Algorithms for Coloring Transit Maps

Subway maps can be hard to read. We can make it easier to read by coloring the transit lines using colors that are as different as possible. This is done in two parts.

Part 1. Pick colors that are as different as possible.

Part 2. Color the transit lines using the chosen colors.

About Part 2

Here are two ways of doing Part 2.

Option 1. If two train lines intersect, then they should be colored as differently as possible.

Option 2. Maybe the subway system has many different lines with many different colors like in New York or Hong Kong. To make it easier to spot which region of the subway map to look at, we should color train lines that are in a similar part of the map by similar colors.

These are not the only options. What problems do you have with how transit lines are colored, and how can you address them?

About Part 1

There are at least two meanings for "*different colors*".

Option 1. The first is to choose colors that have high relative color contrast.

Option 2. You've probably used a color picker before. In the RGB model, a color is specified by how much red, green, or blue is used (from 0 to 255). But distances in the RGB model do not correspond to the perceptual distances of colors. See this. So to tell how close colors are together, we need something better than the RGB model. This is given by the CIECAM02 color model, an improvement on the 1976 CIELAB color model and currently the best (?) perceptually uniform color spaces, for which

- a point in this color space corresponds to a visible color, and
- the perceived difference between two colors corresponds to the distance between their corresponding points.

Here's one way to implement **Option 2** in Python. Suppose our subway map uses n colors. We then do the following steps.

1. Randomly select n points $x_1, x_2, \ldots, x_n$ in the sRGB space.
2. Use the Python package Colorspacious to convert these points to points $y_1, y_2, \ldots, y_n$ in the CIECAM02 space.
3. Compute the minimal pairwise distances between these points $y_1, y_2, \ldots, y_n$: $\min_{i \neq j} \left|\left| y_i - y_j \right|\right|$
4. Now repeat Steps 1 – 3 until we get points $y_1, y_2, \ldots, y_n$ that give a very big minimum in Step 3. Such points $y_1, y_2, \ldots, y_n$ are then far apart from each other, so their corresponding colors are perceptually "far apart" from each other too.

Note

Option 2 is a special case of:

The (Generalized) Tammes Problem. Given a number n and a set C, find n points in C that maximizes the minimal distances between any two of the n points. In other words, solve the following optimization problem:

$$\max_{\boldsymbol{y} \in C^n} \min_{i \neq j} ||y_i - y_j||$$

Here, the vertical bars denote taking the norm of a vector

$$||(x, y, z)|| = \sqrt{x^2 + y^2 + z^2}$$

Studying this problem may lead us to faster implementations of Part 1, Option 2.

Of course, this is one of many further directions you can go in. You can give suggestions on what you want to do.