In Proc. 9th Int'l Middle Eastern Multiconference on Simulation and Modeling (MESM 2008), Aug 26-28, Amman, Jordan.

TOWARD OPTIMAL ARABIC KEYBOARD LAYOUT USING GENETIC ALGORITHM

Tareq M. Malas, Sinan S. Taifour, and Gheith A. Abandah
Faculty of Engineering and Technology
The University of Jordan, Amman 11942, Jordan
E-mail: tarik.malas@gmail.com, staiii@gmail.com, abandah@ju.edu.jo

KEYWORDS
Genetic Algorithm, Optimization, Arabic Keyboard Layout.

ABSTRACT

The common Arabic keyboard layout is derived from the Arabic typewriter keyboard's layout. This layout is not optimized for typing speed and has several problems. We use a genetic algorithm to optimize the design of the Arabic keyboard layout for typing speed. In the genetic algorithm, we use a fitness function that includes into consideration key distance, finger used, and hand alternation. We also use this fitness function to evaluate the optimized layout against other layouts including the common layout. The optimized layout is relatively 35% faster than the common layout and has additional advantages.

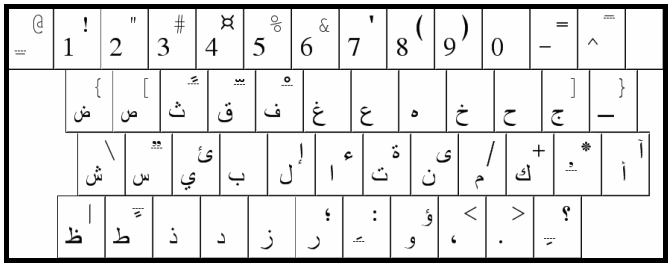
INTRODUCTION

As the keyboard is the main device for text entry in computers, it is very important to have a comfortable keyboard layout which also promotes fast text entry. In a look at the history of the English keyboard layout, we see that the standard layout (QWERTY) is not optimized for efficient typing, but is actually quite arbitrary (Noyes 1998). A more optimized keyboard layout known as the Dvorak Simplified Keyboard (DSK) has been developed(Campbell-Kelly 2005). But the QWERTY layout remains most widely used because many people were already trained on using the QWERTY, and were reluctant to change.

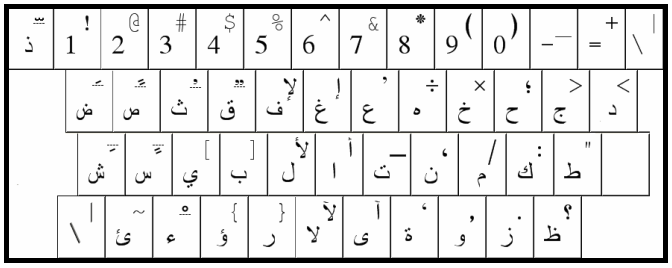
Arabic Keyboard History

The layout of the Arabic keyboard is derived from the Arabic typewriter keyboard's layout. In the 1970s and 1980s, many computer companies developed more than 20 variants for this layout (JISM 1994). These various variants had consistent allocation for some letters (namely ا ب ت ث ج ح خ س ش ص ض ع غ ف ق ك ل م ن ه ي). However, they have differences in allocating the other letters particularly the letters in the lower row (such as أ د آ ئ ر ط ء ؤ ة ز ظ و ى إ ذ). Some of the famous keyboard layouts in the late 1980s are the layouts of Microsoft Arabic Word, Apple MAC, Sakher, AMEER, ALIS, and Nafitha.

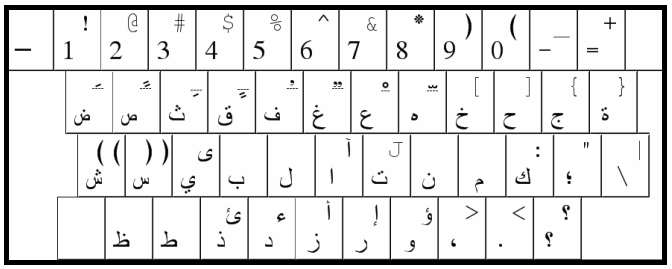
To solve this chaos created by the existence of many Arabic keyboard layouts, the Arab Standardization and Metrology Organization (ASMO) developed a standard for the Arabic keyboard layout shown in Figure 1 (ASMO 1987). This keyboard standard supports the ASMO standard for the 7-bit Arabic characters code (ASMO 1985). However, this keyboard standard was not used by the computer industry and the market adopted instead the currently used Arabic keyboard layout shown in Figure 2. This keyboard layout gained wide acceptance in PCs and servers over other layouts when Microsoft adopted it for its Arabized products. Another Arabic keyboard layout currently in use is the layout shown in Figure 3. This layout is used in Apple MAC computers.



