한국지능시스템학회 논문지 2009, Vol. 19, No. 6, pp. 814-820

최적의 휴대폰 키패드 디자인을 위한 알고리즘

Algorithms for Designing Optimal Keypads of Mobile Devices

김현민* · 김용혁**+

Hyunmin Kim* and Yong-Hyuk Kim**+

* 광운대학교 임베디드SW공학과

** 광운대학교 컴퓨터소프트웨어학과

요 약

일반 12-키 휴대폰 키패드의 알파벳 배치 방식은 한두 가지로 통일되어 있다. 대개 하나의 키에 여러 개의 알파벳을 순서가 있게 배치한다. 영문을 입력할 때 우리나라를 포함한 비영어권 국가에서는 대부분 멀티탭(multi-tap) 방식을 사용하는데, 이 멀티탭 방식에서는 같은 키를 반복해서 눌러야 다음 알파벳을 입력할 수 있게 된다. 본 논문에서는 멀티탭 방식으로 다양한 상황의 영문 데이터를 입력할 때 필요한 키 누름 횟수를 최소화하는 키패드를 디자인한다. 제안한 유전 알고리즘을 이용해 최적의 키패드를 디자인할 수 있었다. 키의 개수는 최소 8개에서 최대 12개까지를 사용하며 알파벳 순서를 유지한 것과 그렇지 않은 키패드 디자인에 대해 다양하게 실험하였고, 결과적으로 모든 경우에 개선된 키패드 디자인을 도출해냈다.

키워드 : 키패드 디자인, 유전 알고리즘, 모바일 기기

Abstract

The arrangement of the general 12-button cellular phone keypad is unified with one or two kind. In general, the several alphabets are arranged in one key. When inputting an English, countries which do not speak English use multi-tap method. We can input next alphabet by pressing key repeatedly. We address the problem of finding multi-tap-based keypad designs which minimize the number of key presses for various case. Genetic algorithms is proposed method for optimal keypad designs. We are to maintain the number of keys, which is ranged from 8 to 12. We also show experimental using non-alphabetically-constrained and alphabetically-constrained arrangement, respectively. Finally, we give improved keypad designs.

Key Words : Keypad design, genetic algorithm, mobile device

1. 서 론

일상생활에서 사용하는 언어는 자신이 어떤 목적을 가지는가에 따라 달라진다. 예로 구어와 문어는 단어 및 문장 구조에서 차이를 많이 찾아볼 수 있다[1]. 휴대폰 역시 마찬가지이다. 휴대폰에 문자를 입력하는 목적에 따라 사용되는 단어는 달라질 것이다. 휴대폰의 키패드는 전화번호만을 입력하기 위해 만들어졌다. SMS(short message service), IM(instant message)의 사용의 증가와 무선 인터넷 기술은 키패드에게 더 많은 역할을 요구하게 되었다. "무선 인터넷 이용 실태"에 따르면 52%가 무선 인터넷을 하는 것으로 나타났다. 이런 변화는 효율적인 키패드 디자인 연구를 요구하게 되었다[2].

키패드 디자인에 대한 연구는 휴대폰 키패드 외의 분야에서도 많이 진척된 상황이다. 몸이 불편한 사람들은 일반적인 키보드를 이용하기 힘들다. 그렇기 때문에 한 손으로 쉽게 문자를 입력할 수 있는 현재 휴대폰에서 사용하는 것과 같은 디자인에 대한 연구가 많이 이루어져 왔다. 현재 우리가 많이 사용하는 방식인 T9, QWERTY, 멀티탭(multi-tap) 방식 이외에도 Less-tap, SYMKEY 등과 같은 방법들이 연구되었다.

QWERTY 방식은 26개의 알파벳이 각 키에 바인딩 되어 있고 일반 키보드와 같은 디자인이기 때문에 빠르게 글을 입력할 수 있다. 2008년에는 QWERTY 키패드를 탑재한 제품의 비중이 무려 30%까지 늘어 작년 같은 기간의 11%에 비해 급격한 상승을 이루었다[3]. 알파벳 하나에 하나의 키가 필요하기 때문에 26개의 키가 필요하고 키를 위한 기판의 사이즈가 그만큼 커지거나 키의 크기가 줄어들 수밖에 없다.

T9 방식의 키 배치는 하나 이상의 알파벳을 8개의 키에 나누어 배치한다. 여기까지는 멀티탭 방식과 유사하다. 멀티탭 방식은 각 키의 바인딩된 문자를 입력하기 위해서 같은 키를 한 번 이상 눌러야 하지만 T9은 소프트웨어 방식을 통해 키 조합을 계산하여 많이 사용되는 단어를 자동으로 입력해주어 키를 누르는 횟수를 줄여준다. 소프트웨어 방식이기 때문에 이를 구동하기위해 데이터 사전이 필요하고 원하지 않는 단어가 나오는 경우 추가적인 동작들이 필요하다. 또한 사용하지도 않는 단어들 때문에 휴대폰 자원을 낭비하기도 한다.

