Optimizing Keyboard Configuration Using Single and Multi-Objective Evolutionary Algorithms

Ahmer Khan and Kalyanmoy Deb
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan, USA
{khanahm2,kdeb}@msu.edu

COIN Report Number 2023003

ABSTRACT

An English language keyboard configuration involves 30 keys – 26 alphabets and four punctuation marks: comma, period, semi-colon, and question mark. Based on the sequence of letters that frequently appear in typing activities, one or more keyboard configurations will be optimal for a specific performance metric. Here, we consider the cumulative distance between two consecutive keystrokes to type out a large piece of text as the optimization objective, as it directly relates to the typing efficiency. The shorter the distance the fingers needs to move between strokes, the faster is the typing speed. When considering such a scenario, researchers often focus on a specific single objective one at a time and come up with the respective optimal keyboard configuration. The complete dissimilarity of these optimal, yet esoteric, keyboards from the current in-practice QWERTY configuration makes the new optimal keyboard configuration harder to learn and deploy on a larger scale. Hence in this study, we look at a multi-objective version of such a problem and consider two objectives – maximizing typing efficiency and maximizing similarity to QWERTY configuration. We make use of the NSGA-II algorithm and produce a Pareto set of keyboard configurations ranging from the most efficient to the most similar configurations. The range of solutions can potentially provide a design *innovation path* for gradually moving from the popular QWERTY configuration to more efficient configurations as a series of improvements.

CCS CONCEPTS

• **Theory of computation** → **Genetic programming**; *Theory of randomized search heuristics*; • **Hardware** → *Hardware description languages and compilation.*

KEYWORDS

Multi-objective optimization, Keyboard layout, combinatorial optimization.

ACM Reference Format:
Ahmer Khan and Kalyanmoy Deb. 2023. Optimizing Keyboard Configuration Using Single and Multi-Objective Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583133.3590580

1 INTRODUCTION

The QWERTY keyboard layout is ubiquitously employed as the standard English keyboard layout since the advent of computers. Considering the long-standing presence of the QWERTY layout, one may be inclined to believe that it presents, to some degree, an optimized arrangement with respect to some evaluation metric. However, there is no such indication that this is actually the case, with unproven theories for key placement ranging from using an arbitrary arrangement to using an arrangement that would prevent mechanical clashing during typewriter operation. To that end, we pose this problem as a combinatorial optimization problem for maximizing the typing efficiency of a user, where we use a distance metric to maximize typing efficiency since, the lesser distance the fingers have to type, the faster the typing speed. Our search space spans $30! = 2.65(10^{32})$ unique arrangements.

Though researchers have previously optimized keyboard layouts using certain evaluation criteria, these studies were based on single-objective optimization producing final solutions far from the heavily used QWERTY configuration limiting their widespread adoption [4, 11, 13–15]. Nivasch and Azaria [12] used a deep learning model with a genetic algorithm (GA) to optimize for the keyboard configuration. A text analyzer was used to generate 3,000 most frequent words and optimize the keyboard configuration based on that [9]. They optimized the keyboard for 26 letters only. The effect of keyboard layout and size is studied on smartphone typing performance [16]. In this paper, we pose the overall problem as a multi-objective optimization problem, where our goal is to maximize user efficiency but minimize dissimilarity from the QWERTY configuration. For the initial phase (single-objective optimization) of our problem, we employ a combinatorial Genetic Algorithm (GA) using 30 keys representing the 26 English alphabet characters A-Z as well as the comma, period, question mark, and semi-colon. For the multi-objective version of the problem, we develop a combinatorial version of the NSGA-II algorithm [3] and generate a Pareto-optimal (PO) set of solutions ranging from the most efficient to the most similar configurations. The range of obtained solutions can provide a path for gradually moving from the QWERTY configuration to more efficient configurations for specialized applications, demonstrating another beneficial advantage of employing an EMO algorithm to find a viable blueprint of solution steps.

In the remainder of this paper, we discuss single-objective and multi-objective evolutionary algorithms (EAs) developed for solving the keyboard configuration problem in detail in Section 2. The evaluation procedure is described in Section 3. Results of single-objective GAs and multi-objective GAs are presented in Sections 4 and 5, respectively. Finally, conclusions are drawn in Section 6.

