Keyboard Optimization

Keshav Saravanan, EE23B035

October 2024

In this assignment, I optimize a keyboard layout for an input string, to minimize the travel distance.

1 Running the Project

To run the project, place the python files in the same directory. There are two ways to run the project - using the python script, or in the Jupyter notebook. The python script saves the optimized heat map as an image file. The Jupyter notebook contains an animation of the heat map being optimized as the simulated annealing is being performed (It does not output the heatmap however).

To run the python script, run `python ee23b035_main.py`. There are optional command line flags for this: `-f` to specify the file to read the input text from, `-o` to specify the output image file, and `-i` to ignore spaces in the input text (since spaces occur very frequently, they tend to reduce contrast between other letters). The heatmap is saved as a separate image file.

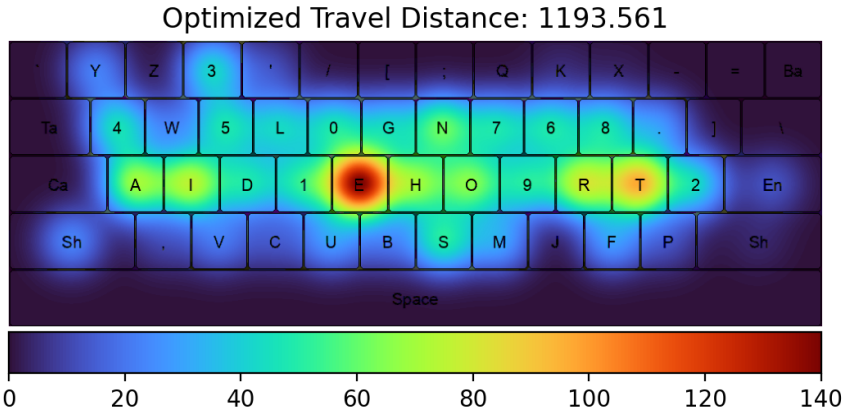

Figure 1: Optimized layout, starting from the QWERTY layout

```
[pythonaddike@pythonaddike-vivobook kb_opt]$ python ee23b035_main.py -i
Unoptimized Travel Distance 1824.844
Optimized Travel Distance: 1368.982
Key Press Frequency:
' 12    , 7     - 0     . 25    / 0     0 42    1 42    2 42    3 42    4 42
5 42    6 42    7 42    8 42    9 42    ; 0     = 0     [ 0     \ 0     ] 0
` 0     a 70    b 15    c 10    d 45    e 136   f 31    g 36    h 61    i 72
j 0     k 4     l 39    m 29    n 57    o 60    p 11    q 1     r 76    s 50
t 91    u 30    v 20    w 17    x 2     y 23    z 0
Backspace: 0
CapsLock: 0
Enter: 13
Shift_L: 18
Shift_R: 10
Space: 0
Tab: 0
```

Figure 2: Key press frequency, and optimized travel distance

To run the Jupyter notebook (which contains the simulated annealing animation), specify whether or not to ignore spaces. The input text is also specified as a multi-line string. The heat map animation is displayed in the Jupyter notebook. You can also set the initial temperature for the annealing.

