

Most keyboard optimization studies optimize for typing speed but don't consider ergonomics or typing with one thumb. A keyboard that is optimized for one-thumb typing and ergonomics would benefit all mobile users and make mobile keyboards more accessible. This paper optimized the mobile keyboard layout for one-thumb typing and ergonomics using a modified version of the MKLOGA genetic algorithm pipeline introduced in the Keyboard Layout Optimization and Adaptation paper by Keren Nivasch and Amos Azaria. The evaluation part of the MKLOGA pipeline was modified to incorporate one-thumb typing and ergonomics into its evaluation scoring of the keyboards. Our results show that our algorithm beat the QWERTY keyboard's evaluated performance score by 36.5% on average, reducing the total distance traveled by 26%, and increasing the comfort score by 4.5%. Our studies suggest the QWERTY keyboard is not the most efficient nor comfortable keyboard and keyboard layouts can be further optimized for those desirable characteristics. The code can be found in the GitHub link here: [https://github.com/SyndicateGit/Mobile\\_Keyboard\\_Optimization\\_Genetic\\_Algorithm](https://github.com/SyndicateGit/Mobile_Keyboard_Optimization_Genetic_Algorithm).

## I. Introduction

The main objective of this paper is to design a mobile keyboard layout that is optimal in both its typing speed and comfortability, to be used for accessibility purposes such as enabling individuals with one functioning hand to perform keyboard typing tasks efficiently. This is to be of great value for the growing population of people who are limited to one functioning hand [7]. With such a keyboard layout, these individuals will no longer have barriers preventing their full utilization of mobile keyboards, which are becoming more widely needed over time [8].

Agent-Based Modeling and Simulation (ABMS) is a technique used for simulating the interactions of agents within a given environment [6]. It models complex systems where individual components, or 'agents', operate based on a set of rules, allowing us to observe emergent behaviors from these interactions [4]. In the case of this work, the agents being observed are different keyboard layouts and their environment is a performance evaluation pipeline that uses typing speed and thumb comfortability as metrics to evaluate the *performance score* of each keyboard layout. Agents interact with the performance evaluation

pipeline through their performance metrics.

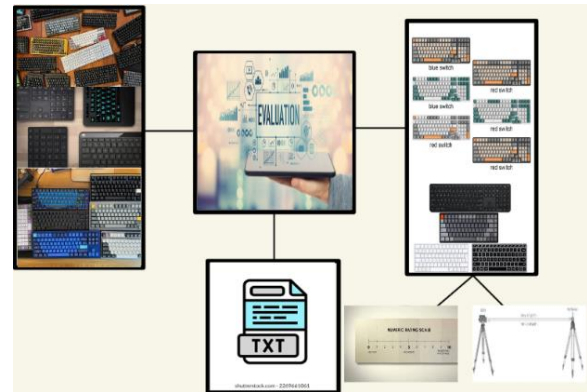


Figure 1 - Basic image of the program

(as shown in Figure 1). As the keyboards (agents) are tested within the simulated environment, data about typing speed and comfort are collected. Poor-performing keyboards (agents with lower overall performance scores) are removed from the pool, mimicking a natural selection process[9]. This ensures that only the most efficient and comfortable keyboard layouts (agents) are retained to undergo a key-swap algorithm to generate the next generation of keyboards[9].

The characteristics of these agents include the mapping of keys to predefined coordinates. This mapping is crucial for evaluating the distance each thumb travels during typing and the associated comfort score, as explored in [9]. The layout of keys and their positioning relative to

common typing patterns are aspects that would affect the agent's performance.

The performance evaluation pipeline is a genetic algorithm; a type of search algorithm that mimics natural selection to find the best or sufficiently good solutions among various options [5]. The genetic algorithm improves keyboard layout performance through three key processes: Selection, selecting the best layouts from the existing pool; Crossover, combining features of preferred layouts to create new ones; and Mutation, adjusting features like key positions to enhance performance. Through such a process, ABMS can evaluate and evolve keyboard layouts within a simulated environment, continuously eliminating layouts with poor performance while retaining and evolving those with better performance [9]

## II. Background & Related Work

QWERTY keyboard layout was designed almost 150 years ago but remains the predominant layout used today. The QWERTY keyboard layout was designed to avoid typebar jamming in typewriters by separating commonly used letters across two sides, not for ergonomics.

Research done by students from Misericordia University, Dallas, PA [7] found that the Dvorak Right-Handed layout outperforms the QWERTY layout for one-handed typing.

