Genetic Approach for a Flexible Cell Phone Keypad with Reduced Keystrokes and Key Jamming for Better Human Technology Interaction

Sabirat Rubya, Samiul Monir, Hasan Shahid Ferdous
Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka - 1000, Bangladesh.
Email: sabiratbittamoni@gmail.com, samiulmonir@gmail.com, hasan.shahid@csebuet.org

***Abstract*— Despite the ever-increasing popularity of mobile devices, text entry on such devices is becoming more of a challenge. Problems with traditional keypads primarily lie with placement of the letters alphabetically on the keys. This configuration, though very easy to remember, requires higher number of keystrokes and key jamming along with a limitation in flexibility of finger movement, as many frequent letters are not easily reachable by the thumb. Besides, users have to move their thumb extensively which also increases typing time and provides less comfort. in this paper, we have considered these issues and proposed a novel solution to the problems identified. The proposed solution focuses on both the structure of suitable human finger movements and ordering of the letters on the keys. We also took the basic Human Computer Interaction principles and general issues into consideration. Designs were found that have performance surpassing the traditional keypads, while maintaining better usability. Performance measurement through simulations of our proposed system have shown a rapid lessening of key jamming by up to 51.38 percent, improvements in flexibility of finger movement by up to 7.31 percent, number of keystrokes by up to 29.99 percent, and reduction of total distance while changing keys by up to 2.04 percent. In summary, this work represents an improved keypad layout for text entry on cell phones and other similar devices.**

***Index Terms*— Cell Phone, Finger Movement, Frequency of Alphabets, Human Factors, Keypad, and Key Jamming.**

I. Introduction

The basic concept of cellular phones began in 1947 with the realization that using small cells with frequency reuse could increase the traffic capacity of mobile phones substantially. Today the usage of cell phones has emerged drastically and billions of people around the world use these devices in their everyday life for communications. While advancing technologies have enabled their use with many types of multimedia content in cell phones, a lot of information is still being entered and processed in text format. Therefore, text entry remains an essential part of human-computer interaction (HCI) with mobile devices. Typical mobile applications such as short text messaging, note taking, appointments and alerts tracking, address or phone number directories, etc. include more or less text input options. Although these applications usually do not require extensive amounts of text input, a certain level of text entry is inevitable. Furthermore, mobile text entry methods may be used under circumstances that are often quite different from those where desktop computers are used, for instance, where socially inappropriate or while moving [1]. In addition, the more casual, unstructured, and hurried text entered into mobile devices can be very different from that used in more formal writing or in speech and regarding this, many approaches to text entry methods are used [2].

SMS, the mostly used application using text-entry feature is the common term for sending short (maximum of 160 characters including spaces) text messages using mobile phones. Text messaging has been a tremendous success in many countries, including Asian countries like Bangladesh, India, Singapore, and Malaysia. Malaysians, for example, were found to have sent 11.7 billion messages in the first three months of 2007, compared to only 7.4 billion in 2006 [3], [4]. SMS allows text messages to be sent and received to and from mobile telephones. The text can comprise words, or numbers, or an alphanumeric combination. The messages are delivered immediately (or when the phone is turned on) and can be reviewed or stored in the phone for as long as one wishes. It is ideal for sending vital information quickly and accurately. Today SMS messaging and chatting have become basic needs for our daily life by being a quick, easy, and cheap way to communicate with anyone, anywhere, and at anytime.

To use SMS and many other functionalities of a cell phone, we need the help of a keypad for text entry. There exist a number of variations on these keypads such as Multi-tap keypad, QWERTY keypad, and Touch Screen keypads. Multi-tap keypads use a key for several alphabets and users may enter an alphabet by tapping keys a particular number of times. This keypad is the

This paper is based on and extension of the paper "A Flexible Keypad Reducing Keystrokes and Key Jamming for Cell Phones," by S. Rubya, S. Monir, and H. S. Ferdous, which appeared in the Proceedings of the 14^{th} International Conference on Computer and Information Technology (ICCIT), 2011 © 2011 IEEE.

This is a research work of the Human Technology Interaction (HTI) research group of the Dept. of CSE, BUET.

Figure 1. Multi-tap keypad.

Figure 2. QWERTY keypad.

most commonly used one, as cell phones with this type of keypads are quite cheap. QWERTY and Touch Screen keypads can have individual keys for both the alphabets as well as numerals. Often they are more expensive than the traditional ones.

Multi-tap keypads are widely used around the world. In this system, the user presses the key multiple times to make a letter selection. For example, the key *2* is assigned with the letters *A*, *B*, and *C*, thus if a user wants to enter a *C*, then he/she has to press the key three times (222) successively as *C* is the third letter placed on the key. The process of typing becomes more complicated when the intended consecutive letters are placed on the same key. For example, to text the word *cab* the user must press the *2* key using the following pattern: 222 (pause) 2 (pause) 22. To select the correct letter on the key, the user must pause to determine the correct letter. This phenomenon is known as *Key Jamming*. Most of the mobile phones employ a time-out process in which the user waits for a specified time (typically one to two seconds) before attempting to enter the next letter. Key jamming is one of the primary reasons for which multi-tap keypad is often criticized for being slow [5]. .

While designing this multi-tap keypad, it was assumed that the probability of occurrence of all letters will be the same and hence, the letters are assigned to the keys in alphabetic order. As a consequence, a frequently occurring letter may remain in a key whose position is not flexible for the fingers to reach and may require multiple key presses. Typing with multi-tap keypad layouts also arise problems for physically challenged people or people having movement problems in their thumb [6]. Two letters that frequently occur successively are assigned to distant keys in many cases. Therefore, one has to move his thumb extensively while typing. This also imposes extra pressure

Figure 3. Touch screen keypad.

on the thumb, thus increasing the typing time.

Innovative solution to these problems, focusing on both physical keypad designs and different text entry methods is introduced in this work. We seek to devise a new multi-tap keypad layout that requires less keystroke, key jamming, and less movement of thumb with increasing flexibility. To make a trade-off among all these factors we have represented the system as an optimization problem and proposed a genetic approach to solve it.

The rest of the paper is organized as follows: Section II discusses related work. Section III provides an overview of the performance criteria based on which this new method is designed. We review our previous approach and some insight on it in Section IV and Section V discusses our current approach in detail. Section VI then describes the data sets and experimental evaluation of our technique. Finally, Section VIII concludes with directions to future works discussed in Section VII.

