Assignment-5 : Optimising Keyboard Layout

Deenabandhan N
EE23B021

Assumptions

- For calculating distances , I have assumed that it is unidirectional.
- While optimising one of the major confusions that I had , was whether to swap the special keys i.e {Enter,Alt_R etc..,}.
- I have assumed that the swapping of special keys is possible as the distance cost of **Shift_L** is less than that of number **1** and swapping these decreases the travelling distance cost if the string is just numbers.
- Another confusion regarding swapping the shift key is having both shift keys on same side will violate touch typing as the layout assumes that left sided keys are mapped to Shift_R and vice versa.
- I have assumed that even having both shift keys on one side is not an issue.
- The reason for my assumption is I have considered pressing a capital letter as pressing **Shift** key and the corresponding letter individually.
- As I have observed that in large texts Space key always gets the maximum frequency which will affect our optimisation as changing space key itself will decrease the cost by significant amount.
- So to avoid that,I discarded the space key in the given string.

How to run the ipynb file

- To run the submitted file , you need to import the layout and string to be used.
- If you need to display the animated keyboard layout that is changing while optimising you have to use the following code snippet.

```
anim=main(clt,st,condt=True)
```

  where st is the input string and clt is the layout to be used in the code.
- If you need to display just the picture of the final optimised layout generated you have to use the following code snippet.

```
anim=main(clt,st,condt=False)
```

- It takes so long for the code to generate animation for keyboard layout whose reason is discussed below.
- You can also change the number of iterations/cooling rate by changing the corresponding variables in the main() module.

Approach

- The assignment is just the extended version of previous one with just two additional modules.
- I have used the same approach as discussed in the previous assignment such as using patch for keyboard having a 2-D array to generate a gaussian heatmap etc..,
- This assignment additionally involves .
  - Using simulated annealing
  - Optimising the previous code to avoid time complexity

Using Simulated Annealing:

- We will run the following algorithm **num_iter** times.
  - Swap two positions of keys randomly : Performed by
    `new_layout()`
    module.
  - Generate a random probability and check if it is less than the
    `np.exp((current_out- new_out) / temp)`
    where temp is reduced by the cooling rate.
  - If less explore i.e move out of the minima else just stick with the local/global minima.

Optimizing the previous code

- On running the previous code , I found that it is highly un optimised so I changed the way in which the arguments are passed.
  - There is a new variable called **param** returned by *init()*.
  - This parameter will return all the dictionaries and travel cost of the string as a list.
  - There is no explicit cost function defined in the code as , I have changed the travelling cost in the new_layout() module itself by subtracting and adding the frequency*(cost_initial/cost_final) of the swapped key.
  - The above functionality is done only to decrease the time complexity.
- I have also changed the dimensions the 2-D array that I have used to generate heatmap as 1000*1000 array resulted in using higher computations.
- Decreasing this resulted in a blurred heatmap.

Insights :

- I have observed that higher the number of iterations , higher the chance of achieving the optimal solution and also making it very high will cause overflow error as exp can't handle such big values.
- Choosing learning rate also matters , as $(0.995)\hat{}100 = 0.$ so there is still a possibility for the algorithm to choose a bad value and explore where as $(0.99)\hat{}100=0.3$ implying that prbability decreases.
- The swapping of shift keys causes a lot of confusion as we don't know which shift key is closer to which letter.
- I guess choosing the position of shift key itself involves another optimisation as each key will have different positions after swapping.

Trying different layouts

- I have used the home keys as text to check the optimisation with number of iterations as 500.

QWERTY :