2 COMBINATORIAL EVOLUTIONARY ALGORITHMS

We employ a combinatorial Genetic Algorithm (GA) [10] for the single-objective phase of the optimization problem and use a combinatorial version of the NSGA-II procedure [3] for the multi-objective version of the problem. For both versions, due to its availability, the QWERTY configuration is included in the initial population to jump start the evolutionary process. This highlights the use of a known solution, if available, in the initial population of an EA approach.

2.1 Single-objective Customized Permutation-based GA

To represent a keyboard with 30 distinct keys in an optimization algorithm, first, we need to decide on a representation scheme which will act as a genome in a GA. An individual (chromosome) in our population is represented as a string of 30 characters as a permutation of 30 numbers. Letters A to Z are numbered serially from 1 to 26. Thereafter, comma, period, question mark, and semi-colon are numbered 27 to 30, respectively. The index of each character in the string is then mapped to a corresponding position on the keyboard, thereby generating the keyboard configuration. This representation allows a total of $30! = 2.62(10^{32})$ different keyboard configurations. This introduces a large number of keyboards to be evaluated exhaustively to find the best possible keyboard. The problem of keyboard configuration calls for a numerical optimization procedure.

The goal in a keyboard configuration optimization is to have the maximum possible typing efficiency of some sort. As a main objective for keyboard configuration, we define a typing efficiency objective. To maximize typing efficiency we make use of a distance metric where the objective is to minimize the total typing distance between two consecutive keystrokes to type out a string of text. To do so, we first map the genomes representing keyboard configuration to square coordinate positions that accurately represent the modern keyboard shape. We then group keys by the fingers they are typed by and then calculate the Euclidean distances between all pairs within each group. From these calculations, we are able to create a lookup table for distances between keys. This ensures that distances are calculated only once at the initialization of the optimization run and used when evaluating the total typing distance for any given layout. Additionally, we do take into account the area of an individual key and the spaces between them in our distance calculation. It is clear that this evaluation procedure for typing efficiency depends on the text being used. We discuss more about this in Section 3.

The crossover operator to recombine two parents needs to be mindful of the validity of generated offsprings. Given its documented success with combinatorial problems, we choose to use the PMX crossover [7] as the crossover operator applied to a pair of parents with probability p_c. For parent selection, we choose to use k-ary tournament selection with replacement.

Since the nature of the problem is combinatorial, we also need a mutation operator that always produces a valid mutated offspring permutation. Here, we use the swap mutation [5] as the mutation operator. In such an operation an even-numbered set of characters is selected based on a mutation probability (p_m) applied to each position. If this results in an odd number of positions, the last position is ignored. The pairs are formed randomly from the selected positions. Each pair then swap positions to create a mutated individual.

At the end of each generation, we implement an elitist survival selection approach, in which both parent and offspring population of the same size are combined and the top half of the combined population (based on typing efficiency) is passed on to the next generation. We use a maximum number of generations ($T_{\max}$) to terminate an optimization run.

2.2 Multi-objective Permutation-based NSGA-II

In addition to the typing efficiency objective, we introduce a competing objective of dissimilarity with the QWERTY configuration creating another minimization objective function and pose the problem as a multi-objective combinatorial optimization problem. The dissimilarity with the QWERTY configuration is calculated as the sum of binary differences in each position of the keyboard. if the character in a position in the produced solution is different than the character in the QWERTY configuration it amounts to a dissimilarity metric value of 1. We then simply sum differences in each of the 30 positions to compute the second objective in the range [0,30]. The motivation for introducing this objective is to find keyboards that would have good typing efficiency but are not too dissimilar to the QWERTY configuration. This is because humans have apathy for change from status-quo, but may get convinced with a slight change if that leads to a better typing efficiency.

We use the popular NSGA-II algorithm [3] for the multi-objective version of the algorithm. We use the same crossover and mutation operators as in the single-objective case, as the representation stays the same. The termination condition is also set based on elapse of $T_{\max}$ generations. We use the pymoo distribution [1], but update the representation, crossover and mutation operators with those discussed above.