II. Related Works

Designing a more flexible keypad for cell phones has been an interesting research topic for years, and many approaches are available. Considerable attention has been devoted to improve the traditional keypad system. For example, the flexibility of thumb movement while typing has been considered in [7] and [8] which also reduce key jamming to some extent. While designing this layout, the keys were ordered from an anatomical point of view, according to thumbs flexibility and pressure to reach them. Afterwards, the letters of the alphabet were arranged based on the frequency of the letters.

A different approach was followed in [9] and [10]. Their proposed architecture kept vowels mostly in the first position of the keys. They took into account the fact that probability of occurrence of vowels is higher than that of consonants. Thus, it could reduce total number of key pressing but no consideration was taken to reduce key jamming or thumb movement. The Human Computer Interaction suggests that the design interfaces should not overload user's memories and they 'should promote recognition rather than recall' [11]. According to this, the traditional layout can increase human technology interaction, as it is easy to remember. Most of the layouts considering performance improvement have failed to keep the letters alphabetically on keys. But in [9], the letters were sorted alphabetically in their corresponding keys.

Different layouts for other languages than English are also proposed from time to time to increase the convenience of the users. For example, Pathan *et al.*, in their work [8] designed an interactive keypad for Bangla letters based on phonetics. They assigned Bangla letters to those keys where English letters of same type of phonetics lie. This approach requires more keystrokes per letter typed. Sharmeen *et al.* [12] also planned to design a flexible layout for Bangla keypad based on letter frequency.

Another problem of using mobile keypad is the shrinking device sizes that require unique input modalities and interaction techniques. In attempting to resolve this issue, researchers have found that dictionary-based predictive disambiguation text entry methods are fairly efficient for text entry on mobile phones. This type of solution is proposed in [13] and [14], where a set of keypad designs are optimized under the constraint of keeping characters in alphabetical order across keys. Again, design of a flexible layout is provided in [15] where users have to move their thumb a little for text input using only five keys.

Wang *et al.* proposed to redesign the user interface of mobile phone based on the motion characteristics of thumb and index finger [25]. They divided the regions of human palm to different areas based on comfort during grasping and typing. Hence, they devised a novel layout that ensured less movement of the zones of discomfort. Their designed layout could ease the heavy workload on thumb as well as better fit the hand with the devices when grasping.

Another near optimal keypad layout was proposed in [24] using genetic algorithm. They combined three criteria to determine layout efficiency: strokes per character, delay due to consecutive use of the same thumb and key jamming. A weighted sum fitness function was defined considering the above criteria which penalized consecutive use of same thumb or same key and repeated use of a key. But the amount of thumb movement was not taken into account as a factor of improving the typing speed.

A notable motivation of our previous work [16] came from the confinement of the discussed layouts as none of them have addressed together all the problems prevailing in traditional keypad. The algorithm that was described in this paper was inspired mainly to satisfy three design goals to increase the typing speed with mobile keypad. These include:

1) The flexibility of thumb movement should be increased.
2) Total number of key pressing should be decreased as much as possible.
3) The algorithm should be capable of reducing key jamming considerably.

Following these three factors, a new keypad layout was designed where ordering and position of letters are different from the traditional keypad. But use of this keypad will reduce the total typing time as well as enhance the flexibility of finger movement.

This paper significantly extends our previous work on devising flexible keypad layout [16] in several ways. First, we consider the total distance traveled by thumb while changing keys to type different letters. Therefore, the current version is more concerned about better human technology interaction than the previous one. This time we also modify the previous algorithm to a meta-heuristic genetic algorithm, making the solution more appropriate. Second, it adds considerable amount of mathematical analysis over the previous version. Third, evaluation is done in a more realistic way and the correctness of the algorithm is assured with t-tests. Finally, we compare our proposed layout against traditional keypad, enhanced keypad by Azad *et al.* [17] , T9 Dictionary based keypad, and our previously proposed keypad.

III. Performance Criteria

The algorithm described in this paper is inspired mainly to satisfy four design goals to increase the typing speed with mobile keypad. These include:

1) The flexibility of thumb movement should be increased.
2) Total number of key pressing should be decreased as much as possible.
3) The algorithm should be capable of reducing key jamming considerably.
4) The average distance covered by a thumb to type a text should be reduced.

To understand and compare our proposed keypad, we may require having a clear understanding of the performance criteria of a cellular keypad. We identify the following factors when we try to evaluate the performance of a keypad.

A. Number of Keystrokes:

A keystroke represents a key press action on the keypad or other equivalent input device. The number of keystrokes is counted by the number of key presses while typing. For instance, to type 'MONKEY' with a traditional keypad, the total keystroke will be 13 (1 + 3 + 2 + 2 + 2 + 3). We will experiment and compare our proposed keypad with other keypads to see whether our proposed one can reduce keystrokes required to type a message.

B. Key Jamming:

When two consecutive letters to be written are in the same key of a keypad, then after typing the first letter, one has to wait for a while to type the next. This phenomenon is called key jamming and this increases the total typing time required. For example, to type MONKEY, key jamming occurs twice (between the pairs M-O and O-N) in traditional keypad. One of our goals in this paper is to reduce key jamming while typing.

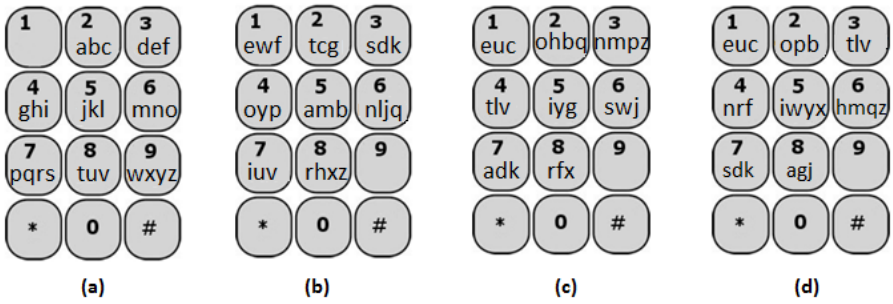

Figure 4. (a) Traditional keypad, (b) Proposed keypad in [17], (c) Our previous proposed keypad [16], (d) Our proposed keypad.