Figures 1: ASMO 663 Arabic Keyboard Layout



Figures 2: Common Arabic Keyboard Layout



Figures 3: Apple MAC Arabic Keyboard Layout

The common Arabic keyboard layout is not optimized for performance and also has many obvious drawbacks. For example, Letter Thal "ذ" is placed in an awkward place in the top-left corner of the keyboard despite its frequent use. This letter is more frequently used than other well-positioned letters such as Letter Ghain "غ". Moreover, the two-letter

combination Lam-Alef "لا" has one key dedicated for it despite the fact that it is not as frequent as other letter combinations.

Related Work

Many researchers used various algorithms to design better keyboard layouts for other languages. A simulated annealing algorithm was used by (Light and Anderson 1993) to optimize the English keyboard. They used the travel time and the frequency of letter pairs to optimize for typing speed. An evolutionary algorithm was used by (Walker 2003) for the English keyboard to evolve it for a layout faster than the QWERTY layout by 30%. Also a genetic algorithm was used by (Goettl et al. 2005) to rearrange the English letters and some punctuation marks.

Ant Colony algorithm was used by (Wagner et al. 2003) to optimize keyboard layouts for English, French, and German text. They used ergonomic criterion to develop the desired keyboard layouts.

A new design for Hindi keyboard layout was done by (Deshwal and Deb 2003) using a genetic algorithm. They used an ergonomic criterion to optimize the layout for typing convenience. We are unaware of similar optimization work done for the Arabic keyboard layout.

Approach Used

A keyboard layout can be optimized for several characteristics such as typing speed, comfort, low error rate, and trainability. In this paper, we use a genetic algorithm to design a layout optimized for typing speed. We also evaluate the typing speed of important Arabic keyboard layouts. Although our focus is on optimizing the typing speed, the designed layout is also more comfortable than the common keyboard layout.

The rest of this paper is organized as follows. The next section describes the procedure used in this research. Section 3 presents the results obtained and analyzes these results. Finally, Section 4 presents the paper conclusions and future work.

PROCEDURE

Our procedure consists of two stages. First we collect data and find the letters' frequencies, and then we use these frequencies in a genetic algorithm to optimize the design of the keyboard layout for speed. The keyboard layout is divided into two sets: the *main set* and the *shift set*. The main set contains all keys that can be accessed without using the shift key, while the shift set contains the same keys when the shift key is pressed.

Finding Frequencies of the Arabic Characters

The used optimization requires information about the Arabic characters and character pairs' frequencies. The data we used to find these frequencies was taken from the Arabic Wikipedia (http://ar.wikipedia.org), which contains articles from almost all fields. So the related character frequencies are not biased to any particular field. This site contains more than 10,000 articles, mostly well-written. These articles can be conveniently and freely downloaded in a single XML file (http://en.wikipedia.org/wiki/Wikipedia:Database_download). The Arabic Wikipedia was downloaded, parsed, and analyzed to find frequency information.

The single XML file, around 200 MB in size, contains all the articles structured with some meta information. The first step was to extract the articles and save them in separate files for analysis. The meta information was kept in an accessible format for latter usage. Due to the large size of this XML file, it is not possible to use a parsing tool that loads the entire file into the memory, such as the Document Object Model (DOM). Instead, a serial event-driven method must be used. We use the Simple API for XML (SAX), which is available in the Python programming language.

The next step was to remove all wikitext. Wikipedia articles are formatted using a special syntax called wikitext (http://en.wikipedia.org/wiki/Wikitext). For removing the wikitext, we used the Regular Expressions (RegExp) tool, which is also available in Python. The regular expressions describe patterns of text that can be matched against longer strings of text or used for substitution. Most of the wikitext can be fully described using rather simple RegExp patterns.