3 EVALUATION PROCEDURE

To evaluate on a short text containing a few words would prove to be insufficient as it would only train the model to optimize typing that text and result in over-fitting. To that end, we initially consider using a large English lexicon of the most frequently used words. However, this also proved to be insufficient as using this for training would not take into account several factors of the English language, such as representing the frequency of which words appear, the semantic use of words in relation to others, and the presence of infrequently used words and the necessity to be able to efficiently type those as well. To solve the problem of which text to use for training, we selected a Kaggle dataset containing a corpus of approximately 1.7 million arXiv abstracts. From this dataset, we clean

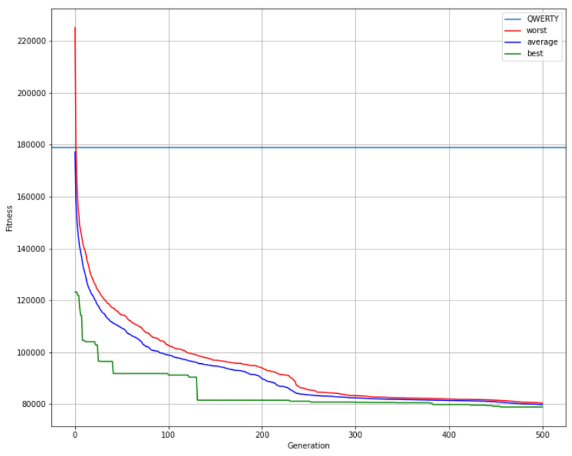


Figure 1: Progress of the proposed single-objective GA for the first 500 generations.

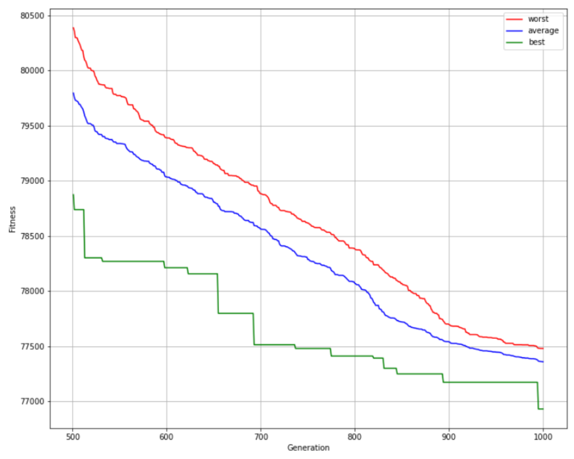


Figure 2: Progress of proposed ingle-objective GA for the next 500 generations.

the data of extraneous characters, such as mathematical symbols, and read 600 abstracts for training and an additional 300 abstracts for validation. The read-in texts then became usable arguments for our objective evaluation function that computes the total distance required to type out the text with a given configuration. There is no simple way to know the minimum and maximum value possible for this objective, as it totally depends on the nature of the text being used.

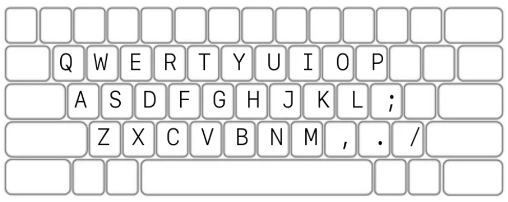
For the multi-objective version, the second objective is a straightforward matching problem of counting the number of positions in which a given keyboard is different from the QWERTY keyboard. Note that the lower and upper bound of this second objective is zero and 30, respectively. A value of zero means that the QWERTY keyboard configuration is achieved.

4 SINGLE-OBJECTIVE GA RESULTS

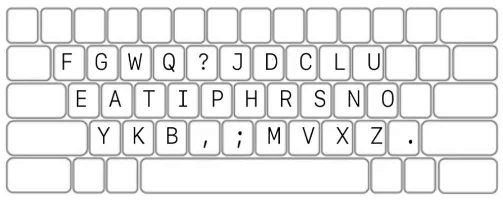
For our experiments, we use a tournament of $k = 5$ individuals, a crossover probability of $p_c = 0.8$, a mutation probability of $p_m = 0.2$, population size of $N = 100$, and maximum number of generations as $T_{\max} = 1,000$. Figure 1 shows the performance of the single-objective GA for the first 500 generations. We plot the total typing distance (f_1) moved by fingers for the QWERTY configuration (sky blue), the best (green) and worst configurations (red) found by our proposed GA, and the average of the whole population in each generation. As can be seen from the graph, some randomly initialized configurations are much better than QWERTY configurations with respect to our evaluation procedure.