C. Flexibility of Thumb Movement:

This is the ease of moving the thumb to a particular key. A key is flexible when it is reachable by the finger (thumb) without pressurizing the physical structure and the internal joints of the thumb much. Sharmeen *et al.* [12] discussed about this property in details. They identified the two types of thumb movement, Flexion and Extension, required to press a key and stated that the pressure in the interphalangeal joint of a thumb increases with the decrease in the joint angles. So forward direction movements are convenient while lateral movements of thumb create extra stress on the user [18]–[20]. Based on this principle, the order of the flexibility of the keys in a keypad is: $1>2>4>5>7>3>6>8>9$. We will consider this property while designing our proposed keypad.

D. Total Distance Covered by the Thumb:

Another criterion for the performance of mobile keypads can be the amount of finger movement that a person requires on an average [21]. It depends on the distance between the keys on which consecutive letters are placed. Taking traditional keypad as an example, typing 'hey' will need more distance to travel than typing 'cat'. In case of the word 'cat', user changes key once from key '2' to key '8' to type 'a' and 't'. To type 'hey', he has to change key twice, from key '4' to key '3' and again from key '3' to key '9'. It is obvious that distance between key '2' and key '8' is more than summation of distance between '4' and '3' and '3' and '9'. As writing SMS in hurry is common, layout that can reduce total distance traveled by thumb will be more acceptable. We add this property to the other three for designing a more user friendly and flexible keypad. The process of calculating distance between keys will be discussed in Section V.

IV. Our Previously Proposed System

We have considered a one-handed and one-finger multi-tap keypad layout where all the typing will be done using only the thumb of the right hand. Typing blank spaces and capitalization of letters are not taken under consideration. While designing our keypad layout, we have kept 'space' in key '0', which is followed by most of the keypads. As occurrence of symbols and punctuation marks (**,** , **.** , **;** , @ , **!** , **?** , (,) etc) is much less frequent than the letters, we have assigned them to the most inconvenient key to press [12], i.e., key 9. The rest of the 8 numeric keys available are used to type the 26 alphabets. Hence, six keys of them will get three letters each and the other two keys will be mapped to four letters each.

To work with, first we classify the letters into three groups: Group-A, Group-B, and Group-C according to descending value of their frequency. In other words, the topmost 8 frequent letters are contained in Group-A, the next 8 higher frequency letters go to Group B and remaining 10 letters are assigned to Group-C. This classification is required to prevent a combination with two or more highly frequent letters to be assigned to the same key.

We know that the performance of any decision-making algorithm can be measured by the total cost or the total utility. An algorithm is considered to provide a better result if either total utility can be increased or total cost can be decreased. In our proposed algorithm, we have intended to increase the utility value. To make decisions like whether a particular combination of alphabets should be assigned to a key or not, we have given emphasis on assigning a good set of utility value according to the following factors:

1) Flexibility of a key according to movement of the thumb.
2) Letter and combination of letters according to the probability of its occurrence.

The 26 letters of English alphabet and all possible two letter combinations generated from them need to have a utility value, which is explained later. For the time being, we assume that the utility of a single letter will be its frequency. In a similar way, the utility of a combination will be measured from the frequency of that combination.

The proposed algorithm basically works in a greedy approach. We aim to increase the total utility of the algorithm by assigning higher utility combinations to more convenient keys to reach. The utility of a three-letter combination assigned to a key is dependent on the following two factors:

1) The frequency or probability of occurrence of each letter in that combination.
2) The frequency or probability of occurrence of all possible unordered combinations of any two letters taken from that three-letter combination.

For example, the utility of the combination abc should be:

freq. of a	+	freq. of b	+	freq. of c
-freq. of ab	-	freq. of ba	-	freq. of ba
-freq. of cb	-	freq. of ca	-	freq. of ac

To understand why the frequency of all possible two-letter combinations are deducted, we need to consider that when two letters from a combination are consecutive in a word, key jamming will occur, thus increasing the wasted time. Therefore, the utility of a combination should decrease with increasing probability of consecutive occurrence of any two letters from the combination, as they would be contained in the same key. Another thing to notice is that, frequency of three-letter combinations does not contribute to the utility value as only pair-wise consecutive letters can cause key jamming.

Now we give insight to the detailed description of the proposed system. At first the utility of all the possible three-letter combinations are calculated using the method stated above. The sample database for this purpose has been selected from a web server where there is a list of SMS used by different people. We have only considered ordered combinations with no repeated letters, as a letter has to be assigned only once. So the combination sequence (abc, acd, ade, ... ,ayz, abd, abe,.....,wxy, wxz, xyz) includes ${}^{26}C_3$ combinations. The combinations with higher utilities mean that the letters in those combinations are more frequent and jamming between them is less. So the only remaining concern is to assign the combinations with higher utilities to the more flexible keys so that the more frequent letters lie on the keys that are more flexible to be used by the users. To do this, we sort the ${}^{26}C_3$ combinations according to non-ascending value of their utilities and the eight keys according to non-ascending value of their flexibilities. Then we proceed by placing the highest utility combination to the most convenient key, the second highest combination to the second most convenient key and so on until all eight keys are assigned to different combinations. We have to be very careful about a number of constraints at each step of assignment. While working with a particular three-letter combination, we need to be assured that the letters constituting the combination come from different classes (Group - A, B, and C). This means no two letters from same group can be assigned to the same key. The logic behind this constraint is that if two letters from the same group are assigned to a key, one of them has to be in the second position in that key. So the total keystrokes will increase for the highly frequent letters in that key. In addition to this, when a combination is assigned to a key, we do not need to consider other combinations having alphabets from that particular combination.

The above process places 24 letters in 8 keys having three letters each. Remaining two letters are combined to any of the 8 keys by repeating the previous calculation of key jamming for four-letter combinations (the three letters already assigned to a key and another letter from those two letters) and thus finding suitable keys for these two letters. Finally, we have to order the letters of a combination in a proper way. For all the combinations assigned to the keys, we order them in non-ascending value of their frequencies. For instance, if the key '5' gets the combination 'abc' and in our pre-calculation we find that, frequency of 'a', 'b', 'c' are 20, 15 and 18 respectively, then we place 'abc' in key '5' in the order 'acb'. This will assure less keystroke for typing highly frequent letters. Our proposed algorithm can be summarized with a pseudocode in Fig. 5.