The last step was to count occurrences of characters and character pairs and generate result files. In this step, the user is prompted every time a new unknown character is encountered. The user is given three options: taking the character into consideration (for Arabic characters), considering it as a special character (for special characters and white space), and ignoring it (for non-Arabic characters). The special characters are given special status to enable finding word boundaries. This allows finding the probabilities of words starting or ending with particular Arabic characters.

Genetic Algorithm for Design Optimization

A genetic Algorithm is a guided random search technique inspired by evolutionary Biology (Goldberg 1989). It involves a population of candidate solutions to a problem, called individuals, which live, reproduce, and die based on their fitness relative to the rest of the population. The genetic algorithm consists of five phases: initialization, evaluation, selection, reproduction, and competition. A typical genetic algorithm requires two definitions:

1. Genetic representation of the solution domain
2. Fitness function to evaluate the solution domain

The optimization performed by a genetic algorithm has the ability to search a large space for a solution which is close to the optimum with relatively short time. The space of possible keyboard layouts is the factorial of 33 keys (8.68×10^{36}). Evaluating all these layouts is infeasible as it takes very long time.

The genetic algorithm is applied on the main set of the keyboard, where we allocated the most frequent Arabic

letters and symbols. The representation of the individuals is done using a 33-element vector, which corresponds to the characters allocated on the 33 keys of the main set.

The used fitness function is the typing time using the keyboard layout. This fitness function takes into consideration attributes that give short typing time such as hand alternation and using the home row. It also take into consideration attributes that give long typing time such as typing two consecutive letters with the same finger. Smaller fitness values are better. The used fitness is the aggregate of the products of all letter pair frequencies f_i and the corresponding letter pair typing times t_i, formally

$$Fitness = \sum_i f_i * t_i \ . \qquad (1)$$

The pair typing times are found using a linear regression model (Hiraga et al. 1980). This model was built after measuring the time intervals between key presses. We used the following model to find the key pair typing time.

$$t_i = 185.8 - 40.0h + 18.3r - 11.0f + 0.514R + 1.07F \quad (2)$$

Table 1 describes the attributes used in this model and the possible values of these attributes. Letters mapped in the shift set are given the maximum pair typing time of the main set pair typing times.

Table 1: Attributes Affective Key Pair Typing Times

Attribute	Symbol	Values
Hand Transition	h	0: same hand, 1: alternate hand
Row Transition	r	Number of rows moved across in the same hand transition
Finger Transition	f	The distance of finger columns in the same hand
Row Weight	R	Linear sum of weights for each row position, where the weights given are 1, 2, and 3 for home, upper, and bottom rows, respectively.
Finger Weight	F	Linear sum of weights on each of the finger positions of the key pair, where the weights given are 4.5, 4.5, 1, 2, and 3 from the outer column inwards.

The following paragraphs describe our implementation of the used genetic algorithm.

Initialization
Initially 2000 individuals (layouts) were randomly generated to form an initial population. The population was generated randomly so that the algorithm is not biased. This helps to avoid convergence to a local minimum. The randomness gives disperse layouts that cover wide regions of the space.

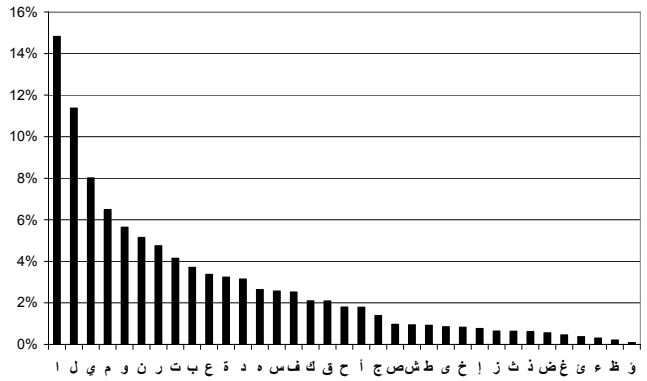
Selection
During each successive generation, a proportion of the current population is selected to breed a new generation. The selected individuals "parents" comprise the best individuals and randomly selected individuals. The randomly selected individuals ensure maintaining high diversity of the population. Thus preventing the algorithm from getting stuck in a local minimum.

