Multi-objective Optimization for Efficient Brahmic Keyboards

Albert Brouillette[1] *Dr. Utpal Sharma*[2]
Dr. Jugal Kalita[1]
(1) University of Colorado, Colorado Springs, Colorado, USA
(2) Tezpur University, Napaam, Assam, India
abrouil2@uccs.edu, utpal@tezu.ernet.in, jkalita@uccs.edu

ABSTRACT
The development of efficient keyboards is an important element of effective human-computer interaction. This paper explores the use of multi-objective optimization and machine learning to create more effective keyboards. While previous research has focused on simply improving the expected typing speed of keyboards, this research utilizes multiple optimization criteria to create a more robust keyboard configuration. As these criteria are incorporated, they will often conflict with each other making this a complex optimization problem. Machine learning techniques were utilized and were proven to be an effective tool in keyboard optimization. The results reported here demonstrate that multi-objective genetic algorithms can be used to efficiently generate optimized keyboards. An English keyboard designed after 20000 generations was able to double the efficiency of the unoptimized QWERTY keyboard for multiple constraints. Despite having twice the number of characters, an Assamese keyboard was generated that performed better than the QWERTY layout.

KEYWORDS: Soft keyboards, user interfaces, Brahmic scripts, optimization, genetic algorithms, Android development..

Proceedings of the Second Workshop on Advances in Text Input Methods (WTIM 2), pages 29–44,
COLING 2012, Mumbai, December 2012.

1 Introduction

As technology progresses, it is important that user interfaces improve the efficiency of information transfer between users and computers. Because keyboards are a continuing, integral part of human-computer interactions, research into improving their efficiency is particularly valuable to improving data entry. The speed at which a user can input data can be greatly affected by the arrangement of keys on a keyboard. Based on the large number of possible key combinations, hand optimizing a keyboard is a tedious process. The use of artificial intelligence can make the process of generating optimal user interfaces much more efficient.

There are multiple considerations in determining the effectiveness of a keyboard. For example, the size, position and spacing of the keys on the device being used can dramatically change the speed that a user can input data. Additional factors such as accommodating user disability and adjusting to user preference can also affect the usability of a keyboard. An effective keyboard needs to be adaptable to the specific needs of each user. Touch-screen devices, with their soft-keyboards, have the potential to provide the flexibility required to adapt to these constraints. Since all of these factors are interconnected, it is difficult to accurately model them separately.

Previous research projects have used single objective optimization to increase the typing speed of a keyboard. While this method has produced good results, it ignores other factors that affect efficient typing. These machine generated keyboards could be more useful if a broader selection of constraints is used. This more generalized approach would allow optimization based on a wide variety of constraints such as ease of learning or user disability. The result of the optimization process is a set of optimal solutions that consider all of the constraints with a variety of weights.

This paper discusses the use of multi-objective optimization in creating efficient keyboards. In order to demonstrate value of this research on a global scale, we chose to develop keyboards for two very different languages, English and Assamese. Assamese is an Indic language from Northeast India, spoken by about thirty million people. The number of characters in this language makes the problem more complex and allows us to develop a more generic approach to the solution. We use Assamese as an exemplar of languages of the Indic class of languages, which are spoken by a more than a billion people, particularly in South Asia. At this time, these languages suffer from a lack of efficient and easily learned soft keyboards

The first constraint to consider is the typing speed, based on the expected typing style of the user. For example, some users will use a single finger, while others might use both of their thumbs for input. Another constraint for consideration is the ease with which a user can learn the layout of the keyboard. This can be represented as a constraint where the keyboard is biased toward a familiar, existing layout.

As constraints are added, some of them may conflict with each other, requiring a compromise to be made. The use of multi-objective genetic algorithms has been an effective and efficient approach to solving complex optimization problems(Konak et al., 2006). Multi-objective genetic algorithms allow the solutions to be optimized based on all of the objectives simultaneously. A solution is considered optimal if it performs better than or equal to the other solutions for every constraint. These optimal solutions will form a set known as a pareto front (Figure 1). This front contains the most efficient solutions, each involving a different compromise between the objectives. The set of solutions can be saved and accessed by the user based on their individual needs.