In this new layout, users will find the frequently used letters arranged in the flexible keys for the thumb to reach and require much less keystrokes. The waiting time between typing two consecutive letters will also be lessened as letters are assigned considering the key-jamming problem. Hence, this new keypad layout will be very much convenient and user friendly in comparison to the traditional keypad. In the next section we will modify it further so that the users require less finger movement while keeping the existing benefits.

V. The New Proposal

We keep our previous decision about placing the punctuation characters at key '9' and using key '1' to '8' for the 26 alphabets grouped in three groups. The proposed new algorithm is basically a genetic algorithm for function optimization. We aim to increase a total fitness value that depends on four different factors: number of keystrokes, amount of key jamming, flexibility of thumb movement, and total distance covered by the thumb. The total fitness value should be such that improvement of the four features is fulfilled.

A. Number of Keystrokes:

As already discussed, our aim is to decrease the total number of keystrokes so that typing takes less time. Number of keystrokes decrease only when more frequent letters are assigned in the earlier position of a key. This ensures only one or two keystrokes for these letters. Again, as per our grouping of alphabet, two or more letters from the same group should not be allocated to the same key to reduce key jamming. If two letters of Group-A are on same key, then one of these more frequently occurring letters must be in second, third or fourth position of the key which violates the rule just stated. So, the fitness value for number of keystrokes can be represented mathematically by the following equation:

$$f_{keystk} = \sum_{l \in L}(freq_l - (C_{keystk} \cdot pos_l) - \sum_{k \in K}(C_{keystk} \cdot N_k)) \tag{1}$$

Here, C_{keystk} is a constant which is tuned based on the size of sample database, $freq_l$ and pos_l denotes frequency and position of the letter l on a key and N_k is the number of letters from same group on key k. L is the set of all English letters and K is the set of all keys.

```
Function generateKeypad(threeComb,allComb) returns keypad

Input:      threeComb- a list of all possible three letter combination.
            allComb -  a list of all possible one and two letter
                              combination.

Output:     keypad – a final keypad for our proposed system.

Local var:  utilitySet – a list of combined set of all possible three
letter                                combinations and their respective
utility value.
            selectedComb – a particular combination.

Loop do
      pick a combination from threeComb
      calculate utility value for this combination
      add the combination and its utility value to utilitySet
end loop

Sort utilitySet according to non-ascending order

loop do until keypad full

      pick a combination from utilitySet
      if any letter of the combination is not exists in any of the keypad
            combination and no other permutation is exists then

            add the combination to keypad
      end if
end loop
```

Figure 5. Algorithm for our previously proposed keypad.

B. Amount of Key Jamming:

Key jamming depends on the frequency of all possible combinations formed by two letters. If the frequency of a combination of two letters that are entailed on the same key is high, then key jamming will increase. In this case, key jamming will occur every time the two letters come consecutively while typing. Therefore, the fitness value for key jamming can be expressed by negating the sum of frequencies of this type of combinations, as in Eq. 2

$$f_{keyjamming} = -\sum_{k \in K} freq_{l1,l2}. \tag{2}$$

Where $l1$ and $l2$ are two letters on the same key. For instance, the above calculation for the combination 'abc' on a particular key will contain the combinations - 'ab', 'ba', 'bc', 'cb', 'ca', and 'ac'.

C. Flexibility of Thumb Movement:

For calculating $f_{flexibility}$ or the fitness for flexibility of typing, we allot a value $keyval_k$ for each key k. The more flexible a key is for the thumb to reach, the higher is the value of $keyval_k$ for that key. According to this policy, key '1' has $keyval_1 = 9$, key '4' has $keyval_4 = 3$, up to key '9' has $keyval_9 = 1$. As the target of our proposed algorithm is to increase flexibility of thumb, the frequent letters should be placed on more convenient keys. The significance of $keyval_k$ is that this value is multiplied by the sum of frequencies of all the letters on that key. Thus, when a convenient key having higher value of $keyval_k$ will contain more frequent letters, the fitness of flexibility for the key will increase. This can be represented by Eq. 3.

Figure 6. Distance measurement between keys.

$$f_{flexibility} = \sum_{k \in K} (freq^k_{l1} + \cdots + freq^k_{l3} + [freq^k_{l4}]) \cdot keyval_k \tag{3}$$

Here, the squared bracket enclosing $freq^k_{l4}$ denotes optional presence of $l4$ on k as six of the eight keys will hold three letters on them.

D. Total Distance Covered by the Thumb:

We assume that the keys on a keypad are all square in size. In addition, the distance between a key and each of the keys laid on immediately left, right, upward and downward of it is one unit. Distance of the keys which are placed diagonally from a key will be measured using Pythagorean Theorem. Fig. 6 gives a clear idea about the distance between two keys. We want to decrease the total distance that a thumb has to travel to type a message on average, because this can help people with movement problems such as Parkinson's disease. We aim to design a layout that will be comfortable for them, as they have to move their thumb as less as possible not hampering the other criteria. To do this, we have to ensure that two consecutive frequent letters must be on two keys that have less distance between them. Therefore, the fitness value for distance $f_{distance}$ will be the difference of frequency of two letter combinations and the distance between the keys containing them multiplied by $C_{distance}$, a constant based on the size of the sample database.

$$f_{distance} = \sum_{l1,l2 \in L} (freq_{l1,l2} - (C_{distance} \cdot distance_{l1,l2})). \tag{4}$$

The total fitness value is weighted summation of these four values. For now we assign same weight to all the four factors so that the designed layout considers these equally. If w_{factor} denotes weight of a factor, then the equation for total fitness will be -

$$f_{total} = w_{keystk} \cdot f_{keystk} + w_{keyjamming} \cdot f_{keyjamming} + w_{flexibility} \cdot f_{flexibility} + w_{distance} \cdot f_{distance} \tag{5}$$

where $w_{keystk} = w_{keyjamming} = w_{flexibility} = w_{distance} = 1$.

E. The Proposed Genetic Algorithm:

Genetic Algorithms (GA), in principle, do not guarantee finding the best solution for a problem. Even our goal was also not to do so here. We showed that the current layout whatsoever is not the best layout. Using the proposed algorithm, we could find layouts that are more efficient comparing to both the traditional layout and some layouts proposed by others from time to time. Another reason for selecting GA for optimizing was to reduce memory requirements for finding the solution. In general, any combination of three or four letters from the twenty six letters can be assigned to any one of the eight keys. Therefore the search space contains a huge number of possible solutions. If we would use any graph based algorithm, many paths and nodes would have to be stored in memory. On the contrary, genetic algorithms always keep the solution that is found best till now.