Reproduction
The reproduction is usually made by crossover and/or mutation from the parents. The produced individuals "children" share many of the characteristics of their "parents". We only use mutation, because there is no direct method to apply crossover. The mutation is applied by swapping the locations of a number of the parent's keys. We breed more children than we need so that we can select the best children to compete with the original population. The children who have better fitness replace some of the original population. This process gives the next generation population, which is different from the initial generation. Generally the average fitness increases by this process.

Termination
The termination is performed manually. The program saves logs of the population every 200 generations. We monitor these logs seeking satisfactory results and convergence. The program was stopped after 160,000 generations. This took 32 hours of execution time on a 2.2 GHz Core 2 Duo notebook.

RESULTS AND ANALYSIS

Figure 4 shows the frequencies for Arabic letters. The most frequent three letters (Alef "ا", Lam "ل", and Yeh "ي") have aggregate frequency of 34%. The most frequent 32 letters in addition to the period were selected for allocation on the main set. The three least frequent letters and the special characters are allocated in the shift set.



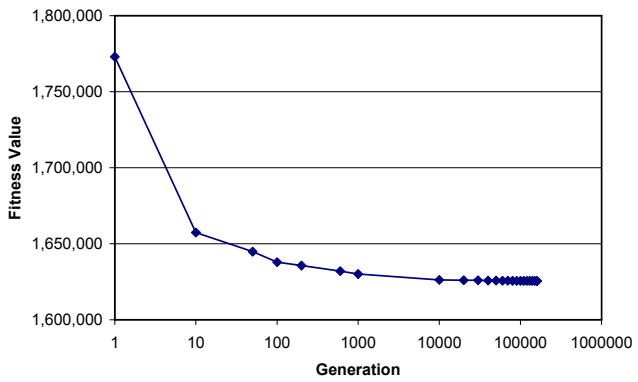
Figures 4: Frequency of the Arabic Letters

Table 2 shows the frequencies of letter pairs. Columns are for the first letter and rows are for the second letter. This table shows that the frequency of the frequent pair Alef followed by Lam "ال" is 8.3%, Lam-Meem "لم" is 1.6%, Yeh-Teh Marbuta "ية" is 1.3%, and Lam-Alef "لا" is 1.3%.

Table 2: Frequency of Arabic Letter Pairs (First letter on the column, second letter on the row)