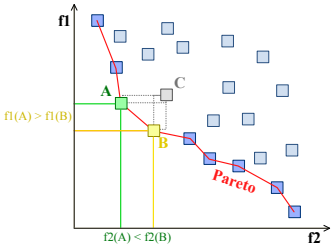

Figure 1: A simplified example of a pareto frontier. The darker points are the set of pareto optimal solutions. Source: WIKIPEDIA

2 Related Research

Some of the earliest research in modeling human-computer interaction was done by Fitts in the 1950's(Fitts, 1954). The model developed was able to compute the average time required for a human to move between two targets. Fitts' model is commonly expressed as the equation:

$$MT = \frac{1}{IP} \log_2 \left(\frac{D}{W} + 1 \right) \tag{1}$$

where IP is the index of performance for the device, D is the distance between the targets, and W is the size of the target.

This model has been used by many other researchers to estimate the time required for a human to move between controls in a graphical user interface(MacKenzie, 1992).

MacKenzie extended Fitts' law to model the human input speed on with a QWERTY keyboard on a mobile device(MacKenzie and Soukoreff, 2002). This model was later compared with human testing data and was shown to be a reasonably accurate estimate of actual human performance(Clarkson et al.).

In addition to analyzing the physical action of pressing the keys on the keyboard, researchers have also investigated the ability of users to learn a new keyboard layout. As researchers have worked on optimizing keyboards, it has been acknowledged that one of the limiting factors in users typing speed is the time spent searching for the keys(MacKenzie and Zhang, 1999). Smith and Zhai suggested adding an alphabetic bias to an optimized keyboard(Smith and Zhai, 2001). Lee and Zhai investigated techniques to help users quickly learn a new keyboard(P. U. Lee, 2004).

Much of the early development in keyboard design focused on creating variations of alphabetic and QWERTY layouts for the English language. The primary objective was to improve the ability of the keyboard to conform to mechanical limitations.

As technology improved, designers were able to focus on improving the typing speed of the keyboards. MacKenzie created one of the first character-frequency optimized layouts, the OPTI keyboard(MacKenzie and Zhang, 1999). The use of the Metropolis random walk algorithm was able to further increase the efficiency of soft keyboards(Zhai et al., 2000). These techniques have been able to improve expected typing speed up to 42 WPM.

Genetic algorithms have been used to achieve even higher levels of efficiency in keyboard design(Raynal and Vigouroux, 2005). (Hinkle et al., 2010) and (Hinkle et al., 2012) had worked extensively with optimization of Assamese and other Indic language keyboards. Their optimization was carried out with the single objective of maximization of input speed. They had created four different soft keyboards for Assamese: flat alphabetic, layered alphabetic, flat GA-designed and layered GA-designed.[1]

Gajos and Weld developed a technique for evaluating the effectiveness of user interfaces with respect to the individual needs of each user. With this analysis, they were able to generate personalized user interfaces with their SUPPLE project(Gajos et al., 2010).

3 Genetic Algorithm Optimization

The multi-objective optimization for this project was done using the NSGA-II multi-objective genetic algorithm(Deb et al., 2002). This algorithm was chosen based on its efficiency and effectiveness in multi-objective optimization. The algorithm was implemented using the JCLEC Java library[2]. This is an open-source library that provides implementations for several genetic algorithms.

For the purpose of keyboard design, two evaluators were written for designing keyboards on two devices. One evaluates the keyboard for its effectiveness for use as an on-screen keyboard for a desktop computer accessed with a mouse. The other evaluator tests the keyboard's effectiveness for both single finger and two-thumb input on a touch-screen mobile phone. The genetic algorithm was implemented using ordered crossover for recombination. As in (Hinkle et al., 2012), a mutation probability of 0.08% was used.

3.1 Preparation and General Keyboard Layout

Before starting the optimization process, it was necessary to determine a basic shape and pattern for the keyboards. In his research on user interface design, Ahlström found that expert users can make selections 35% faster from a square menu(Ahlström et al., 2010). Hinkle used a square layout for designing keyboards for Brahmic scripts (Hinkle et al., 2012). Based on this research, we decided that the on-screen keyboards should be designed assuming an essentially square shape with the keys being filled starting at the top.