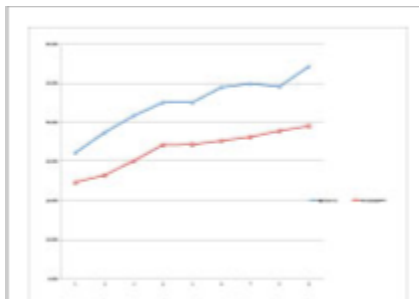


Figure 2 - Trial Data

As shown in Figure 2, In nine trials, participants using the Dvorak layout demonstrated an improvement in typing speed, reaching up to 54% of their two-handed speed. In contrast, those using the Reverse QWERTY layout (simulated QWERTY) showed lower performance at 39.1% of their two-handed speed. These results suggest that alternative layouts designed for one-handed typists, like Dvorak, can provide substantial advantages in typing speed and efficiency compared to traditional QWERTY keyboards.

Yin and Su introduced genetic algorithms as an approach for keyboard optimization [8]. Genetic algorithms consist of repeating three iterative stages of Selection, Crossover and Mutation that mimic natural selection in selecting for traits that benefit an organism's survival. The same three iterative processes were applied to optimize a population of candidate keyboard layouts. Fitness was measured by the sum of distance between every single character in the training corpus (collection of texts) to the nearest finger resting position. Their selection process consists of picking the top 10 performing keyboards to be the parent layouts which contribute to generating the next generation of candidate keyboard layouts.



Figure 3 – Key Swap Algorithm

During crossover, a subset of key locations would be copied from one parent to the child and the rest filled in by the second parent. (See Figure 3) A random amount between 0 to 5

mutations is applied to each child layout where two keys are swapped at random to help the model escape local optima. Yin and Su focused on optimizing distance traveled for computer keyboards for two or one hand typing speed but not for ergonomics. Mobile keyboards have limited space and rely on one thumb typing.

While distance evaluation based on the default resting finger position is accurate for computer keyboard typing, for mobile users a more accurate representation for typing would involve moving sequentially from one key to the next without returning to a resting position.

QWERTY keyboards have the keys spread out across two hands in width which is ergonomic for two handed typing but not for mobile keyboards. The ideal mobile keyboard manipulating space would be different from that of the standard keyboard.

Ana Beatriz Herthel and Anand Subramanian focused on single-finger keyboard optimization using a free form layout displayed as a grid. The grid layout with center thumb resetting position is more optimal than QWERTY layout since the average distance between each key and the center is less than the wide horizontal distance across a QWERTY keyboard [9]. However, their model's performance evaluation is based on typing distance alone. Although keys are closer on average, keys that are out of reach or too close to the bottom of the screen require the thumb to be stretched or contracted to an uncomfortable position. In addition, it assumes the thumb always returns to a center resting position rather than sequential typing that would be more representative of typing on a mobile keyboard.

Le Xiong ergonomics paper finds the optimal thumb manipulation space on a mobile phone using a Thumb Jacobian Matrix with research participants using a soft lock keyboard with movable keyboard area [10]. They explored

which areas on the mobile screen are most comfortable and least comfortable and presented it on a matrix. Le Xiong et al provides a theoretical basis for planning new layouts of smartphone keyboards for designers. (see figure 4).

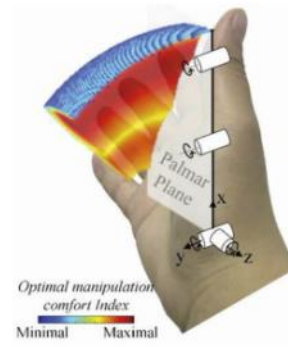


*Figure 4 – Thumb Comfortability Matrix*

Thumbly and Word Flow for IOS presents a keyboard layout more natural to the movement of the thumb [11]. However, they still use the QWERTY layout which was designed for two handed typing and not optimized for one thumb typing. Thumbly cost \$1.99 to purchase and Microsoft Word Flow got pulled from the app store for not being so popular. We believe comfortable keyboard layouts shouldn't be behind a paywall and want it to be accessible for all users.

Our model plans on incorporating the thumb comfortability matrix from Le Xiong et al to set initial coordinates of the keys to be where the thumb is most comfortable. Thumb comfortability scores will be incorporated into the performance evaluation of keyboards in the genetic algorithm pipeline proposed by Yin and Su. We are modifying the distance evaluation to be sequential typing rather than from a resting default position to better represent typing on a mobile keyboard.