접수일자 : 2009년 9월 18일
완료일자 : 2009년 11월 15일
+ 교신저자
이 논문은 2009년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.

본 논문에서 사용하는 방식은 키 배치만을 조절하여 효율적인 키패드를 만드는 것이다. 소프트웨어적인 방법을 고려하지 않기 때문에 누르는 방식은 멀티탭 방식을 따른다. 예를 들면 “ghi”가 할당된 키에서 ‘i’라는 문자를 입력하기 위해서는 키를 3번 눌러야 한다. 키패드를 디자인하기 위해 동적 프로그래밍과 유전 알고리즘 방법을 제안한 알파벳 순서를 유지한 키패드는 저자들의 이전 학술대회 논문[4]에 사용된 방법에 데이터의 종류와 키의 개수를 변화시킨 다양한 결과를 도출한다. 알파벳 순서를 유지하지 않은 키패드는 기존의 동적 프로그래밍 방법은 사용하지 않고 새로운 유전 알고리즘을 제안한다. 기존의 이진 표현 대신 순열 표현을 사용하고 이에 맞게 교차방법은 PMX를 사용한다. 단지 순서만 바뀌는 것으로 키에 알파벳을 배치하는 것을 결정할 수 없다. 유전 알고리즘을 통해 구한 자식 해에 동적 프로그래밍을 적용해 키패드 배치를 결정하고 이를 평가한다.

실험에 사용된 제한사항 몇 가지를 설명한다. 키의 개수는 8개에서 12개까지로 제한한다. 알파벳만을 실험대상으로 한정한다. 문자를 입력하다 보면 여러 가지 다른 기호들이 들어갈 수 있는데 이는 각 휴대폰마다 다르게 설정되어 있다. 우리는 문자 이외의 기호들은 공백으로 취급한다. 실험을 위한 문장은 실제 영어권에서 사용하는 데이터를 사용한다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 키패드 디자인에 관련된 기존 연구에 대해 설명한다. 3장에서는 실험의 평가방법과 실험에 쓰인 데이터에 관해 살펴보고, 4장에서는 효율적인 키패드를 디자인하기 위한 방법으로 동적 프로그래밍과 유전 알고리즘을 제안한다. 5장에서는 다양한 실험 결과를 정리하고 6장에서 결론을 내린다.

2. 관련 연구

키패드를 효율적으로 디자인하기 위한 연구는 계속 진행되어 왔다. 그 중에는 유전 알고리즘을 사용한 연구도 존재한다.

Lesher 등[5]은 몸이 불편한 사람들을 위한 키패드 디자인을 설계했다. 한손으로 사용할 수 있는 키패드 디자인은 모바일 기기에서 사용하는 키패드와 매우 유사하다. 논문에서는 유전 알고리즘을 사용했는데 빠르게 평가하기 위한 방법을 고안했다. 평가방법은 가중치를 주는 k-gram 방식을 사용했는데 각 키에서의 위치에 따라 가중치를 주고, k개의 문자를 하나의 집합으로 보고 이 집합의 알파벳이 같은 키에 들어있으면 가중치를 주었다. 이는 연속된 문자가 같은 키에 존재하면 이동키를 눌러주는 행동을 계산해 준 것이다.

Pavlovych와 Stuerzlinger[6]는 기존의 T9 배치를 조정한 Less-tap 디자인을 설계했다. T9이 8개의 키에 알파벳이 순서대로 있는 것에 착안해 기존의 사람들이 이것에 익숙해져 있음을 전제로 하고 각 키에서 자주 사용하는 알파벳을 키의 앞쪽으로 배치했다. 반복해서 키를 누르는 것을 최소화하기 위해 이런 방법을 사용했다. 이것은 미국식 키패드에 익숙한 사용자에게 유용한 방법이다.

저자들의 이전 학술대회 논문[4]에서는 제약된 상황에서 영문 SMS를 입력하기 위한 최적의 키패드 디자인을 설계했다. 제약사항으로 8개 또는 9개의 키를 이용하고, 알파벳 순서를 유지했다. 키 입력 방식은 T9의 예측 방식이 아닌 멀티탭 방식을 사용했다. 키패드 디자인을 생성하기 위해 동적 프로그래밍과 유전 알고리즘을 사용했다. 실험 데이터는 자주 사용하는 SMS 데이터와 싱가포르 국립대학(National University of Singapore)에서 연구 목적으로 조사한 SMS 데이터를 사용해 테스트했다[7]. 기존의 T9 기반의 키패드 디자인, Alphabetic 기반의 키패드와 비교해 결과를 보여줬다. 본 논문은 이전 연구를 확장하여 키의 개수, 입력데이터 종류, 알파벳 순서 유지 여부의 제약사항을 더 다양화하며 최적의 키패드 디자인을 찾고자 한다.