This approach was impractical for a mobile device because the necessary size of the keyboard would have filled the entire screen. A practical solution to this problem was to base the keyboards on a variation of the QWERTY keyboard commonly used on mobile devices. Understandably, this shape would not work for languages with more characters. We designed the keyboards for the Assamese language assuming the use of two smaller QWERTY-shaped keyboards. The user can quickly switch between these keyboards as they type.

The first experiment was to establish a basis for comparison by evaluating the unoptimized QWERTY layout with the fitness function. The result was an estimated typing speed of 47 WPM for the first objective. The second objective reported on average distance of 2.88 keys between two characters in the same group. This analysis is shown in Figure 2.

[1]The first two were created by hand and the last two were created by single-objective GA optimization. The first was simply an alphabetic layout of the keys on a single keyboard and the second had a layer of alphabetic diacritics that popped up when one typed a consonant. The third was similar to the first, and the fourth to the second. The assumption in all four of these keyboards was that one types with a single finger. In addition, there was only one board

Figure 2: The analysis of the QWERTY keyboard using our evaluator. This layout has an estimated typing speed of 47 WPM and an average distance of 2.88 keys between two characters in the same group. The lower diagram shows the group number of each character.

3.2 Values for Fitts' Law Constants

Our evaluation of the keyboards relies on Fitts' Law to estimate the top typing speed for each keyboard layout. In order to accurately calculate the input speed for each device, it was first necessary to measure the Fitts' law Index of Performance(IP) for each device. This requires human testing to find the average movement time on each device. The approach to this calculation is similar to that used by Zhai(Zhai, 2004). For this experiment, we gave the human testers a series of randomly placed targets of varying sizes. The average time between targets was measured and used to calculate the Index of Performance based on the Index of Difficulty(ID).

3.2.1 Setting up the Experiment

The equation to calculate the IP for Fitts' Law is:

$$IP = \frac{ID}{MT} \tag{2}$$

where

$$ID = \log_2\left(\frac{D}{W} + 1\right) \tag{3}$$

The basic approach to this calculation is to have a user move between two targets with a varying ID with the test program calculating the average time(MT) between the targets. After several tests, it is possible to use the average ID and MT to calculate the IP for the current device. A program was used to generate random targets and calculate the time required to move between them(See Figure 3).

in each case. We present a comparison of these keyboards with our results in Table 3 .

[2]Available at http://jclec.sourceforge.net/

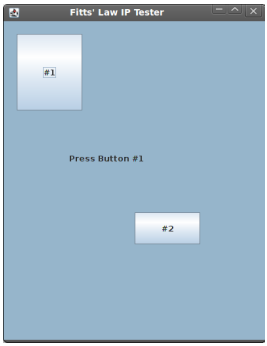

Figure 3: A screenshot of the IP testing program.

	Average Time	Average ID	IP
Mouse	0.53	2.58	4.9
Touch-screen: Right thumb	0.54	2.58	4.8
Touch-screen: Left thumb	0.63	2.58	4.1

Table 1: Results of IP test.

3.2.2 Results

The test program was used to calculate the IP for two applications. The results were obtained from 1000 data points after 100 consistent calculations. The first device was a desktop computer accessed using a mouse. This returned an IP value of 4.9. The second test was with a touch-screen device. The goal of this test was to calculate the constants for two-thumb typing. This required a separate calculation for each thumb. The result for a right-handed user was an IP of 4.8 for the right thumb and an IP of 4.1 for the left thumb.

3.3 Objective Constraints: Typing Speed

A primary objective in the keyboard optimization problem is to increase the typing speed. Our approach is to reduce the average time needed to move between two characters. This should result in the highest frequency characters being placed close together, minimizing the movement distance.