	ا	ل	ي	م	و	ن	ر	ت	ب	ع	ة	د	ه	س	ف	ك	ق	ح	أ	ج	ص	ش	.	ط	ى	خ	إ	ز	ث	ذ	ض	غ	ئ
ا	0.0	1.2	0.7	1.0	1.1	0.6	0.8	0.3	0.7	0.6	0.0	0.4	0.7	0.4	0.2	0.5	0.4	0.3	0.0	0.3	0.2	0.2	0.0	0.2	0.0	0.1	0.0	0.1	0.1	0.2	0.1	0.1	0.0
ل	8.3	0.5	0.4	0.3	0.7	0.0	0.0	0.2	0.2	0.7	0.0	0.1	0.1	0.2	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.1	0.0	0.2	0.3	0.0	0.1	0.1	0.0	0.0	0.1
ي	0.3	0.9	0.1	0.5	0.4	0.6	0.8	0.3	0.6	0.2	0.0	0.6	0.2	0.4	1.3	0.2	0.2	0.3	0.1	0.2	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.2	0.1	0.1	0.1	0.1	0.2
م	0.8	1.6	0.3	0.1	0.4	0.1	0.1	0.3	0.1	0.2	0.0	0.2	0.3	0.2	0.0	0.2	0.0	0.2	0.2	0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
و	0.2	0.4	0.3	0.4	0.0	0.2	0.3	0.3	0.2	0.1	0.0	0.3	0.2	0.2	0.1	0.2	0.2	0.1	0.3	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
ن	1.2	0.3	0.9	1.1	0.5	0.0	0.1	0.1	0.3	0.3	0.0	0.1	0.1	0.1	0.0	0.2	0.0	0.0	0.3	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0
ر	0.8	0.2	0.6	0.3	0.5	0.0	0.0	0.3	0.4	0.3	0.0	0.2	0.1	0.1	0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.2	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.0
ت	0.9	0.7	0.3	0.2	0.2	0.3	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.4	0.1	0.2	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ب	0.4	0.4	0.2	0.1	0.2	0.1	0.3	0.2	0.0	0.2	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ع	0.3	0.6	0.2	0.4	0.2	0.0	0.1	0.2	0.3	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ة	0.1	0.3	1.3	0.2	0.0	0.2	0.4	0.0	0.1	0.2	0.0	0.3	0.0	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
د	0.5	0.3	0.4	0.3	0.2	0.2	0.1	0.1	0.2	0.3	0.0	0.1	0.1	0.0	0.0	0.0	0.3	0.3	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ه	0.2	0.4	0.3	0.1	0.1	0.3	0.1	0.3	0.2	0.1	0.0	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
س	0.4	0.4	0.3	0.3	0.2	0.2	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
ف	0.2	0.3	0.1	0.0	0.2	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ك	0.2	0.5	0.2	0.1	0.1	0.0	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ق	0.2	0.4	0.2	0.1	0.2	0.1	0.1	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ح	0.2	0.4	0.1	0.2	0.1	0.0	0.1	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
أ	0.0	0.6	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ج	0.2	0.3	0.1	0.2	0.2	0.1	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ص	0.1	0.2	0.0	0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ش	0.1	0.3	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
.	0.1	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ط	0.1	0.1	0.1	0.0	0.1	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ى	0.0	0.7	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
خ	0.1	0.2	0.1	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
إ	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ز	0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ث	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ذ	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ض	0.1	0.0	0.1	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
غ	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ئ	0.3	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 5 shows the convergence curve of the genetic algorithm. This curve shows the best fitness value in a generation. Note the log scale of the generation number. This curve shows that the algorithm stabilizes fast after around 1000 generations (10 minutes of execution).



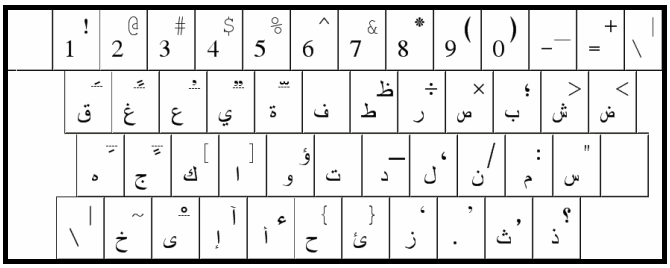
Figures 5: Convergence of the Genetic Algorithm

Figure 6 shows the optimized keyboard layout as found by the genetic algorithm. This layout has the common letters located in the comfortable home row and gives good hand alternation. The following can be noticed from the main set layout:

1. The most frequent key-pair Alef-Lam "ال" are under different hands, on the middle row, and under the strongest fingers (the index and the middle fingers).
2. Most of the frequent letters are placed in the middle row and/or using the strongest fingers, as in the case of Yeh "ي", Waw "و", and Teh "ت".
3. The least frequent letters are placed in the bottom and top rows and most of them are under the pinky finger (the weakest finger), as in the case of Thal "ذ", Dad "ض", and Ghain "غ".

No algorithm was used to optimize the shift set because the letters in this set are infrequently used in Arabic typing. The shown arrangement was done manually to keep the layout

similar to the common layout. However, some modifications were necessary to accommodate for the additional letters not included in the main set. The letters which were moved to the shift set are put on keys of similar letters in the main set; where users expect to find them. For example the pair Tah "ط" and Zah "ظ", and the pair Alef with Hamza Above "أ" and Hamza "ء" are put on same keys.



Figures 6: Arabic Keyboard Layout Optimized for Speed

A comparison is made between the optimized keyboard layout and some other layouts by applying the fitness function. Table 3 presents the results of this comparison. The worst layout was found by running the genetic algorithm with the fitness function complemented. The random layout was built by generating a random layout. We note that the common layout is better than the ASMO standard layout. The main reason for this difference is mapping the moderately used letter Teh Marbuta "ة" to the shift set in the ASMO layout. Taking the interval between the worst and optimized layouts as a reference base, the common layout is 6% better than the ASMO standard and the optimized layout is 35% better than the common layout.

