EE2703: Assignment 5

Keyboard Optimization

EE23B092

1 Implementation

The code makes use of the libraries - numpy, random, matplotlib. The submission directory includes qwerty_layout.py along with the code file (ee23b092.py) and this report.

The entire code in condensed into a single function named optimization() which takes a string of text as a parameter (named text). This function contains 4 functions in its body - calculate_distance(), optimize_layout(), plot_cost_reduction() and visualize_keyboard().

1.1 calculate_distance(text, keys, characters)

This function takes in text given by the user and calculates the total distance travelled by the finger(s). This is similar to the function implemented in the previous assignment. keys and characters are key parameters which by default are set to the keys dictionary and characters dictionary of qwerty_layout.py file.

To calculate the distance, the function goes through the text on a character-by-character basis and stores the distance the finger takes in total for each character and sums it all up to give the final output.

For each character in the text, it checks all the characters required to input it from the character dictionary. It accesses the position and the start character from each of these sub-characters. It calculates the euclidean distance between the position of each of the sub-character and the position of each of the start character, and stores this distance in the dictionary.

Final output is the sum of the values of the dictionary.

1.2 optimize_layout(text, keys, characters, iterations=1000, temperature=1000, cooling_rate=0.9)

This function performs the simulated annealing and returns the best_layout and the distances over the iterations (which is subsequently used for plotting). Initialize the best_layout and best_distance to the current values w.r.t. this layout. The default parameters are iterations, temperature and cooling_rate. These values have been set through trial-and-error.

During each iteration, two random keys are chosen to perform the swap. We are only concerned with the keys dictionary whilst swapping and optimizing. To perform the swap, along with value dictionary swapping, we also need to change the start character as well if one of the chosen keys include a home row key.

We accept the swap in two instances - when the new distance is better than the old one or if it is not better, the swap is accepted with some probability (part of simulated annealing). Else, we undo the swap. Then, the temperature gets updated and the distances are included. The final output returned by the function consists of the best_layout (dict) and costs of distances (list).

1.3 plot_cost_reduction(costs)

This function takes in the costs of distances over all iterations and plots it. The plot shows Total Distance vs. Iterations.

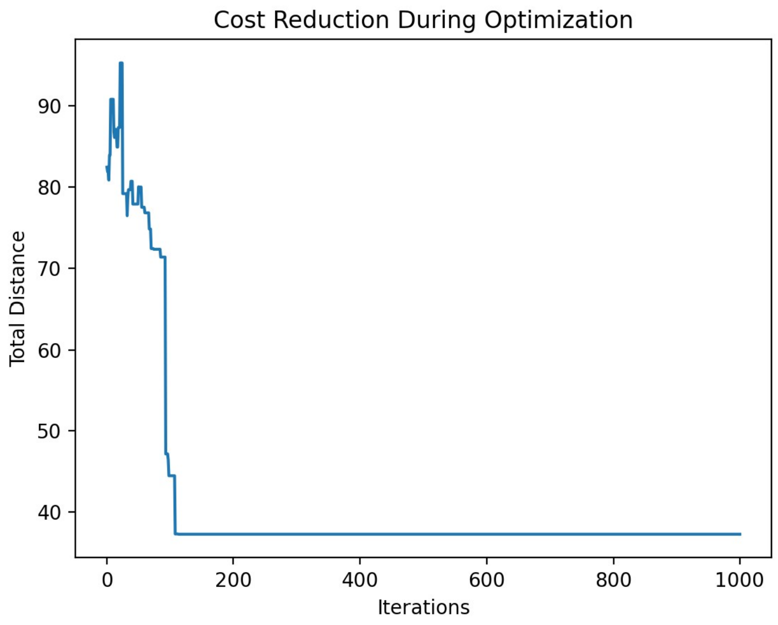
1.4 visualise_keyboard(keys)

This function takes in the layout and visualizes the keyboard. All the keys are square in shape (to avoid potential overlaps). The code is similar to the code in the previous assignment.

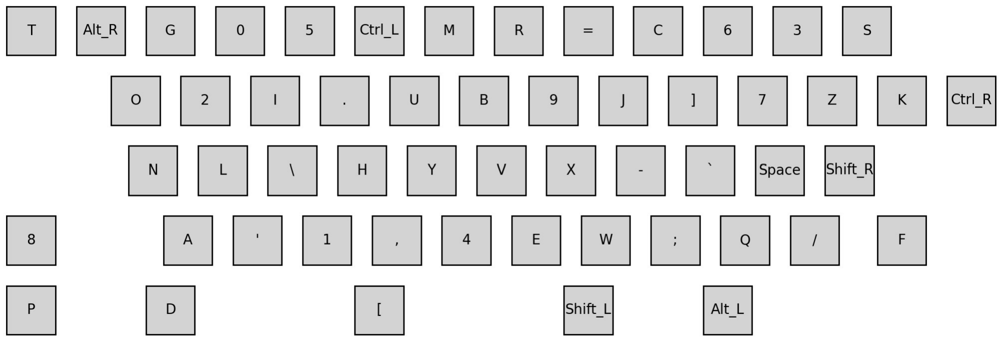
2 Sample Input and Output

The sample input text is a multi-line string:- Hey! how have you been? I have been well.

2.1 Total Distance vs. Iterations



2.2 Visualizing the Keyboard


T Alt_R G 0 5 Ctrl_L M R = C 6 3 S
O 2 I . U B 9 J] 7 Z K Ctrl_R
N L \ H Y V X - ` Space Shift_R
8 A ' 1 , 4 E W ; Q / F
P D [Shift_L Alt_L