The average time between characters is calculated using an adaptation of Fitts' Law.

$$\tilde{t} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{P_{ij}}{IP}\left[\log_2\left(\frac{D_{ij}}{W_i}+1\right)\right] \tag{4}$$

where P_{ij} is the frequency of each digraph, IP is the calculated Index of Performance for the current device, D_{ij} is the distance between the two keys, and W_i is the width of each key. When

a digraph consists of a repeated letter, it is assumed to take constant time. The experimental value for this constant is 0.127.

For two-thumb typing on the touch screen device, we calculate the average time between characters using a method similar to MacKenzie's model(MacKenzie and Soukoreff, 2002). The Fitt's law equation is used when the digraph is formed from two characters pressed with the same thumb. When the digraph consists of characters pressed with opposite thumbs, the mean time is chosen to be the greatest value between, either $^1/_2$ of the constant T_{REPEAT}, or the Fitts' law calculation to move from the character last pressed by that thumb. For the Assamese two-keyboard arrangement, we add an experimental constant value for the average time required to change between keyboards.

3.4 Objective Constraints: Ease of Learning

The simplest approach to creating an easy to learn interface is follow a pattern that is already familiar to the users. Smith and Zhai did research comparing the performance of novice users on optimized keyboards with and without an alphabetic bias(Smith and Zhai, 2001). Given two keyboards with similar predicted typing speeds, they found that the users were able to learn the alphabetic biased keyboard more quickly and performed up to 10% better.

The approach taken in this paper is to group alphabetically close characters together. For each language, we organized the characters into groups based on their position in the alphabet.

We implemented this constraint as a minimization problem for the average distance between any two characters in the same group. It should be noted that this does not consider the time between characters, it is simply the visual distance between the characters. The objective is to allow the user to more quickly find each character in these smaller alphabetic clusters.

3.5 Groups Split Across Two Keyboards

This constraint is an extension to the ease of learning that is unique to the Assamese mobile keyboard. The layout of the Assamese mobile keyboard is split between two smaller, QWERTY-sized keyboards. This constraint prevents the character groups from being split between the two keyboards. This is implemented as a penalty for keyboards that have many split groups. The goal is to minimize the number of split groups.

4 English Keyboard Optimization

We used the English language as our basis for comparison with the other languages. The smaller character set made it easier to evaluate the results of the optimization.

The specific keyboard shape for the single-input on-screen keyboard, was a 6×5 square grid. The two-input mobile keyboard was modeled after the QWERTY layout. For the ease of learning constraint, we created 5 character groups, {{a, b, c, d, e, f},{g, h, i...}...}.

4.1 Optimization: Single-Input

Using the specifications above, we used the genetic algorithm to optimize the key positions. The optimized keyboard was generated from a population of 5000 solutions allowed to evolve over 20000 generations. The result was a pareto front containing 900 optimal solutions. For a representative solution, we selected the solution with the highest ratio (Objective 1/Objective

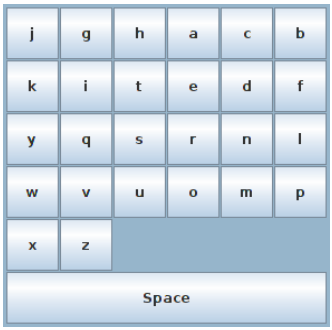

Figure 4: Optimized English keyboard generated with a population of 5000 over 20000 generations. This layout has an estimated typing speed of 46 WPM and an average distance of 1.1 keys between two characters in the same group.

2). The evaluator reported this solution as having an estimated typing speed of 46.43 WPM and an average distance of 1.1 keys between two characters in the same group(See Figure 4).

4.2 Optimization: Two-Input Mobile Keyboard

We ran a test with a population of 5000 solutions evaluated over 20000 generations. This experiment generated a large set of optimal keyboard layouts. A representative solution had an estimated typing speed of 84.2 WPM and an average distance of 1.43 keys between two characters in the same group(See Figure 5). Based on the evaluator, this keyboard performs nearly 2 times well as the QWERTY keyboard for both of the constraints.

4.3 Optimization: Three-Constraint Mobile Keyboard

In order to observe how a compromise is made between a set of constraints, we wanted to run the genetic algorithm with more constraints. For a final test we set up the algorithm to optimize a mobile keyboard for both single finger and two-thumb input. The goal of this experiment was to create a more universal keyboard that would allow a user to typing with either of these techniques. We maintained the ease of learning constraint so this test implemented three constraints.