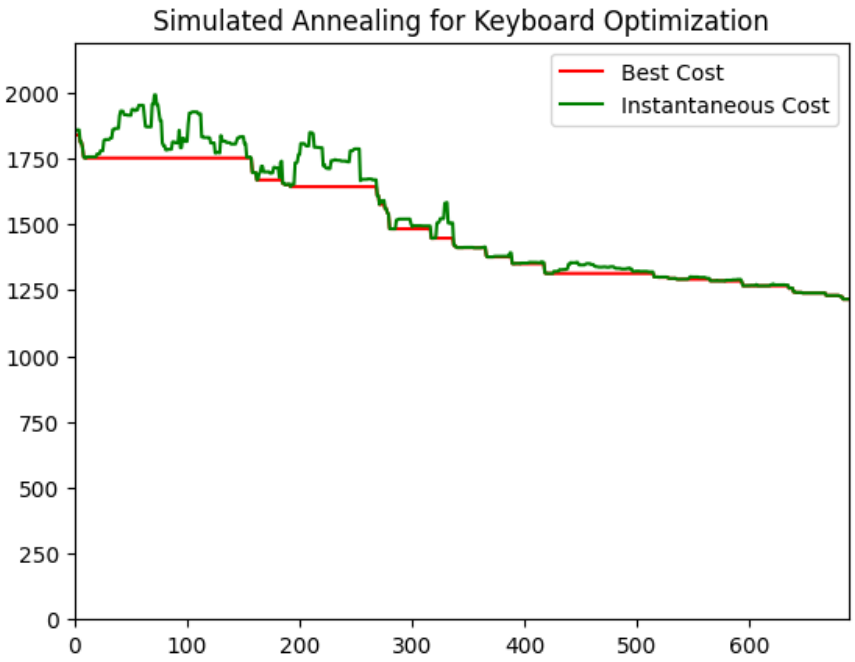

Figure 3: Caption

2 Configuration

This simulation is patched onto my submission for the previous assignment - Keyboard Analysis, and so, carries over all the original configuration options. Additionally, I use the old keyboard layout for this assignment, as I had used the same for the previous assignment as well. The starting layout can be chosen in `ee23b035_config.py`.

The script also has some extra parameters in `ee23b035_config.py`, related to the annealing:

1. `TEMP` - The initial temperature for annealing
2. `ALPHA` - The decrement factor
3. `T_MIN` - The temperature at which to stop annealing

3 Approach

For the simulated annealing, my cost function is just the travel distance. Over each iteration, the temperature is reduced to `ALPHA` times, until it reaches `T_MIN`. This results in the probability of choosing a worse solution decreasing as time goes by.

The `simulated_annealing` function is a generator - In each iteration of the annealing, it randomly swaps two keys, and computes the cost function. If the cost decreases - the swap is automatically chosen. If not, it is chosen with an exponentially decreasing probability. If the swap is not chosen, then the layout is swapped back to the original layout.

The keyboard is then drawn with the optimized layout and the heatmap is generated. In the case of the Jupyter Notebook, the heatmap is not drawn, but each iteration of the annealing is drawn in an animation.

4 Observations

In the graph of the instantaneous cost vs iteration, it can be seen that towards the right end of the x-axis, the instantaneous cost jumps around a lot less. This is indicative of the exponentially decreasing probability - further and further down, the probability of picking a worse solution is less.

Also, with a string with less than 8 unique characters, the simulated annealing should result in a very small travel distance (ideally, zero). This is because all 8 characters could be placed in the home row positions and therefore no travel distance is present. This is true in my solution. For the input string `"zxiu320'"`, the output is -

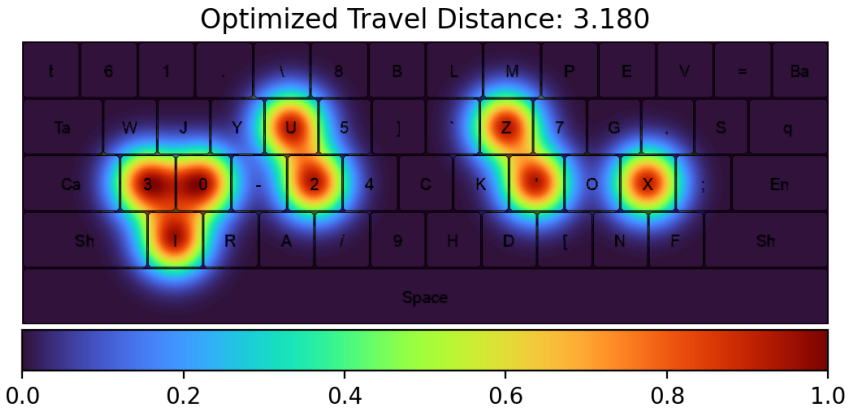

Figure 4: Optimized layout for short text

Which is quite close to the ideal layout.