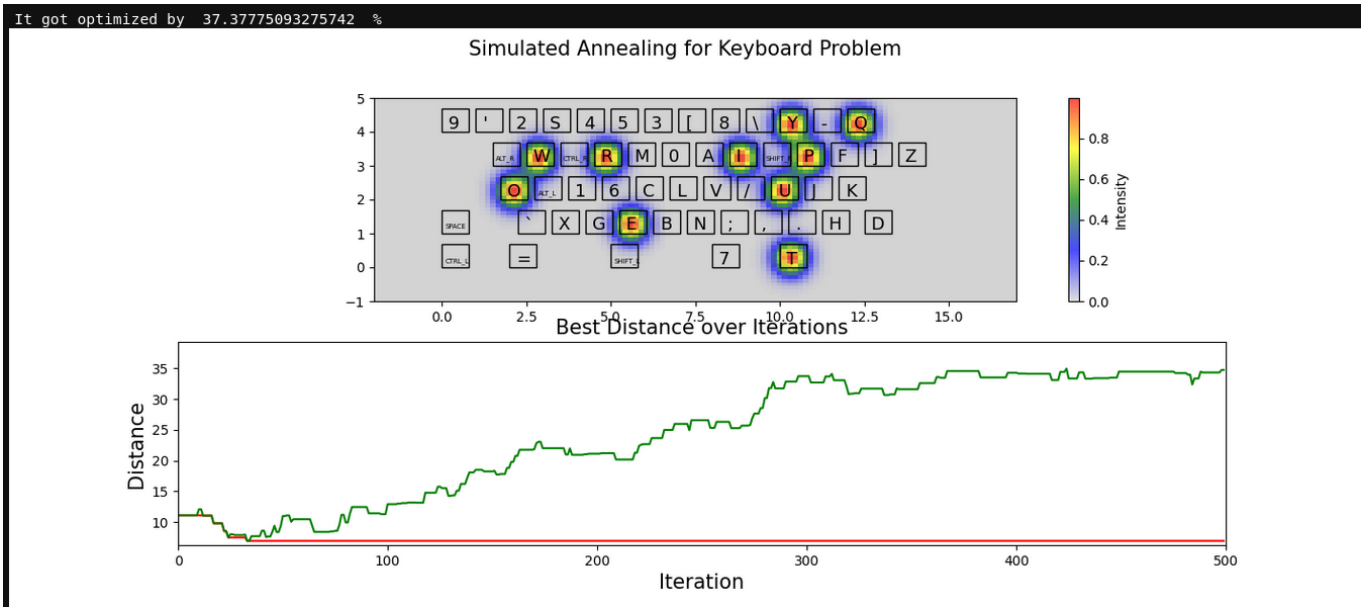

Figure 1: Qwerty_Layout

COLEMAK :

