APL - Assignment 5 (Keyboard Optimisation)

Nishanth Senthil Kumar (EE23B049)

October 2024

1 How to run the code

- The submitted code is a Juypter notebook, so please run all the cells sequentially, if it behaves unexpectedly, start again from the beginning (topmost code cell).
- A "layout.py" python file is present in the zip file. Please make sure that it is present in the same directory as the notebook.
- If a custom input file is given, please make sure it is named "layout.py" and has the 2 dictionaries which are named "keys" and "characters".
- To change the input text, modify the text in the "input_string" variable and run the code again.
- Refer to the instructions given at the top of the notebook before running the code.
- Please wait 30-45 seconds for the animation to render.
- I have also included a "colemak.py" file for an alternative layout file, it may be used in place of "layout.py" if required.

2 Assumptions/Conventions followed

- The distance calculations is as per the convention given by sir in programming quiz 4.
- For this assignment, I have decided to ignore the space character and not consider the frequency of using the space bar. The space key is the most common key used in most languages, and including it in the optimisation makes the keyboard optimise only for the space bar, and the output sacrifices some other less frequent keys. Hence, I have decided to ignore them.
- I have kept the number of iterations to 80, as the animation takes too long to render for larger values of number of iterations. If you want accuracy to increase, increase the number of iterations and ignore the animation.
- An animation overflow warning might pop up, please ignore it, i made the number of iterations low enough for most of the graph to come in the animation.
- I decided to not animate the keys of the keyboard switching as it was taking too long for the animation to render.

3 Approach

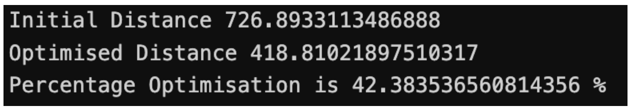
- Drawing of the keyboard is done by using by Assignment 4 code.
- The optimisation is done with the help of simulated annealing. This is similar to the famous travelling salesman problem, where instead of ”new routes”, we have ”new layouts” and we try to minimise the distance travelled by the finger.
- Instead of shuffling cities and generating new routes, here we have to swap keys and generate new layouts to minimise from.
- I maintain a home keys list, which has all the home keys of a layout. I generate 2 keys (not including space), and swap them in the input dictionary. I also check with the home keys list, if a key is present in the list, then this key is deleted from the list and the other key is added to the list, and every occurrence of the first key is replaced by the second key in the ”keys” dictionary.
- Like this, the dictionary is manipulated, and the same code used in assignment 4 can be used for distance calculations and plotting.
- The simulated annealing algorithm was taken from sirs solution of Travelling salesman problem, which works for this problem with some minor modifications.
- I implemented the animation as well, which was given in the demo given by sir. Since the time taken to generate the animation even for 100 iterations became large, I removed the animation of the keyboard and only animate the minimum distance vs iterations plot.

4 Observations

- The increase in number of iterations increase the accuracy usually, this is because the number of different layouts that are tried out increases.
- Decreasing the cooling rate(by a small amount) along with iterations increases accuracy slightly. To ensure that the animation covers most of the graph, I have set the cooling rate to 0.985 and the number of iterations to 80.

5 Sample Output

- This is some random Whatsapp forward. The input layout is QWERTY.
  input_string= ”’COULDN’T PRONOUNCE THE NAME EITHER?Don’t sweat it-what matters is having an AMAZING time with us! We’re SUPER EXCITED to invite you to SHAASTRA’S SOCIAL CAMPAIGN LAUNCH on SUNDAY, 29TH SEPTEMBER, at 6:30 PM, happening at the HFC LAWNS!Get ready for an evening packed with FUN ACTIVITIES, GAMES, and a chance to be part of something BIG! Whether you’re coming for the FUN or to support the CAUSE, it’s going to be a NIGHT TO REMEMBER!Grab your friends, and LET’S MAKE IT EPIC! See you there!”’



```
Initial Distance 726.8933113486888
Optimised Distance 418.81021897510317
Percentage Optimisation is 42.383536560814356 %
```