This keyboard was designed based on the QWERTY shape like the two-input mobile keyboard. In Figure 6, we show a representative keyboard from the pareto set. This keyboard has an estimated typing speed of 81.99 WPM for two-thumb input and 26.96 WPM for single-finger input. There is an average distance of 1.46 keys between two characters from the same group.

4.4 Human Testing

The human testing had two objectives: to prove that this keyboard was an improvement over an unoptimized alphabetic layout and to show that our ease of learning constraint was effective in improving typing rates for novice users.

The human testers were given two different keyboards selected from the pareto front: the

Figure 5: Optimized English mobile keyboard generated with a population of 5000 over 20000 generations. This layout has an estimated typing speed of 84 WPM and an average distance of 1.43 keys between two characters in the same group. The lower diagram shows the group number of each character.

Figure 6: English keyboard optimized for both single finger and two-thumb input. This keyboard was generated with a population of 5000 over 20000 generations. This layout has an estimated typing speed of 82 WPM for two-thumb input and 27 WPM for single-finger input. There is an average distance of 1.46 keys between two characters in the same group. The lower diagram shows the group number of each character.

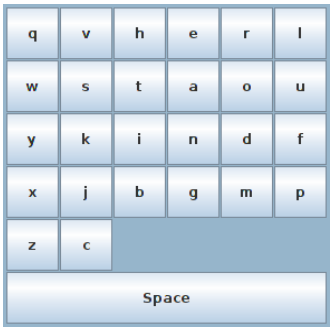

Figure 7: Optimized English keyboard generated with a population of 5000 over 20000 generations. This layout has an estimated typing speed of 50 WPM and an average distance of 1.9 keys between two characters in the same group.

keyboard with the highest predicted typing speed(See Figure 7), and the keyboard with the best compromise between the two objectives(Shown above in Figure 4). In order to facilitate a reasonable comparison, the testing process was similar to that used by Smith and Zhai(Smith and Zhai, 2001). The first task given to the testers was to type the entire alphabet. This was representative of the average speed that the users could find a character of the keyboard. The testers were then given a set of phrases to type. For a direct comparison, this set of phrases was the same as those used by Smith and Zhai.

The testers typed in 10 minute sessions once a day for a week. Figure 8 shows a graph of the average typing speed over these 7 sessions. The keyboard with the alphabetic bias performed around 10% better than the optimal keyboard for the first test. As the testing progressed, the performance on the optimal keyboard improved until it matched the alphabetic keyboard.

Based on the character frequency analysis, the optimal keyboard has a top typing speed 10% higher than the alphabetic keyboard. However, in an empirical test, novice users were able to type 10% faster on the alphabetic keyboard. The learning curve for the alphabetic keyboard was much quicker. The testers were able to find all of the characters on the alphabetic keyboard twice as fast as the optimal keyboard. On average, the testers found the keys in 10 seconds for the alphabetic keyboard and in 24 seconds for the optimal keyboard.

5 Assamese Keyboard Optimization

The Assamese language has more than twice the number of characters used in English. Hick's Law relates a users selection speed to the number of choices available(Hick, 1952). From this, we can assume that the efficiency is improved when a keyboard has fewer characters. We noticed that the Assamese language has two characters two represent each vowel. The vowel is written explicitly at the beginning of a word, but it is represented as a diacritic mark when used inside a word. We decided to create this distinction inside the user interface program instead of creating separate keys. When the user presses the vowel key, the appropriate symbol is displayed based on the context.

The specific keyboard shape for the single-input on-screen keyboard was a 8×7 square grid.

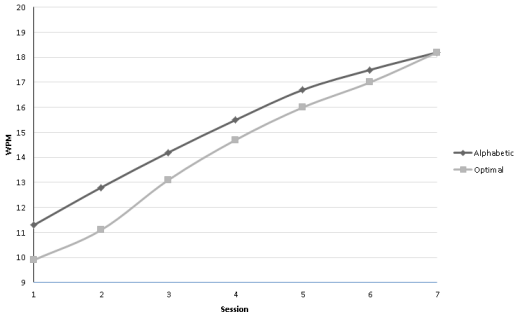

Figure 8: Results from human testing. The typing speed is an average computed from the results of 5 testers. During the early tests, a novice user was able to type around 10% faster using a the keyboard with alphabetic bias.

