MTH6412A: Commercial traveler project (phase 5)

Objective: application

The final part of the project is to put your TSP solving method into action to **reconstruct jagged images**. Download the **shredder.zip** file available on the course's Moodle site. This contains several directories:

- bin contains some Julia tools.
- images / original contains multiple original images in PNG format.
- images / shuffled contains the same images, shredded.

The "jagged" images were created by cutting each image into a number of vertical bands and reordering those bands in random order. The bands are of **constant** width.

The goal of this phase is to **reconstruct the images as best as possible**. This can be done using the TSP by imagining that each vertical band represents a node of the complete graph, and the weight of each edge is a measure of dissimilarity between two vertical bands.

The idea is to assume that in the original image, two adjacent bands are **very similar**, i.e., very few dissimilar. We thus seek a **simple path of maximum length through the nodes of the graph**, which is also of minimum weight. As we have seen in class, this problem is equivalent to TSP.

To formulate the problem as a TSP, a fictitious node (number zero) has been added as well as a

zero-weight edge connecting this fictitious node to each other node of the graph. By looking for a minimal tour on this modified graph, then by removing the zero node, we obtain the seeking path.

Julia Modules

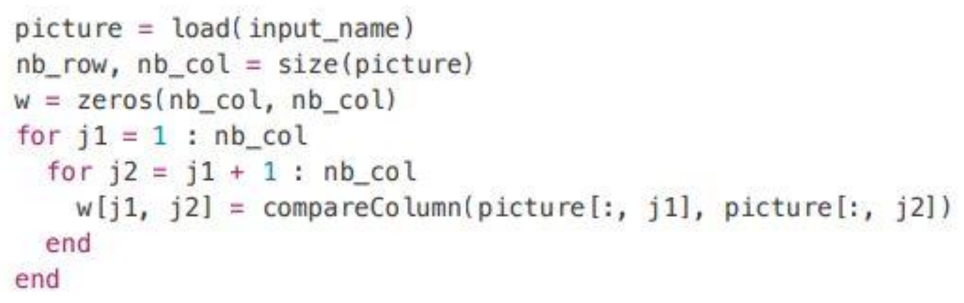
You will need to install the following Julia modules:

- FileIO
- Pictures
- ImageView
- ImageMagick

Method

For each shredded image, a tsp file has already been created for you in the tsp / instances directory. These tsp files should be read with your updated version of read_stsp.jl (which reads edge weights). The data field of the nodes will simply be the index of the node.

Alternatively, you can read a jagged image yourself and get the dissimilarity between each pair of columns using the commands:



```
picture = load( input_name)
nb_row, nb_col = size(picture)
w = zeros(nb_col, nb_col)
for j1 = 1 : nb_col
  for j2 = j1 + 1 : nb_col
    w[j1, j2] = compareColumn(picture[:, j1], picture[:, j2])
  end
end
```

Once a route has been identified, build a list of nodes along that route without removing node zero. Using the write_tour () function, create a .tour file in TSPLib format that describes your tour.

Sample .tour files are available in the tsp / tours directory; these were identified by a method of solving the TSP, but do not necessarily give an optimal solution.

Given a .tour file, we obtain the corresponding reconstructed image using the reconstruct_picture () function. This function assumes that the route given by the .tour file starts at node 0.

It's up to you to create your .tour files using your RSL and HK implementation and generate the corresponding reconstructed images. Play around with all the settings you want to try and get the best possible reconstruction.

For each reconstructed image, give the length of the best tour found, the original image as well as the reconstructed image side-by-side.

References

The code used in this part of the project is from **https://github.com/robinhouston/imageunshredding**.

I suggest you browse this website for a little more information. The files under tsp / tours were generated by the LKH library, an implementation of Lin and Kernighan's heuristic described at **http://webhotel4.ruc.dk/~keld/research/LKH**.

Guidelines

Write readable, ventilated, documented, and commented code. You can refer to the guidelines for writing Julia code: https://docs.julialang.org/en/v1/manual/style-guide

Your methods should be documented following the scheme given in Julia's official documentation: https://docs.julialang.org/en/v1/manual/documentation

Submit your report as a Pluto notebook in PDF format on Moodle and record the julia jl version on the phaseX branch of your fork. Markdown cells should guide the reader through your report.