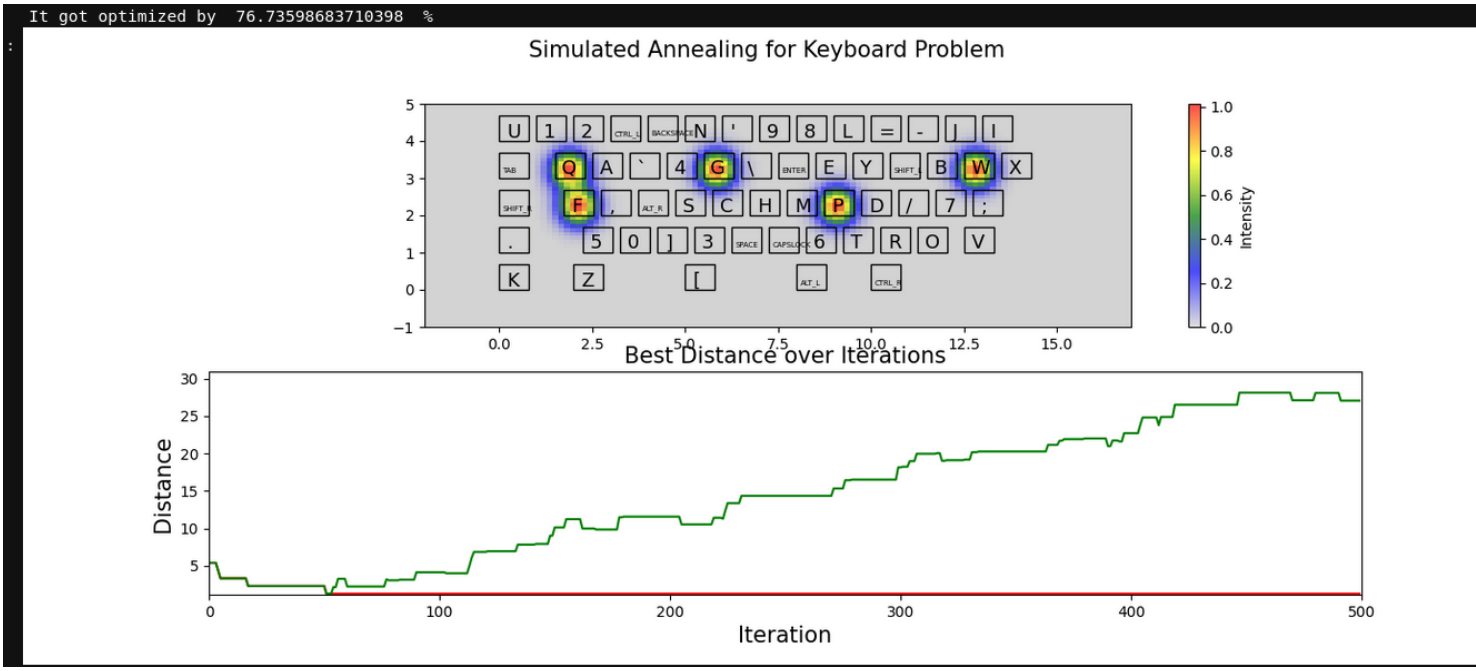

Figure 2: Colemak_layout

DVORAK :

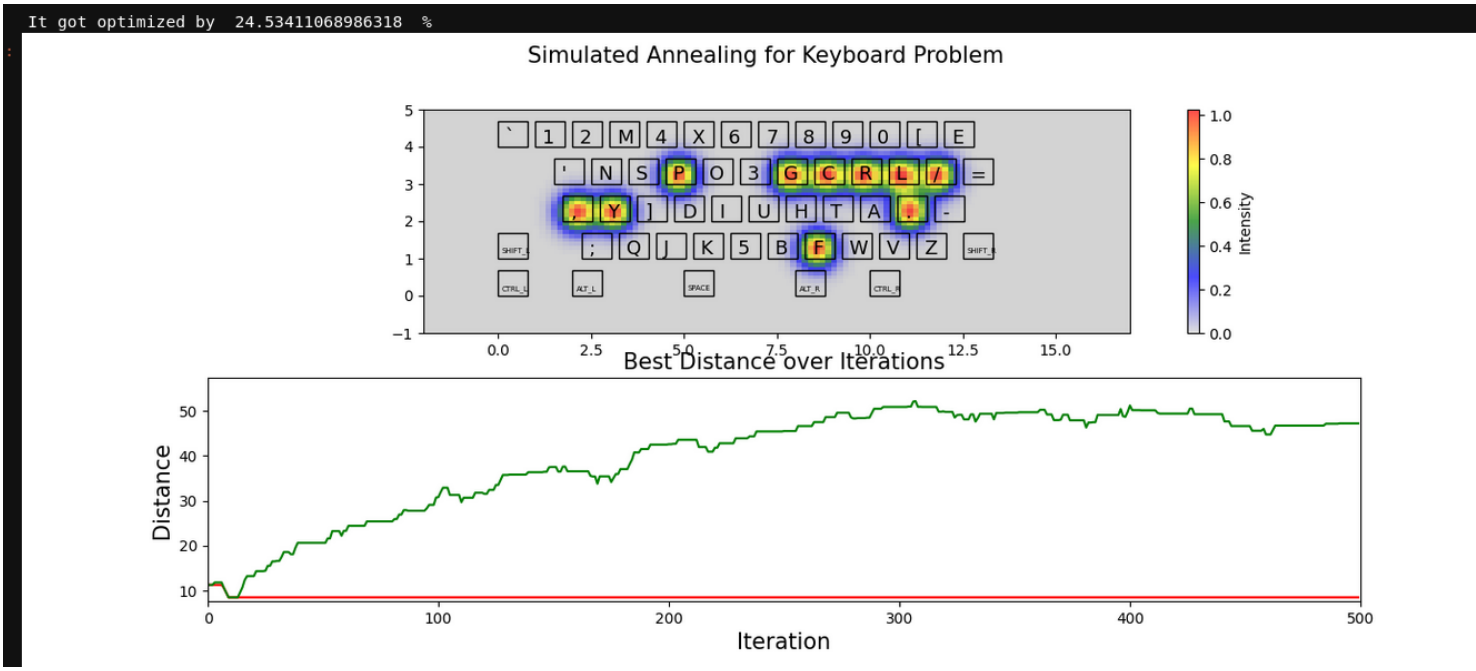

Figure 3: Dvorak_layout

- I have inferred that the Colemak is the most optimized one so Simulated annealing hardly does anything.
- The video for optimizing the string of question statement with **QWERTY** is here.

References :

- I have referred to matplotlib documentations.
- I have referred to Counter module.
- I would like to thank Mayank Sharma EE23B045 Karthik Kashyap - EE23B030 for rectifying the higher complexity cases.