Figure 3a shows the layout of the QUERTY keyboard. Figure 2 shows the performance of the GA for the next 500 generations. Here, we do not plot the performance of QWERTY as it is a fixed configuration. We can observe a smooth linear decline in the distances of the worst solution and the average of the population hinting at the convergence of the algorithm and by the 1,000-th generation, it seems the GA has converged well by then. Figure 3b shows the final optimized layout found by the proposed GA.

From our experiment, we found that even within the initial population, the best solution outperforms the QWERTY layout. In addition, it appears that the average of solutions surpasses QWERTY in only a few generations. With these results, we can infer that QWERTY is far from being an optimized solution and that it is certainly possible to have a better keyboard configurations with an objective like maximizing typing efficiency. It is not entirely clear how the QWERTY configuration was designed originally. Since humans take training to improve the typing efficiency and have no other keyboard configuration to compare their learning efficiency with, it stays as an open challenge and effort to re-design the keyboard configuration for a better workspace efficiency. This study indicates and motivates such an effort in the near future.



(a) QWERTY keyboard configuration with fitness: 178779.



(b) GA-optimized keyboard configuration with fitness: 76933.

Figure 3: Comparison of GA-optimized keyboard with the popularly used QUERTY keyboard. Two keyboards are quite different from each other.

5 MULTI-OBJECTIVE NSGA-II RESULTS

We use the NSGA-II algorithm with our above-mentioned EA hyper-parameter values. We use a population size of 100 individuals and run the algorithm for 200 generations. We customize our initial population with QWERTY configuration and some popularly-known configurations throughout history, e.g. Dovark, Colemak [2, 8]. The generated Pareto front can be seen in Figure 4, where f_1 is the typing efficiency objective, while f_2 is the dissimilarity from the QWERTY objective. If we look at the extreme point on the Pareto front it would hint at the QWERTY configuration as it has zero dissimilarity, but a high typing inefficiency. If we move to the left

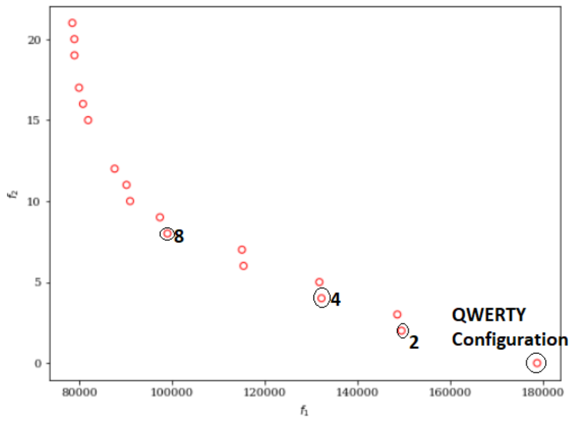


Figure 4: Obtained Pareto front for the multi-objective version of the keyboard optimization problem.

of the Pareto front, we see keyboard configurations get improved in typing efficiency but become more dissimilar to the QWERTY configuration, hence providing an *innovation path* to gradually improve the QWERTY configuration to better and more efficient Keyboard configurations.

5.1 Effect of Multiple Runs

Next, we evaluate the proposed algorithm's behavior over multiple runs. We execute five runs, each with a different random initial population, and plot the attainment surface [6] of the five different parent fronts generated. Figure 5 shows that the best (0%), median (50%), and worst (100%) attainment curves are all almost identical, implying the algorithm's robust performance over multiple runs. The results are not affected by the stochasticity in the algorithmic procedure and converge to the final front (Figure 4) every time.