Now we give insight to the detailed description of our algorithm. The major steps are crossover and mutation. Details of these steps will be discussed shortly. Among the population of first generation, 50 percent are generated randomly and the other 50 percent are taken from the traditional layout and layouts proposed in [16] and [17]. All these layouts are represented in a general form where the letters assigned to a key is followed by that key and these keys are then separated by comma. For example, the traditional layout will be expressed as: *abc2, def3, ghi4, jkl5, mno6, pqrs7, tuv8, wxyz9*. Crossover and mutation operations are applied on this initial population to get a new population.

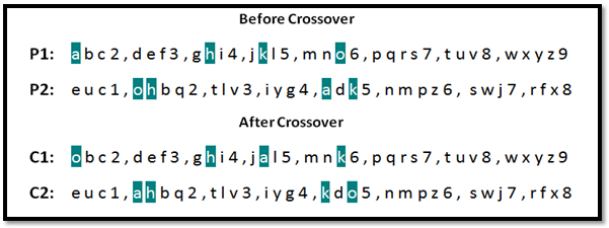

Figure 7. Crossover operation.

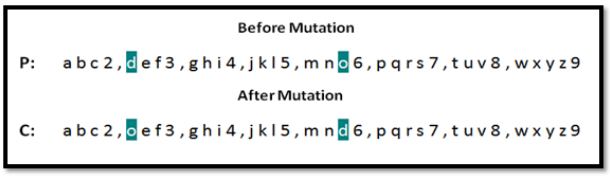

Figure 8. Mutation operation.

1) Crossover: In case of one point crossover, a single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children. Again, two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms. Single point and two point crossovers cannot be performed in case of our problem as there remains a probability of having duplicate letter(s) in one or both children after crossover. We apply uniform crossover between pair of individuals, but in a quite different manner. First we take one member from the pair and then select a number of letters from random positions. Next, positions of same letters are found out in another member. These letters are then serially swapped between the two members of the pair. A possible crossover operation is sketched in Fig. 7.

2) Mutation: Generally, mutation alters one or more gene values in a child from its initial state. But we cannot mutate a letter randomly in our devised method because of the probability of placing duplicate letters in the same chromosome or child (button here). To work around, we select two positions in an individual instead of one and swap the two letters in those positions. Fig. 8 outlines the state of a chromosome before and after mutation.

After mutation, the fittest n offspring are selected to be used in the next generation. Thus we execute these operations for g generations and keep record of the layout

with the highest fitness value. We used $n = 40$ and $g = 1000$. We run the algorithm for many times and present the best layout that we achieved from this genetic algorithm. Our proposed algorithm can be summarized in Fig. 9.

Following this meta-heuristic algorithm, we are able to obtain a keypad layout that performs outstandingly well than the traditional keypad in all of the four performance criteria mentioned before. As the proposed layout is quite different from the traditional one, a learning time will be needed for the users to get used to this design. But as soon as they become experienced to type with this new keypad layout, they will be greatly benefitted. The output layout along with other devised and proposed layouts is shown in Fig. 4.

All these calculations are performed considering normal mode or *ABC* mode of text input (alphabets only mode). There are some predictive text entry modes among which the T9 or Text on 9 keys mode is the mostly used one. T9's objective is to make it easier to type text messages. It allows words to be entered by a single key press for each letter, as opposed to the multi-tap approach in which selecting one letter often requires multiple keystrokes. It combines the groups of letters on each phone key with a fast-access dictionary of words. It looks up in the dictionary for all words corresponding to the sequence of key presses and orders them by frequency of use [13]. It is noteworthy that the obtained layout performs better in the T9 mode too. Since in T9 mode, only one keystroke is needed to type a letter, the total number of keystrokes will be the same for all layouts. Also, key jamming does not occur in predictive text mode. But the other two criteria can be considered to compare performance of text entry in T9 by different keypad layouts. The flexibility of thumb is increased and the total distance traveled is decreased respectively in same proportion as it does in normal text entry mode.

VI. Performance Studies

In this section we describe the datasets, experimental procedure, and then discuss and analyze the obtained results. Based on these results, we compare the performance of our proposed layout according to the criteria mentioned previously.

A. Data Sets:

For training dataset, we have used a sample SMS database obtained from [9]. This database consists of thousands of SMS, which contains more than 15000 words. This training set was used to do all pre-calculations such as counting frequencies of all single letters as well as two-letter combinations. These two information are at the core of the procedure for fitness measurement.

We tested and compared our new layout using other databases rather than the training data to enhance the probability of getting better performance in varying databases. For testing purpose, we have collected short messages from four different websites [9], [11], [13], [22].

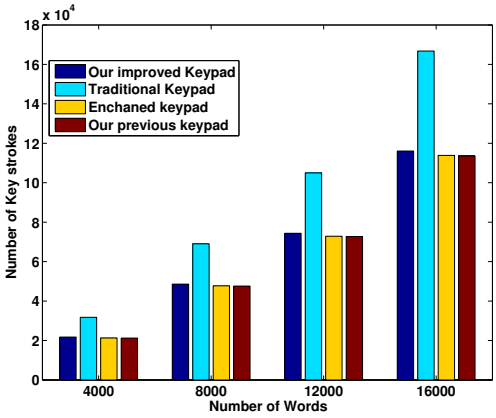

Figure 10. Number of keystrokes required using different keypads.

These testing inputs had 4000, 8000, 12000 and 16000 words respectively. The comparison was carried out with traditional keypad, the keypad proposed by Azad *et al.* [17] and our previously proposed keypad [16].

B. Experiment Setup:

We have implemented our algorithm with JAVA SDK 1.6 and performance measurement was done using MATLAB 10.

C. Performance Study:

As we have mentioned earlier, the prime motivation of our proposed algorithm was to reduce the total number of keystrokes, key jamming, and movement of the thumb as well as to extend users' flexibility to type. Although the new keypad did not win against all the keypads (i.e., traditional, enhanced, and flexible ones) in each of these four cases, yet a promising result came out that could make us think about improving the traditional layout practically.