An efficient mobile keyboard that is also ergonomic reduces hand cramps and promotes accessibility for all users.

### III. Simulation Details

In this section, we outline the simulation framework and methodology employed in our keyboard optimization project.

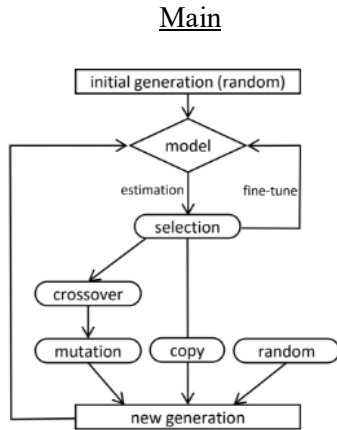


Figure 5 – Flow Chart

The study used the MKLOGA pipeline from the paper “Keyboard Layout Optimization and Adaptation” by Keren Nivasch and Amos Azaria, with a modification to the evaluation/selection step.

100 random keyboard layouts were generated as our initial population. These layouts were evaluated based on a score calculated from the total comfort score over the total distance traveled. The top 10 performing keyboards are selected to produce the next generation of keyboards through a crossover key swap algorithm. The process produces 80 child keyboards from two randomly selected parents to contribute half of their key locations to the child. The children have 3 of their keys randomly swapped to introduce genetic variance. The 10 parents, 80 children, and an additional 10 randomly generated keyboards are used as the next generation to repeat the cycle.

What differentiates our model from the one used by Keren and Amos is the difference in evaluation. They evaluated keyboards under

two-handed typing conditions and without accounting for comfortability while our model evaluated based on one-thumb typing and considered thumb ergonomics.

#### Data Used

The database used for the evaluation pipeline of this model is taken from kaggle.com [12], which the name, the-reddit-dataset-dataset-comments.csv. The file contains almost 5500 comments made on the /r/datasets subreddit. Comments written by bots or those that include strings that are normally not typed by users (such as links) were removed for a more accurate representation of content that users would use the keyboard to type.

#### Key Comfort

The comfort scores assigned to each key position in the keyboard layouts were inspired by the work of Xiong, Fu, and Deng in [2]. In it, they have 10 participants use their thumbs to evaluate the comfort range when each participant uses an average-sized mobile screen (iPhone 6). The results show a varying range of comfort ranges, represented with a comfort index that highlights parts of the screen that are more comfortable in warmer / redder colour and less comfortable colder / bluer colour. The results of all the tests are used to calculate and represent an average. In this model, this average is used, along with the chosen keyboard layout from Thumbly for IOS, to approximate the comfort level of each key position, as shown in Figure below.

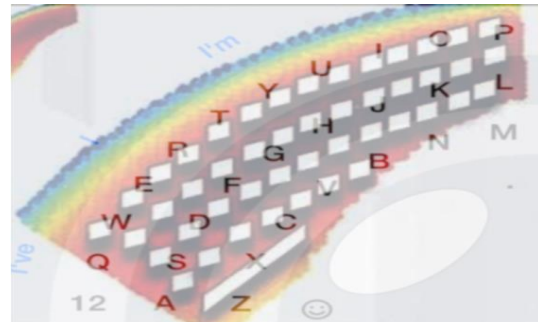


Figure 6 – Keyboard layout

### Evaluation Pipeline

In the `evaluate.py` module for a keyboard class, we've developed two functions to calculate the travel distance during typing.

The first function computes the Euclidean distance between any two keys on the keyboard, identified by `key1` and `key2`. It begins by retrieving unique codes for each key from `self.key_assignment` and then fetches the (x, y) coordinates for these codes from `self.key_coordinates`. Using the Euclidean distance formula, it calculates and returns the distance.

The second function estimates the total distance needed to type a character that requires the SHIFT key, by considering: The distance from the last typed key to "SHIFT", and then from "SHIFT" to the current character's key. It normalizes uppercase characters to lowercase for consistent mapping.

If there's no previous key (`last_key` is None), it directly calculates the distance from "SHIFT" to the current character's key, returning a tuple of the total distance and the current (normalized) character.

These functions aim to simulate the physical travel distance on a keyboard, especially for characters needing the SHIFT key.