The two-input mobile keyboard was modeled as two QWERTY-shaped keyboards.

The large number of characters in the Assamese language made it difficult to determine the best method for grouping the characters for the ease of learning objective. We decided to create one group for the vowels and divide the consonants into 9 sub-groups based on the consonant rows. In order to determine the best combination of groups, we ran a series of tests to find the relation between the size of the groups and the predicted typing speed. Table 2 shows the results of these tests. There appears to be little variation in the typing speed between the different combinations of groups. The best results were achieved when the vowels were not grouped together. The combination of 4 groups appears to be the best compromise between group size and typing speed.

5.1 Optimization: Single-Input

Using the specifications above, we used the genetic algorithm to optimize the key positions. As with the English keyboard optimization, the optimized keyboard was generated from a population of 5000 solutions allowed to evolved over 20000 generations. The result was a pareto front containing 900 optimal solutions. For a representative solution, we selected the solution with the highest ratio (Objective 1/Objective 2). The evaluator reported this solution as having an estimated typing speed of 38.8 and an average distance of 1.6 keys between two characters in the same group(See Figure 9).

5.2 Optimization: Two-Input Mobile Keyboard

The next experiment involved generating an optimized mobile keyboard for the Assamese language. This test was run with a population of 5000 over 15000 generations. This test implemented the third constraint to eliminate groups being split between the two keyboards. For this test, we created 4 character groups to cluster for the second constraint. A representative solution had an estimated typing speed of 52 WPM and an average distance of 2 keys between two characters in the same group(See Figure 10).

# of Groups	Avg. Group Size	WPM	Avg. Group Dist.	Vowels Free
1	40.0	38.99	3.342	YES
1	26.0	38.29	2.632	NO
2	20.0	39.01	2.333	YES
2	17.3	38.43	2.333	NO
3	13.3	38.50	1.887	YES
3	13.0	38.07	1.917	NO
4	10.0	39.10	1.639	YES
4	10.4	38.46	1.759	NO
9	4.44	38.39	1.022	YES
9	5.20	38.01	1.190	NO

Table 2: Results of changing the number of groups.

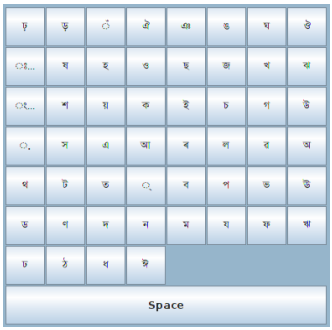

Figure 9: Optimized Assamese keyboard generated with a population of 5000 over 20000 generations. This layout has an estimated typing speed of 39 WPM and an average distance of 1.6 keys between two characters in the same group.

Keyboard	WPM
Flat alphabetic	25.1
Layered alphabetic	33.9
Flat GA-designed	34.2
Layered GA-designed	40.2
Flat multi-objective	38.8
Multi-objective mobile	52.0

Table 3: Comparison of our keyboards with those designed by (Hinkle et al., 2012).

Figure 10: Optimized Assamese keyboard generated with a population of 5000 over 20000 generations. This layout has an estimated typing speed of 52 WPM and an average distance of 2 keys between two characters in the same group. The evaluated alphabetic groups are highlighted.

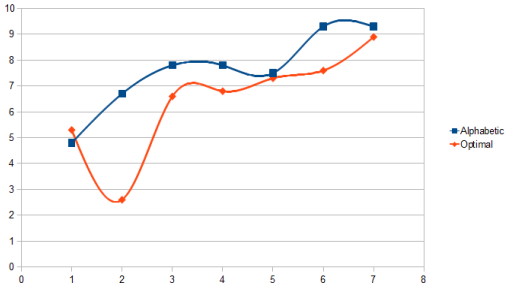

Figure 11: Preliminary results from human testing of the Assamese keyboards.