6 CONCLUSIONS

Our goal in this study has been to find the most typing-efficient keyboard configuration for English language alphabets. We have shown that our combinatorial genetic algorithm can produce an optimized configuration that greatly outperform the ubiquitous standard QWERTY layout. But keeping in mind the pragmatic issues of deployment challenge due to the popular use of the QWERTY keyboard, we have also proposed a multi-objective approach to find 18 different trade-off keyboard configurations. The QWERTY configuration stays at one extreme of the resulting Pareto

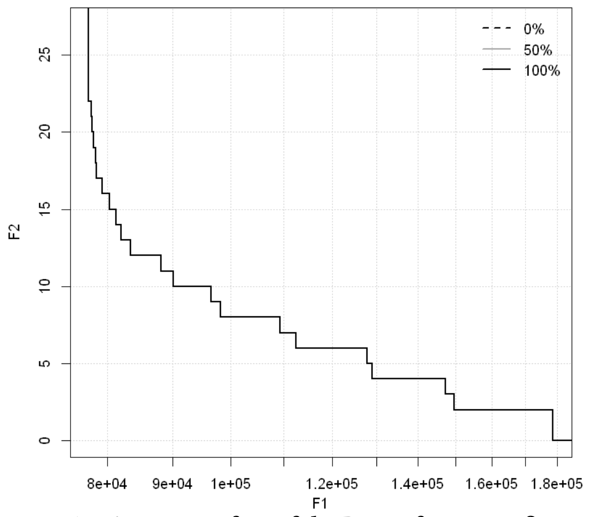


Figure 5: Attainment surface of the Pareto front over five runs of the NSGA-II algorithm shows consistent performance.

front. We have argued that the Pareto set can be viewed as a series of gradual updates, starting with the QWERTY configuration, to gradually achieve a high typing-efficient keyboard configuration, if desired. The concept needs further exploration for achieving design innovations from the status quo in other problems. In addition to the position of the keys, the layout of keyboard shape can also be optimized for users to have a better typing efficiency. These ideas can also be applied for keyboards in other languages.

REFERENCES

[1] J. Blank and K. Deb. 2020. pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509.
[2] R. C. Cassingham. 1986. *The Dvorak keyboard: The ergonomically designed typewriter keyboard now an American standard.* Freelance Communications.
[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. 2002. A fast and Elitist multi-objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.
[4] P. S Deshwal and K. Deb. 2006. Ergonomic design of an optimal Hindi keyboard for convenient use. In *2006 IEEE International Conference on Evolutionary Computation.* IEEE, 2187–2194.
[5] A. E Eiben and J. E Smith. 2015. *Introduction to evolutionary computing.* Springer.
[6] C. M. Fonseca, V. Grunert da Fonseca, and L. Paquete. 2005. Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In *Third International Conference on Evolutionary Multi-Criterion Optimization, EMO-2005.* Berlin: Springer, 250–264.
[7] D. E. Goldberg and R. Lingle. 2014. Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications.* Psychology Press, 154–159.
[8] D. Gutiérrez, M.A. Ramírez-Moreno, and A. G. Lazcano-Herrera. 2015. Assessing the acquisition of a new skill with electroencephalography. In *7th Int. IEEE/EMBS Conf. on Neural Engineering (NER).* IEEE, 727–730.
[9] O.A.H. Habibi and O. Korhan. 2020. Application of a genetic algorithm to the keyboard layout problem. *PloS one* 15, 1 (2020), e0226611.
[10] J. H. Holland. 1975. *Adaptation in Natural and Artificial Systems.* MIT Press.
[11] M. Kafaee, E. Daviran, and M. Taqavi. 2022. The QWERTY keyboard from the perspective of the Collingridge dilemma: lessons for co-construction of human-technology. *AI & SOCIETY* (2022), 1–13.
[12] N. Keren and A. Azaria. 2021. A Deep Genetic Method for Keyboard Layout Optimization. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI).* IEEE, 435–441.
[13] E. Khorshid, A. Alfadli, and M. Majeed. 2010. A new optimal Arabic keyboard layout using genetic algorithm. *International Journal of Design Engineering* 3, 1 (2010), 25–40.
[14] C. Liao and P. Choe. 2013. Chinese keyboard layout design based on polyphone disambiguation and a genetic algorithm. *International Journal of Human-Computer Interaction* 29, 6 (2013), 391–403.

[15] K. Robin. 1975. Typing speed, keying rates, and optimal keyboard layouts. In *Proceedings of the Human Factors Society Annual Meeting*, Vol. 19. SAGE Publications Sage CA: Los Angeles, CA, 159–161.

[16] C. J. Turner, B. S. Chaparro, I. M. Sogaard, and J. He. 2020. The Effects of Keyboard Layout and Size on Smartphone Typing Performance. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 64. SAGE Publications Sage CA: Los Angeles, CA, 985–989.