Table 3: Comparison for Arabic Keyboard Layouts

Layout	Fitness Value	Relative Performance
Worst	1,942,000	0%
Random	1,773,000	53%
ASMO 663	1,756,100	59%
Apple MAC	1,740,600	64%
Common	1,735,000	65%
Optimized	1,637,000	100%

CONCLUSIONS

In this paper we have used a genetic algorithm to optimize the layout of the Arabic keyboard. The optimized layout gives 35% relative increase in the typing speed. Moreover, the optimized layout solves some of the problems found in the common Arabic keyboard layout. This layout gives better allocation for the frequently used letters such as Thal "ذ", and does not dedicate precious key locations for two-letter key combinations such as Lam-Alef "لا". We have also found that the common layout by is 6% better than the ASMO 663 layout.

For future work, we intend to train volunteers on using the optimized layout to obtain experimental data about the optimized layout's performance, comfort, and trainability. We can collect timing data from these volunteers and use the letter pair times to adjust the model used in Equation (2). Moreover, we intend to try to take more factors in the optimization such as typist comfort and error rate.

REFERENCES

Arab Standardization and Metrology Organization (ASMO). 1985. *Standard 449: 7-Bit Arabic Code*. Arab League, Cairo, Egypt.

Arab Standardization and Metrology Organization (ASMO). 1987. *Standard 663: Arabic Keyboard Layout*. Arab League, Cairo, Egypt.

Campbell-Kelly M. 2005. The User-friendly Typewriter. *The Rutherford Journal*. Retrieved Jul 20, 2008 from http://www.rutherfordjournal.org/article010105.html

Deshwal P.; and Deb K. 2003. "Design of an Optimal Hindi Keyboard for Convenient and Efficient Use." Technical report KanGAL 2003005, Indian Institute of Technology, Kanpur.

Goettl J.; Brugh A.; and Julstrom B. 2005. "Call Me E-Mail: Arranging the Keyboard with a Permutation-Coded Genetic Algorithm." In *Proceedings of the ACM Symposium on Applied Computing* (Santa Fe, New Mexico, USA, Mar), 947-951.

Goldberg E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman, Boston, MA.

Hiraga Y.; Ono Y.; and Yamada-Hisao. 1980. "An Analysis of the Standard English Keyboard." In *Proc. 8th Conf. on Computational Linguistics* (Tokyo, Japan). 242-248.

Jordan Institution for Standards and Metrology (JISM). 1994. *Minutes of Meeting of the Arabic Letter Committee*. Mar 10, 1994, Amman, Jordan.

Light L.; and Anderson P. 1993. "Typewriter Keyboards via Simulated Annealing." *AI Expert*, (Sep).

Noyes, J. 1998. "QWERTY-The Immortal Keyboard." *Computing & Control Engineering Journal* 9, No.3 (Jun), 117-122.

Wagner M.; Yannou B.; Kehl S.; Feillet D.; and Eggers J. 2003. "Ergonomic Modeling and Optimization of the Keyboard Arrangement with an Ant Colony Algorithm." *European Journal of Operations Research* 148, No.3 (Aug), 672–686

Walker C. 2003. "Evolving a More Optimal Keyboard." Technical Report (Dec).

AUTHOR BIOGRAPHY

Tareq Malas is a bachelor student in the Computer Engineering Department at the University of Jordan. He is expected to graduate in Sep 2008. He has won a full scholarship for pursuing postgraduate studies in King Abdullah University of Science and Technology (KAUST).

Sinan Taifour is a bachelor student in the Electrical Engineering Department at the University of Jordan. He is expected to graduate in Feb 2009.

Gheith Abandah holds BS degree in Electrical Engineering from the University of Jordan (1985) and MS and PhD degrees in Computer Engineering from the University of Michigan (1995 and 1998). He is an Assistant Professor of Computer Engineering in the University of Jordan since 1998, and department chair since 2006. His areas of research are Computer Arabization, Arabic Optical Character Recognition, Computer Architecture, Parallel Processing, and Software Development.