Fig. 10 shows the performance of our proposed keypad in terms of the number of keystrokes required to type the words in our various testing dataset. We can note that our keypad performs much better than the traditional keypad but a little more keystrokes are required than the completely enhanced keypad [17] and the proposed layout in [16]. One important point to remember here is that, in our previous work we had considered three criteria only and had not dealt with the total distance covered by thumb and the completely enhanced keypad was proposed aiming to reduce keystrokes only. Here, we have considered four criteria with similar weight to each of them and that is what may be the reason of getting slightly worse result than two layouts in term of number of keystroke. But it is also noteworthy that we have to compromise a little as the ratio in which the number of keystrokes lessens than the traditional one is much higher than the ratio in which it increases from the other two. Average number of keystrokes required is decreased by up to 29.99 percent from the traditional keypad and

```
n ⟵ number of individuals in each population
g ⟵ number of generations
seeds ⟵ initial collection of individuals
best ⟵ best layout based on fitness

P ⟵ seeds
best ⟵ null

for each individual P_i ∈ P do
      AssessFitness(P_i)
      if Fitness(P_i) > Fitness(best) then
            best ⟵ P_i

for i = 1 to g do
      P ⟵ the fittest n individuals
      Q ⟵ { }
      for each individual P_i ∈ P do
            for each individual P_j ∈ P do
                  Children C_a, C_b  ⟵ Crossover (Copy(P_i), Copy(P_j))
                  C_a ⟵ Mutate (C_a)
                  C_b ⟵ Mutate (C_b)

                  AssessFitness(C_a)
                  if Fitness(C_a) > Fitness(best) then
                        best ⟵ C_a
                  AssessFitness(C_b)
                  if Fitness(C_b) > Fitness(best) then
                        best ⟵ C_b
                  Q ⟵ Q ∪ {C_a, C_b}
      P ⟵ Q ∪ P
return best
```

Figure 9. Genetic algorithm for finding optimized keypad.

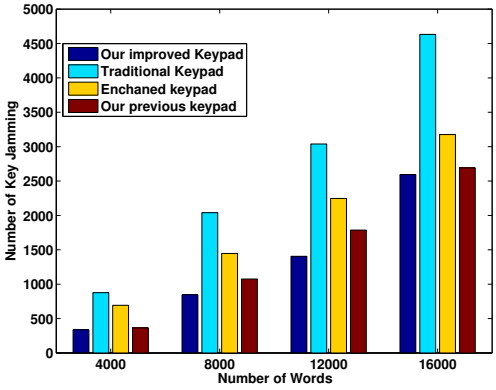

Figure 11. Number of key jamming happened using different keypads.

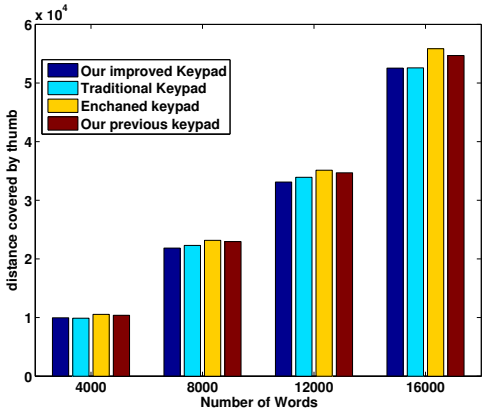

Figure 12. Amount of finger movement using different keypads.

increased by 1.97 percent, and 2.21 percent respectively from keypads in [17] and [16].

Then we experimented on the amount of key jamming while typing (Fig. 11). We can see that the standard keypad shows very low performance and the keypad proposed in [17], though performs better than the standard keypad, cannot reduce key jamming significantly. Our improved keypad performs better than all the three layouts chosen for testing. The difference in key jamming is significant when tested with large number of words. Reduction in key jamming from other keypads considered here is 51.38 percent, 31.49 percent and 12.23 percent respectively.

Another important criterion is measuring the distance that a thumb travels using the gap between two consecutive keys. Traveling less distance means consecutive letters are more probable to be in nearby keys which guarantee less movement and increase flexibility of the thumb. This feature also ensures the ease of use of our devised layout by people of all ages. From Fig. 12 we can observe that traditional layout performs better than the enhanced and our previously proposed layouts when the size of database increases and our new proposed layout outperforms all of these by reducing the distance in significant amount, that are 2.04 percent, 5.8 percent and 4.24 percent respectively.

Finally we measured the performance of our proposed

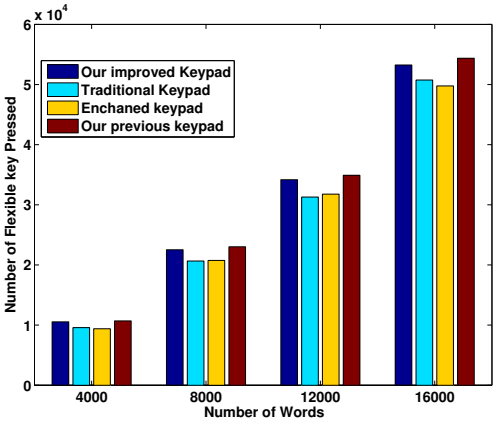

Figure 13. Number of flexible keystrokes using different keypads.

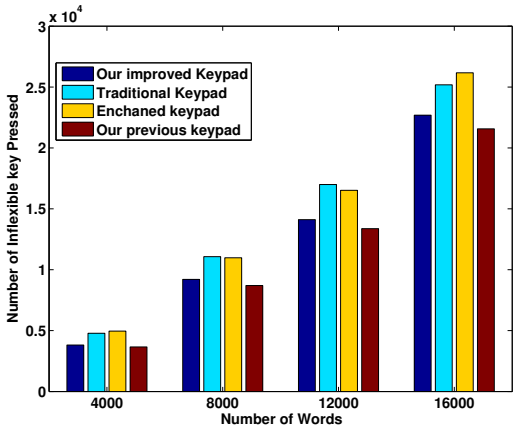

Figure 14. Number of inflexible keystrokes using different keypads.

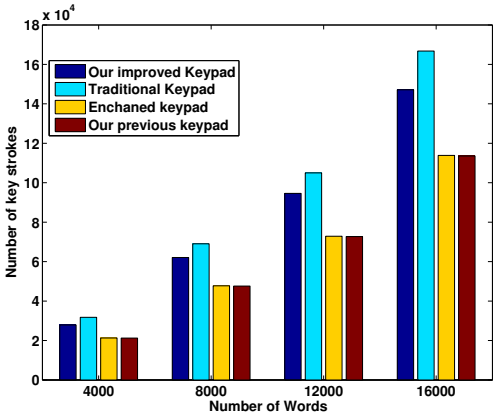

Figure 15. Comparison of number of keystrokes required after sorting the keypad.