For calculating the comfort score, the function retrieves the comfort score associated with each typed character from the `keyboard.key_comfort` dictionary, which maps key codes to comfort scores. This comfort score, influenced by key locations, is then added to a total comfort score. After processing all characters, the total comfort score aggregates the individual comfort scores for each typed character. The function also calculates a `combined_score`, which is the comfort score normalized against the total travel distance, multiplied by 1000 for scale adjustment. This combined score offers an efficiency or overall comfort measure per unit of travel distance on the keyboard, considering

both the physical effort and the inherent comfort of pressing the keys.

### Gui

Central to our keyboard optimization project is the Graphical User Interface (GUI), a pivotal component facilitating the exploration and analysis of keyboard layouts. The GUI serves as a visual conduit, providing users with a comprehensive view of keyboard design iterations and their associated performance metrics.

Built upon the `turtle` graphics and `tkinter` libraries, the GUI offers a user-friendly interface for interacting with the optimization algorithms and evaluating the efficacy of different keyboard configurations. It enables users to visualize keyboard layouts in real-time, allowing for intuitive navigation and comparison between designs.

The GUI code leverages two key functions: `display_mirrored_keys_with_horizontal_flip()` and `display_keyboard_gui()`. The former function is responsible for rendering mirrored keyboard layouts with horizontal flips, crucial for assessing layout symmetry and ergonomics. By utilizing `turtle` graphics, it plots each key on the screen based on its coordinates, enabling users to visualize the layout's spatial distribution.

The latter function, `display_keyboard_gui()`, creates a graphical window using `tkinter`, displaying the keyboard layout alongside performance evaluation scores. This window provides users with a detailed view of the layout's design and its corresponding metrics, fostering a deeper understanding of the optimization process.

In summary, the GUI serves as a vital tool in our endeavor to optimize keyboard layouts, offering researchers and practitioners a platform for experimentation, evaluation, and refinement.

## Results

We simulated 1000 generations with 1000 keyboards per generation. This time 100 top-performing parents are selected to produce 800 children and 100 random keyboards are contributing to each generation. 1,000,000 keyboards were generated for this run.

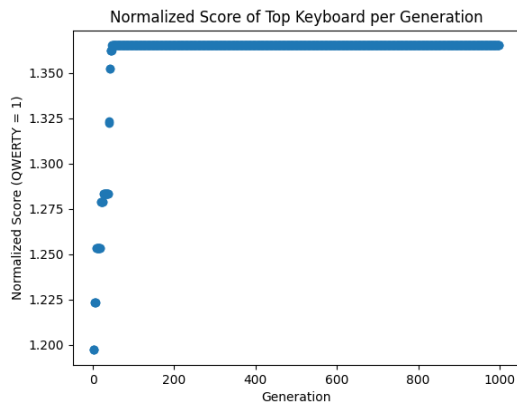


Figure 7 – Normalized score

Figure: 7) Performance Score of top keyboards per generation relative to QWERTY in 1000 generations of 1000 keyboards per generation.

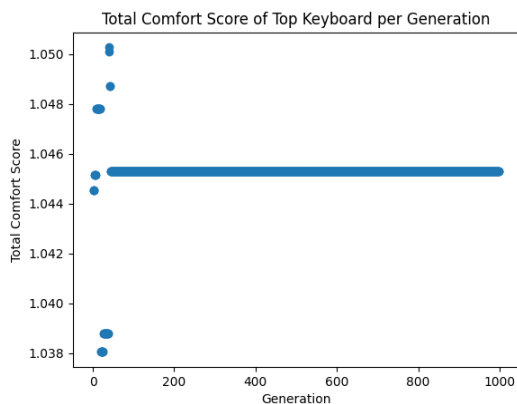


Figure 8 – Comfort score

Figure 08) Total Comfort Score of top keyboards per generation relative to QWERTY in 1000 generations of 1000 keyboards per generation.

We see a compromise of comfort score in favor of reduced total distance traveled by the sudden decrease and plateau at a lower level than the

peak.

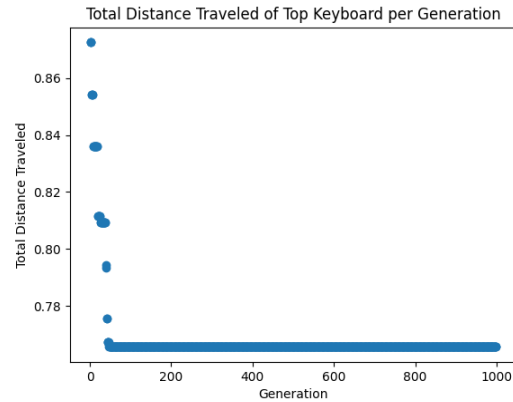


Figure 9 – Total Distance Travelled

Figure: 9) Total distance traveled of top keyboard per generation relative to QWERTY in 1000 generations of 1000 keyboards.