Figure 1: The initial distance and the optimisation

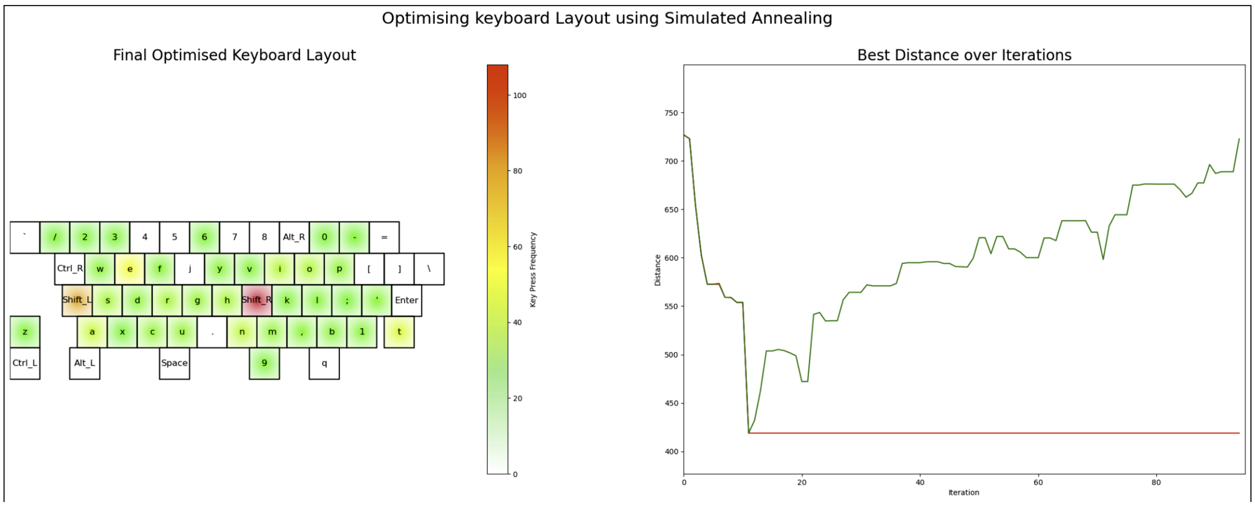
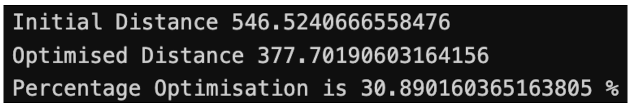


Figure 2: The graphs and keyboard

- For the same input, using Colemak keyboard.



```
Initial Distance 546.5240666558476
Optimised Distance 377.70190603164156
Percentage Optimisation is 30.890160365163805 %
```

Figure 3: The initial distance and the optimisation

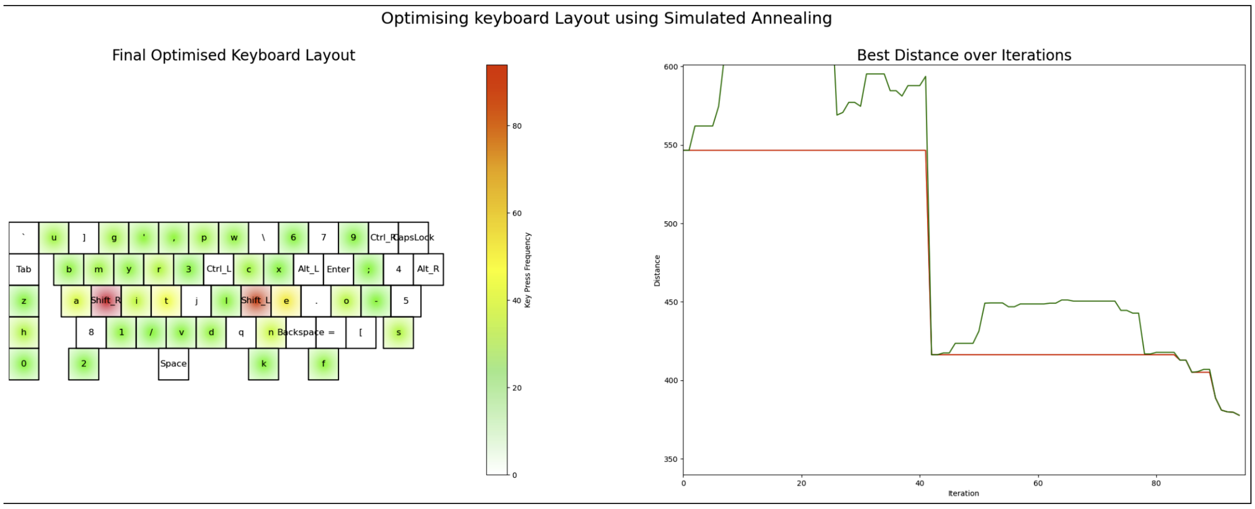


Figure 4: The graphs and keyboard

- As we can see, the initial distance for Colemak is much less than Qwerty.
- On an average, with enough iterations, They will settle at the same optimal keyboard.
- I thank Deepak Charan S (EE23B022) for the Colemak layout.