TABLE I.
SUMMARY OF PERFORMANCE COMPARISON AMONG DIFFERENT KEYPADS.

	Enhanced Keypad in [17]	Flexible Keypad in [16]	Proposed Keypad
Number of Keystrokes	$639.40_{31.2\%}$	$637.83_{31.5\%}$	$652.02_{29.9\%}$
Number of Keyjamming	$18.91_{29\%}$	$14.78_{44.5\%}$	$12.95_{51.4\%}$
Keystrokes on convenient keys	$279.18_{0.3\%}$	$307.52_{8.6\%}$	$301.23_{7.3\%}$
distance travelled while typing	$311.75_{5\%}$	$306.78_{3.4\%}$	$293.67_{1\%}$

system in terms of key flexibility. We took two measures: the number of key presses on keys that are flexible to reach and the number of presses on inconvenient keys as identified earlier. From Fig. 13, it is clear that our system improves the number of flexible keystrokes compared to both the traditional keypad and the enhanced keypad [17]. Consequently, Fig. 14 also shows that our improved keypad is having less keystrokes in inflexible keys but a little higher than keypad in [16] for small databases. But on an average, our new layout shows improvement than all the other three in comparison using large databases. Our system decreases the inflexible key press required by up to 7.31 percent than the standard layout and thus improves performance.

We present average values of these four criteria obtained using each layout in Table. I. All the values are calculated for 100 words. The numbers in the subscript of each value signifies the deviation in performance of the particular layout from traditional keypad based on the criterion mentioned in that row. It is notable that the layout from our previous work is almost as good as the proposed layout. But unlike the proposed one, it shows much worse performance when compared about others features (i.e., flexibility). On the other hand, the new layout shows significant improvement in most of the features and compromise very little on other criteria.

One drawback of the proposed layout is that the letters are not placed alphabetically on the keys. To overcome this to an extent considering better human technology interaction, we sorted the keys on each key in the order they come in alphabet [23]. For instance, key '1' will be assigned to the three letters in the order 'ceu'. We then experimented of the performance shown by this new layout. As only the ordering of the letters on the keys is changed, other criteria except the number of keystrokes will illustrate same result. Even after the letters are sorted, the proposed system requires much less keystroke than the traditional keypad, reducing 10.96 percent keystrokes on an average (Fig. 15).

Next, we performed a t-test on the fitness values obtained from different runs of our code against the fitness value of the standard layout. We tuned the weight of various factors to get different layouts that show better result based on only that particular factor. The outcome of t-test ensures accuracy of our proposed genetic algorithm. The results are summarized in Table. II.

VII. FUTURE WORKS

Many tweaking factors can affect the designed layout of the mobile keypad. Though our sample database is quite large and we used varied number of SMS from

TABLE II.
T-TEST RESULTS FOR DIFFERENT KEYPADS.

	Fitness of Algorithm Output		Fitness of Traditional Layout		t-Test Result
	Mean	Standard Deviation	Mean	Standard Deviation	
All	$7.99E5$	$2.33E4$	$3.67E5$	0	+
Keystroke	$4.28E5$	$1.55E4$	$2.76E5$	0	+
Key Jamming	$1.12E4$	$3.93E3$	$4.20E3$	0	$\approx$
Flexibility	$3.26E5$	$1.16E4$	$3.18E5$	0	$\approx$
Movement of Thumb	$4.10E4$	$2.23E3$	$4.10E4$	0	$\approx$

different websites, if changed, it can produce a slightly different result. Also, while calculating the fitness according to keystrokes and key jamming, one can consider other properties rather than frequency. Depending upon letters included in the groups and their utility values, the proposed algorithm can show some minor changes in its output. But the procedure remains same and can be generalized. Another interesting future work would be to investigate possible improvements in text entry performance by replacing the alphabetical constraint that was used in our work with another layout constraint, such as keeping letters close to a QWERTY design.

The proposed algorithm may also be used for other languages than English. Since different languages have different number of alphabets and contain words with conjugate letters, to accommodate those in fixed number of keys, number of letters included in a key will vary. To produce a model layout with less key pressing and key jamming, and extended flexibility of finger movement for these languages, we can use the '*' or '#' key to implement other functionalities.

In future, we also plan to arrange user trials based on a prototype system on a mobile phone having our proposed layout to gain a practical impression of the typing speed. Comparing this result against traditional layout can ensure correctness of our method. Again, while calculating total distance, we assumed that all the keys are square in size which is not the practical case. Method of finding distance can be improved to get better result. In a nutshell, results shown here can be accumulated to give a thought for using the new layout as a replacement for the current layout with noteworthy enhancement of typing experience.

VIII. CONCLUSION

Text entry on cell phone devices has always been an interesting and vital problem for HCI researchers. Two main challenges which differentiate this particular research area from more generalized text entry are the ever-decreasing sizes of mobile devices and the "anyplace/anytime" use of mobile devices. Finding input modalities and interaction techniques that work well with these devices makes this problem space more challenging and difficult. However, progress in this area will potentially benefit the increasing users of mobile devices. This may also bring about advances in universal usability of information systems. This work has made significant contributions in the field of mobile text entry research, focusing primarily on improving the traditional layout of keypad. Here, we have described a novel approach for designing a multitap keypad for cellular phones considering frequency of alphabets, key jamming, flexibility of the thumb, and amount of thumb movement to use the keypad. Our new design has surpassed in all four performance criteria used for text entry when compared to the standard keypads. Though a learning period will be needed to get accustomed to the new layout, yet it will provide an accelerated speed for typing. The new layout will cause a reduction in number of extra keystrokes and unnecessary key jamming with improved usability and performance for novice users.

ACKNOWLEDGMENT

We thank all the members of the HTI research group, Dept. of CSE, BUET for their support and the reviewers for their detailed remarks.

REFERENCES

[1] G. R. Hayes, J. S. Pierce, and G. D. Abowd, "Practices for Capturing Short Important Thoughts", in *Ext. Abstracts of CHI 2003* , pp 904–905.