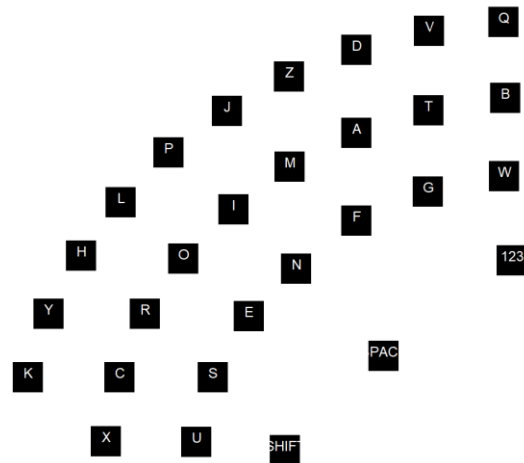


Figure 10 – Keyboard Layout

Figure 10) Example optimized keyboard after 1000 generations.

The top performing keyboard of 1,000,000 keyboard simulation outperforms QWERTY by 36.5% in performance score, 4.5% in comfort, and 26% reduced total distance traveled. Most of the optimizations happened in the first 100 generations before plateauing off. We have noticed that the letter 'E' seems to always be near the space and shift key (10/10 of the 50 generation runs). This made sense since it is the

most common English alphabet used and given the frequency keeping it close to space and shift would decrease the total distance traveled. The algorithm seems to favor optimizing for decreasing total distance traveled than the comfort score since position E is one of the least comfortable positions and the comfort score often decreases in compromise for reduced total

distance traveled. To optimize for more comfort, there should be a greater penalty in scoring implemented in future studies.

## References

- [1] R. Cristina, "The Impact of Mobile Technology In Our Lives," 15 Jun 2021. [Online].  
Available: <https://blog.mobiversal.com/the-impact-of-mobile-technology-in-our-daily-life.html>.
- [2] S. & Z. S. Azenkot, "Touch behavior with different postures on soft smartphone keyboards," September 2012. [Online].  
Available: <https://dl.acm.org/doi/10.1145/2371574.2371612>
- [3] E. Bonabeau, "Agent-based modeling: methods and techniques for simulating human systems," 14 May 2002. [Online].  
Available: <https://www.pnas.org/doi/10.1073/pnas.082080899>.
- [4] C. M. M. a. M. J. North, "Agent-based modeling and simulation," 2009. [Online].  
Available: Proceedings of the 2009 Winter Simulation Conference (WSC), Austin, TX, USA.
- [5] D. S. F. van, "Genetic algorithms for map labeling. s.n.," 2001. [Online].
- [6] M. Mitchell, "An introduction to genetic algorithms MIT," 1998. [Online].
- [7] D. S. C. B. G. Denis Anson, "One-Handed Typing: Layouts Do Matter," 2017. [Online].  
Available: <https://www.resna.org/sites/default/files/conference/2017/cac/Anson.html>.
- [8] T. A.-A. J. Ali Fadel, "QWERTY Keyboard? }.?BZQ is Better!," 30 November 2020. [Online].  
Available: <https://ieeexplore.ieee.org/document/9264079>.
- [9] A. S. Ana Beatriz Herthel, "Optimizing single-finger keyboard layouts on smartphones," August 2020. [Online].  
Available: <https://www.sciencedirect.com/science/article/pii/S0305054820300642>.
- [10] C. F. & S. D. Le Xiong, "Area Design of Keyboard Layout for Comfortable Texting Ability with the Thumb Jacobian Matrix," 3 October 2021. [Online].  
Available: <https://www.tandfonline.com/doi/full/10.1080/104>
- [11] J. Titcomb, "Microsoft's iPhone keyboard has a great way to type one-handed," 26 January 2016. [Online].  
Available: <https://www.telegraph.co.uk/technology/2016/01/26/microsofts-iphone-keyboard-has-a-great-way-to-type-one-handed/>.
- [12] Data base used for the keyboard text file  
<https://www.kaggle.com/datasets/thedevastator/the-meta-corpus-of-datasets-the-reddit-dataset>