I see the issue now. The problem occurs because the plt.cm.tab20b colormap is being used, which only contains 20 unique colors. If there are more than 20 unique affiliations, the colors start repeating. To handle this, we can use a larger colormap or combine multiple colormaps to ensure each unique affiliation gets a unique color. Here’s how you can modify the code to achieve this:

1. Use multiple colormaps if there are more unique affiliations than colors in a single colormap.
2. Ensure each unique affiliation has a unique color by concatenating multiple colormaps if necessary.

Here’s the updated code:

import pandas as pd

from pyvis.network import Network

import matplotlib.pyplot as plt

import matplotlib.colors as mcolors

import numpy as np

# Read discussion network data from a csv file

dnet\_df = pd.read\_csv('discussion\_network\_with\_affiliations.csv')

discussion\_df = pd.read\_csv('discussion\_data.csv') # Assuming this file contains user\_id and affiliation

# Aggregate the data by contributor\_affiliation and consumer\_affiliation, sum the values of consumption

dnet\_grouped = dnet\_df.groupby(['contributor\_affiliation', 'consumer\_affiliation'])['consumption'].sum().reset\_index()

# Sort the DataFrame by consumption (highest to lowest)

dnet\_grouped\_sorted = dnet\_grouped.sort\_values(by='consumption', ascending=False)

# Create a pyvis network with a white background

net = Network(height="750px", width="100%", bgcolor="white", font\_color="black", directed=True)

# Create an empty dictionary to store added nodes

added\_nodes = {}

# Define the minimum weight threshold for displaying edges

min\_weight\_threshold = 3

# Create a dictionary to store the count of rows for each pair of interchangeable affiliations

interchangeable\_counts = {}

# Count the occurrences of each pair of interchangeable affiliations

for \_, row in dnet\_grouped.iterrows():

contributor\_affiliation = row['contributor\_affiliation']

consumer\_affiliation = row['consumer\_affiliation']

if contributor\_affiliation > consumer\_affiliation:

pair = (consumer\_affiliation, contributor\_affiliation)

else:

pair = (contributor\_affiliation, consumer\_affiliation)

interchangeable\_counts[pair] = interchangeable\_counts.get(pair, 0) + 1

# Collect all unique affiliations from discussion\_df

unique\_affiliations = discussion\_df['affiliation'].unique()

# Generate a color map for the affiliations

num\_colors\_needed = len(unique\_affiliations)

if num\_colors\_needed <= 20:

colors = plt.cm.tab20(np.linspace(0, 1, num\_colors\_needed))

elif num\_colors\_needed <= 40:

colors = np.vstack((plt.cm.tab20(np.linspace(0, 1, 20)), plt.cm.tab20b(np.linspace(0, 1, 20))))

else:

colors = np.vstack((plt.cm.tab20(np.linspace(0, 1, 20)),

plt.cm.tab20b(np.linspace(0, 1, 20)),

plt.cm.tab20c(np.linspace(0, 1, num\_colors\_needed - 40))))

affiliation\_color\_map = {affiliation: mcolors.to\_hex(color) for affiliation, color in zip(unique\_affiliations, colors)}

# Add nodes for each unique affiliation (handling potential missing values)

for affiliation in pd.concat([dnet\_grouped\_sorted['contributor\_affiliation'], dnet\_grouped\_sorted['consumer\_affiliation']]).unique():

if not pd.isna(affiliation):

# Set default size and color

node\_size = 20

node\_color = affiliation\_color\_map.get(affiliation, 'lightblue')

net.add\_node(affiliation, label=affiliation, size=node\_size, color=node\_color)

added\_nodes[affiliation] = True

# Add edges to the network with the aggregated consumption values and adjusted width

for index, row in dnet\_grouped\_sorted.iterrows():

contributor\_affiliation = row['contributor\_affiliation']

consumer\_affiliation = row['consumer\_affiliation']

consumption = row['consumption']

# Add edge only if contributor and consumer affiliations are not NaN and weight meets threshold

if not pd.isna(consumer\_affiliation) and not pd.isna(contributor\_affiliation) and consumption >= min\_weight\_threshold:

# Ensure we add edges in both directions for interchangeable affiliations

edge\_key = (contributor\_affiliation, consumer\_affiliation)

if interchangeable\_counts.get(edge\_key, 0) > 1:

# Add edge between affiliations in both directions if not added already

if edge\_key not in added\_nodes:

net.add\_edge(contributor\_affiliation, consumer\_affiliation, weight=consumption, color='black', width=min(10, consumption \* 0.2))

added\_nodes[edge\_key] = True

if (consumer\_affiliation, contributor\_affiliation) not in added\_nodes:

net.add\_edge(consumer\_affiliation, contributor\_affiliation, weight=consumption, color='black', width=min(10, consumption \* 0.2))

added\_nodes[(consumer\_affiliation, contributor\_affiliation)] = True

else:

# Add only one edge if the affiliations are not interchangeable or there's only one occurrence

if edge\_key not in added\_nodes:

net.add\_edge(contributor\_affiliation, consumer\_affiliation, weight=consumption, color='black', width=min(10, consumption \* 0.2))

added\_nodes[edge\_key] = True

# Set the layout algorithm to force-directed

net.barnes\_hut(gravity=-8000, central\_gravity=0.3, spring\_length=200)

# Save the network to an HTML file with added label

net\_html = net.generate\_html()

# Create the HTML string for the affiliation colors

affiliation\_colors\_html = "<h2>Affiliation Colors</h2><table border='1'><tr><th>Affiliation</th><th>Color</th></tr>"

for affiliation, color in affiliation\_color\_map.items():

affiliation\_colors\_html += f"<tr><td>{affiliation}</td><td style='background-color:{color}; color:{color};'>{color}</td></tr>"

affiliation\_colors\_html += "</table>"

# Combine the network HTML with the affiliation colors HTML

full\_html = net\_html.replace("</body>", f"<h1 style='text-align:center;'>Discussions User Centric with Threshold value '3'</h1>{affiliation\_colors\_html}</body>")

# Save the combined HTML to a file

with open("discussions\_user\_method\_1\_three.html", "w") as f:

f.write(full\_html)