[2] P. Tarasewich, "Wireless Devices for Mobile Commerce: User Interface Design and Usability", B. E. Mennecke and T. J. Strader (eds.), "Mobile Commerce: Technology, Theory, and Applications", Idea Group Publishing, pp 25-50, 2002.

[3] The Star, 11.7 billion SMSes sent in first three months. Retrieved August 10, 2007, from "www.thestar.com.my/news/story.asp?file=/2007/8/8/nation/18525524&sec=nation".

[4] GSMA's Hubbing Programme to Streamline Global Delivery of SMS Message, GSM Association Press release (12th October, 2006), "http://www.gsmworld.com/news-room/pressreleases/2093.htm".

[5] S. I. Mackenzie. "Mobile Text Entry Using Three Keys." *Proceedings of the Second Nordic Conference on Human-Computer Interaction - NordiCHI*, pp. 27–34. New York, 2002.

[6] T. Evreinova and G. E. Evreinov, "Four-Key Text Entry Augmented with Color Blinking Feedback for Print-Handicapped People with Ocular Pathology", in *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA 2005)*.

[7] M. M. Morshed, Y. K. Thu and Y. Vrano, "Frequency Based Two-Layer Multitap BangIa Input Method for Mobile Phones," in *Proc. 10th International Conference on Computer and Information Technology, (ICCIT)*, pp. 1-7, Dec. 2007.

[8] A. K. Pathan, S. R. Choudhury, A. K. M. N. Islam, A. Azfar ,"Design of an Interactive Bangla Mobile Keypad Based on Phonetics", in *Proc. ICCIT 2004*.

[9] S. N. Srirama, M. A. A. Faruque, M. A. S. Munni, "Altenative to Mobile Keypad Design: Improved Text Feed", in *Journal of Computing Research Repository (CORR)*, vol. abs/1007.3,2010.

[10] S. N. Srirama, M. A. A. Faruque, M. A. S. Munni, "Alternatives to Mobile Keypad Design: Improved Text Feed, in *Proc. International Conference on Computer and Information Technology*, 2003, JU, Dhaka, Bangladesh.

[11] A. Dix, J. Finally, G. D. Abowd, R. Beale, "Human-Computer Interaction", Prentice-Hall, Englewood Cliffs, NJ, 1997.

[12] R. Sharmeen, M. A. K. Azad, S. Ahmad and S. M. Kamruzzaman, "Smart Bengali Cell Phone Keypad Layout", in Proceedings of *International Conference on Computer and Information Technology*, (ICCIT 2005).

[13] J. Gong, "Improved text entry for mobile devices : alternate keypad designs and novel predictive disambiguation methods", PhD Dissertation, Northeastern University, 2008

[14] J. Gong, and P. Tarasewich, "Alphabetically Constrained Keypad Designs for Text Entry on Mobile Devices", in *Proc. of CHI2005*, pp. 211–220.

[15] M. D. Dunlop, M. M. Masters, "Investigating five key predictive text entry with combined distance and keystroke modeling", in *Journal of Personal and Ubiquitous Computing archive*, Vol. 12 Iss. 8, Nov. 2008.

[16] S. Rubya, S. Monir, H. S. Ferdous, "A Flexible Keypad Reducing Keystrokes and Key Jamming for Cell Phones", in *Proc. 14th International Conference on Computer and Information Technology (ICCIT 2011)*, Dhaka, Bangladesh.

[17] M. A. K. Azad, R. Sharmeen, S. Ahmad and S. M. Kamruzzaman, "Completely Enhanced Cell Phone Keypad", in Proceedings of the *1st International Conference on Information Management and Business (IMB2005)*.

[18] Structure of human thumb: "www.anatomy-resources.com", as on October, 2011.

[19] Structure of thumb joints: "www.handuniversity.com/topics.asp? Topic_ID=19", as on October, 2011.

[20] Structure of thumb: "www.ncbi.nlm.nih.gov/entrez/", as on October, 2011.

[21] A. Oulasvirta, J. B.Lehtovirta, "A Simple Index for Multimodal Flexibility", in Proceedings of the *28th international conference on Human factors in computing systems*, Pg 1475-1484 , 2010.

[22] "www.affordablephones.net", as on January, 2012.

[23] J. Preece, Y. Rogers, and H. Sharp, "Interaction Design: Beyond Human computer interaction", Wiley, N.Y., 2002.

[24] J. Lee, B. McKay, "Optimizing a personalized cellphone keypad", in Proceedings of the *Proceedings of the 5th international conference on Convergence and hybrid information technology (ICHIT'11)*, Pg 237-244, 2011.

[25] E. M. Y. Wang, S. S. Y. Shih, "A Study on Thumb and Index Finger Operated Interface for Personal Mobile Devices: Mobile Phone Keypad and Joystick", in Proceedings of the *16th International Conference on Industrial Engineering and Engineering Management, 2009. IE&EM '09.*, Pg 766-770, 2009.

Sabirat Rubya was born in Dhaka, Bangladesh. She has completed her Bachelors of Science (B.Sc.) degree in Computer Science and Engineering (CSE) in 2012 from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). She is a member of the Human Technology Interaction (HTI) research group there. Her current interests include User Interface, Data Mining and Genetic Algorithms.

Samiul Monir has completed his Bachelors of Science (BSc.) in Computer Science and Engineering (CSE) in March, 2012 from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). He is currently working under Human Computer Interaction(HTI) research group in CSE,BUET. His main research focus is based on User Interface, Machine learning and Data Mining. He recently got a full free scholarship to a 3 days workshop in Summer school of National University in Singapore.

He was born in Dhaka, Bangladesh. He is currently working as a Lecturer in Stamford University in Bangladesh.

Hasan Shahid Ferdous was born in Dhaka, Bangladesh. He has completed his Bachelors of Science (BSc.) in Computer Science and Engineering (CSE) in 2008 from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), the finest institution of its kind in Bangladesh. Then he has served BRAC University and BUET as a lecturer. He has completed his Masters (by Research) degree in 2011 from Gippsland School of Information Technology (GSIT), Faculty of Information Technology, Monash University, Australia. Currently, he is serving as a lecturer in the Dept. of CSE, BUET and leading the Human Technology Interaction (HTI) research group there. His current interests include OpenStreetMap (OSM), eLearning, and Natural User Interface (NUI). He has received multiple grants from the Government of Bangladesh and World Bank.