Keyboard_Layout_optimization

October 13, 2024

1 Keyboard Layout Optimization Using Simulated Annealing - EE23B060

1.1 Overview

This program implements a keyboard optimization algorithm to minimize finger travel distance while typing. It utilizes the QWERTY keyboard layout and uses simulated annealing to optimize the arrangement of keys based on the frequency of character usage in the provided input text.

1.2 Key Components of the Code

1.2.1 1. Keyboard Layout and Character Mapping

The code defines two primary dictionaries: - **`qwerty_keys`**: Contains key names and their positions on the keyboard, as well as their corresponding starting characters for typing. - **`qwerty_characters`**: Maps characters (including uppercase letters, numbers, and symbols) to their respective key sequences.

1.2.2 2. Character Frequency Counting

The `letter_count` function calculates the frequency of each character in the input text, including any necessary modifier keys (like Shift).

1.2.3 3. Distance Calculation

The `distance` function computes the Euclidean distance between two keys on the keyboard. This distance is used to calculate the total travel distance for typing characters using the `calc_distance` function.

1.2.4 4. Layout Optimization

The `optimize_layout` function employs simulated annealing to optimize the keyboard layout. It randomly swaps keys and evaluates the impact on typing distance, gradually reducing the temperature of the annealing process to find a more optimal arrangement.

1.2.5 5. Visualization

The `visualize_keyboard` function creates a heatmap visualization of the keyboard layout, showing the frequency of key usage in the input text. The keys are colored based on their usage frequency.

1.2.6 6. Analysis and Optimization

The `analyze_and_optimize` function demonstrates the analysis of the input text, visualizes the original layout, optimizes it, and then visualizes the optimized layout.

1.3 Simulated annealing

Simulated annealing is a probabilistic optimization algorithm inspired by the annealing process in metallurgy, where controlled heating and cooling of materials lead to a stable structure. The core idea behind simulated annealing in our context of keyboard layout optimization is to start with an initial configuration of the keys (here QWERTY) and iteratively make small changes (or swaps) to this configuration.

1.3.1 Key Steps of the Algorithm:

1. **Initial Configuration**: Begin with a standard layout (here, QWERTY).
2. **Temperature Parameter**: A "temperature" parameter is introduced to control the probability of accepting worse solutions.
3. **Neighbor Selection**: Randomly select two keys to swap, creating a new layout.
4. **Cost Function**: Calculate the total finger travel distance for the current layout and the new layout.
5. **Acceptance Criteria**:
   - If the new layout has a lower cost (less distance), it is accepted.
   - If the new layout has a higher cost, it may still be accepted with a probability that decreases as the temperature lowers. This allows the algorithm to escape local minima.
6. **Cooling Schedule**: Gradually decrease the temperature to reduce the acceptance of worse solutions over time.
7. **Termination**: The algorithm continues until a stopping criterion is met, such as a maximum number of iterations or when the temperature is sufficiently low.

1.4 Key Design Choices in my code

In this implementation, the positions of certain keys (Shift, Space, Alt, and Control) are fixed and do not change during optimization.

1.5 How to Use the Notebook

1.5.1 Changing the Input String

1. In the second block of code, change the `input_text` and run it to analyse the keyboard.

1.6 Conclusion

The simulated annealing algorithm effectively optimizes the keyboard layout by minimizing finger travel distance.

1.7 References

1. https://en.wikipedia.org/wiki/Simulated_annealing
2. https://www.geeksforgeeks.org/simulated-annealing/