Keyboard Optimization EE24B141-A3

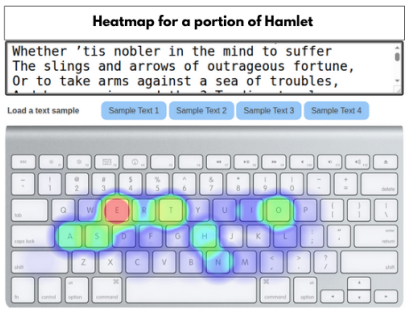
Aim: Finding the optimal keyboard layout for a given text
Files: kbd_optim.py

Instructions to run: *python3 kbd_optim.py -f shakespeare.txt -iters 1000 -t 1 -alpha 0.999 -n 5 -epoch 1*

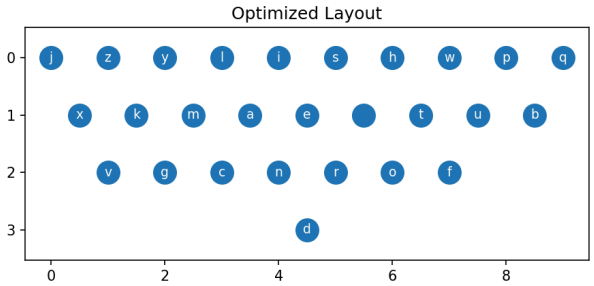
The file accepts upto 6 command line arguments which are specified below. If any values are not given then the program will run on appropriate default values. A command with example values filled out is given below.

Argument	Value	Default
-f	Text file	"the quick brown fox jumps over the lazy dog" "APL is the best course ever"
-iters	No.of iterations	50000
-t	Initial temperature	1
-alpha	Alpha decay	0.999
-epoch	Epoch value	1
-n	No.of characters swapped per trial	5

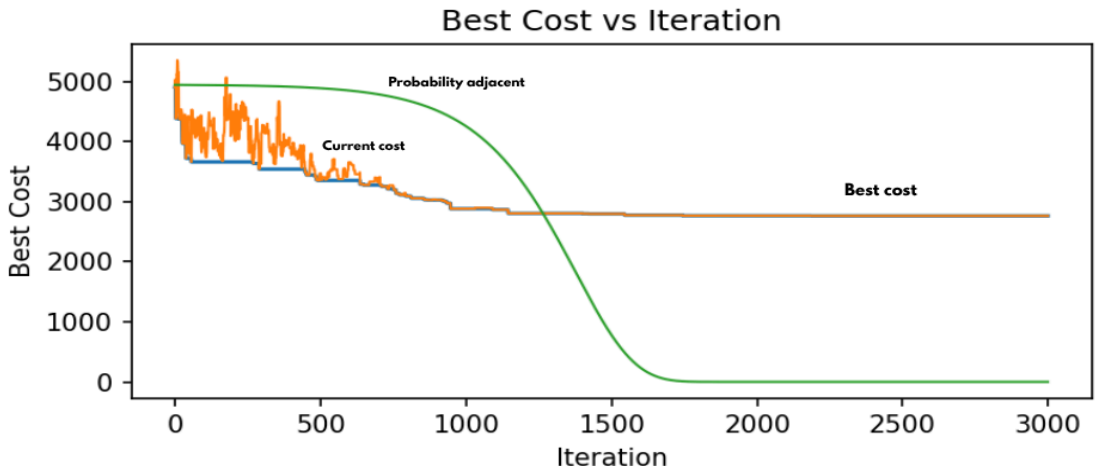
Results: (All results are optimised for shakespeare's hamlet)



The heatmap for the example text shows that "a, e, t, s, o, h" are commonly used. Running the model with default parameters we get the layout on the right where the most common keys are all clustered near the center (spacebar included).



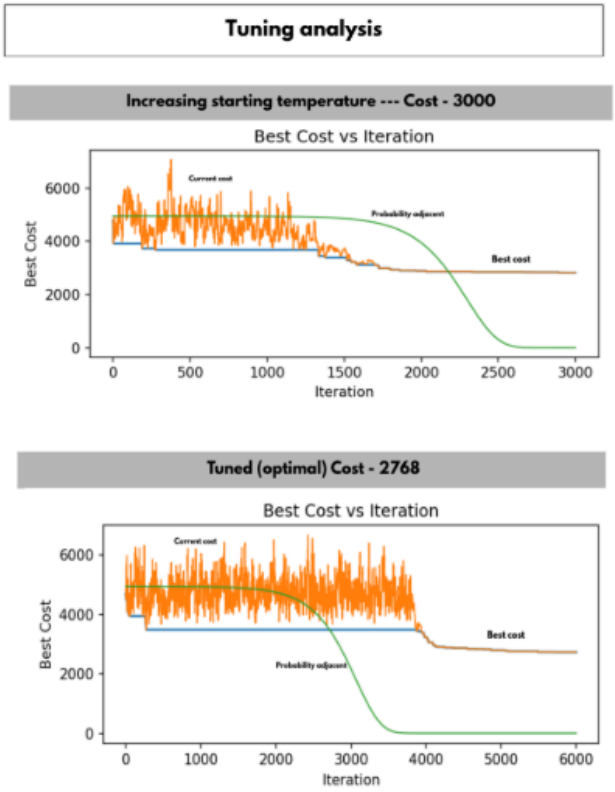
The cost was brought to half over 3000 iterations and a look at the graph for best costs over iterations shows us how this optimisation happens - rapidly initially and slower later. The probability adjacent line is a scaled plot of e^(-1/T) for the system and as it decreases, the system's tendency to try with solutions that are worse than the current one decreases.



Tuning:

Without tuning the other parameters appropriately, increasing the temperature gives increasingly worse best costs.

Generally, the probability term dropping to 0.5 before half the iterations tends to give best results. This can be achieved by increasing iterations, duration of epoch or decreasing alpha. The best results were found while varying duration of epoch rather than changing the others



Conclusion:

For optimisation problems, always taking the greedy approach can lead to getting stuck in locally optimal solutions which might not be the global optimum. Randomly picking solutions that have worse cost than the current one, in the hopes of getting to the global optimum won't lead to a final answer as it will continue picking random values even once it reaches the end of the search period.

Annealing deals with this with a hybrid approach: exploring unlikely routes initially and then transitioning to the greedy method as iterations continue to ensure the optimum solution is reached. The best results are obtained when the tendency to explore worse solutions drops off before half-way through the iterations after which the greedy approach takes over to optimise.