21]:
```
plt.figure(figsize=(8, 6))
plt.plot(plotx, ploty, color='red', label='Disparities', alpha=0.7)
plt.title('Shepard Plot')
plt.xlabel('Dissimalirities')
plt.ylabel('Distances/Disparities')
plt.grid(True)

# Adding the line plot representing the sorted pairs
plt.scatter(flatten_dissimilarity,flatten_Y, color='blue', label='Distances')
plt.legend()

plt.show()
```