5.3 Human Testing

Under conditions identical to those in Section 4.4 , preliminary results were obtained from a single tester. While not as conclusive as the results for the English language, the graph in Figure 11 shows a similar pattern for the learning curve.

6 Future Work

The results reported for the performance of the English keyboards showed a significant improvement in early typing speeds for novice users. In order to make a conclusive comparison, the results reported for the Assamese language will need to be verified through additional human testing. Through the process of testing we hope to confirm the optimal number of character groups and the ease of learning. In order to facilitate the human testing process, we plan to make the keyboards available on-line for potential users to download. We also plan to make our mobile keyboards available on the Android market in return for user feedback on the usability

of the keyboards.

Valuable information could be gained by optimizing keyboards for other languages. An interesting comparison could be made between the results for different languages. In the immediate future, a comparison could be made between the Assamese language and the Bengali language which shares the same character set.

Conclusion

The design of efficient user interfaces is critical for continued progress in human-computer interaction. The large number of variables in user interface design make it difficult to optimize interfaces. Multi-objective genetic algorithms provide a convenient method for optimization in applications that have a large number of constraints.

The use of multi-objective genetic algorithms was shown to produce very good results for creating optimized keyboards. An English keyboard designed after 20000 generations was able to double the efficiency of the unoptimized QWERTY keyboard for multiple constraints. Despite having twice the number of characters, an Assamese keyboard was generated that performed better than the QWERTY layout. Future work will validate the results discovered so far.

Acknowledgment

The work reported in this paper was supported by NSF Grant ARRA 0851783.

The authors would like to thank Devraj Sarmah for his assistance in testing the Assamese keyboards.

References

Ahlström, D., Cockburn, A., Gutwin, C., and Irani, P. (2010). Why it's quick to be square: modelling new and existing hierarchical menu designs. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1371–1380, New York, NY, USA. ACM.

Clarkson, E., Clawson, J., Lyons, K., and Starner, T. An empirical study of typing rates on mini-qwerty keyboards. *CHI '05 extended abstracts on Human factors in computing systems*, pages 1288–1291.

Deb, K., Pratap, A., Agarwal, S., and Meyrivan, T. (April 2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*.

Fitts, P. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381.

Gajos, K. Z., Weld, D. S., and Wobbrock, J. O. (2010). Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12–13):910 – 950.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4:11–26.

Hinkle, L., Brouillette, A., Jayakar, S., Gathings, L., Lescano, M., and Kalita., J. (2012). Design and evaluation of soft keyboards for brahmic scripts. *ACM Trans. Asian Language Inform. Process*.

Hinkle, L., Lezcano, M., and Kalita., J. (2010). Designing soft keyboards for brahmic scripts. *ICON 2010: International Conference on Natural Language Processing*, pages 191–200.

Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: a tutorial. *Reliability Engineering and System Safety*, 91:992–1007.

MacKenzie, I. and Soukoreff, R. (2002). A model of two-thumb text entry. In *Proceedings of Graphics Interface 2002*, pages 117–124. Toronto: Canadian Information Processing Society.

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1):91–139.

MacKenzie, I. S. and Zhang, S. X. (1999). The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 25–31, New York, NY, USA. ACM.

P. U. Lee, S. Z. (2004). Top-down learning strategies: Can they facilitate stylus keyboard learning? *International journal of human-computer studies*, 60(5-6):585–598.

Raynal, M. and Vigouroux, N. (2005). Genetic algorithm to generate optimized soft keyboard. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1729–1732, New York, NY, USA. ACM.

Smith, B. and Zhai, S. (2001). Optimal virtual keyboards with and without alphabetical ordering–a novice user study. In *Proceedings of the INTERACT 2001: Eight IFIP Conference On Human–Computer Interaction*, pages 92–99, Tokyo, Japan.

Zhai, S. (2004). Characterizing computer input with fitts' law parameters - the information and non-information aspects of pointing. *International Journal of Human-Computer Studies*, 61(6):791–809.

Zhai, S., Hunter, M., and Smith, B. A. (2000). The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 119–128.