3. 평가 방법 및 사용 데이터

키패드가 얼마나 효율적인지 실험하기 좋은 방법은 실제 사람이 입력을 하며 얼마나 시간이 단축되었는지를 체크하는 것이다. 하지만 숙련도와 개인의 차이가 존재하기 때문에 객관적 자료로 사용하기는 어렵다. 본 논문에서는 데이터 입력에 필요한 실제 키를 누르는 총 횟수를 세어 비교하기로 한다. 이를 위해 다음 몇 가지 규칙에 따라 움직인다. 같은 키에 있는 연속된 문자를 입력하기 위해서는 키를 누른 후에 옆으로 이동 후 다시 키를 눌러야 한다. 만약 T9 기반 디자인 휴대폰에서 "hi"라는 문자를 입력하는 경우 ‘h’ 와 ‘i’ 가 같은 키에 바인딩 되어 있다. 계속해서 같은 키를 누르는 경우 다음 알파벳으로 입력 문자가 바뀌기 때문에 일반적으로 일정한 시간을 대기하거나 다음 위치로 이동할 수 있는 키를 눌러야 한다. 본 논문은 이동키를 눌러 입력을 마무리하는 것으로 처리해 키를 누르는 횟수를 한 번 늘리는 방식으로 처리했다. 문자를 입력하는 도중 공백문자를 입력하기 위해서는 문자입력을 위한 이동키를 한 번 누르고 다시 공백을 위해 이동키를 한 번 더 눌러야 한다. 다시 말해서 단어를 입력하는 도중 공백문자 한 칸을 넣어 주기 위해서는 총 두 번의 이동키를 눌러야 한다.

대문자와 소문자를 구분해 계산하였다. 대문자를 사용하다가 소문자를 사용하기 위해서는 함수키를 한 번 누르면 되기 때문에 누르는 키 횟수를 한 번 늘리는 것으로 처리했다.

키패드를 평가하기 위해 본 논문은 두 가지 데이터 집합을 사용했다. "Training set"은 동적 프로그래밍에서는 각 알파벳의 등장 빈도를 계산하기 위해 사용되었고, 유전 알고리즘에서는 평가 함수를 위해 사용되었다. "Test set"은 만들어진 키패드 디자인을 최종 평가하기 위해서 사용되었다.

각 데이터 집합은 SMS 메시지, 문어(written english), 구어(spoken english), 논문, 사전으로 각 상황에 따른 영문 데이터를 바탕으로 실험했다. SMS 메시지는 하나는 정형화된 데이터를 모아둔 사이트에서 SMS 리스트를 추출했고[8], 싱가포르 국립대학에서 연구 목적으로 조사한 SMS 리스트에서도 데이터를 추출했다[7]. 문어는 뉴욕 타임즈의 2009년 2월 기사들을 데이터로 만들었다. 구어는 유명한 미국 드라마인 프렌즈(friends)의 대본을 데이터로 만들었다. 논문과 사전은 표준적인 논문과 영어사전을 바탕으로 데이터를 생성했다. 각 데이터들의 사이즈는 표 1과 같다.

표 1. 입력데이터 사이즈 (단위: 바이트)
Table 1. Sizes of input data

	Training set	Test set
SMS	692,238	175,923
구어	122,263	32,262
문어	120,831	31,422
논문	23,341	23,341
사전	1,246,531	1,246,531

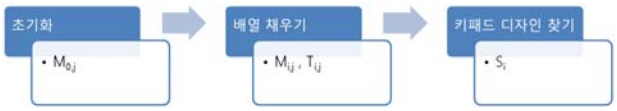


그림 1. 제안한 동적프로그래밍의 흐름도
Fig. 1. Flow chart of the proposed dynamic programming

$DP(n,m)$ // n: # of letters, m: # of keys
{
1. for each $j=1,2,...,n$
2. $M_{0,j} \leftarrow F_{1,j}$;
3. for each $i=1,2,...,m$
4. for each $j=i+1,i+2,...,(n-m+i)$
5. $M_{i,j} \leftarrow \min_{i \le k \le j}(M_{k,j-1}+F_{k+1,i})$;
6. $T_{i,j} \leftarrow \arg min_{i \le k \le j}(M_{k,j}+F_{k+1,i})$;
7. $p \leftarrow M_{m,n}$;
// 실제 구분자를 찾기 위한 backward trace
8. for each $i=m,m-1,...,1$
9. $S_i \leftarrow p$;
10. $p \leftarrow T_{i,p}$;
}

- $M_{i,j}$는 i개의 알파벳 중 j개의 구분자가 존재할 때 최소로 키가 눌러질 때의 횟수.
- $T_{i,j}$는 $M_{i,j}$가 최소값을 가질 때의 구분자 위치.
- $F_{i,j}$는 i부터 j 알파벳까지 빈도의 합.
- S_i는 i개의 구분자를 가지는 최적 키패드 디자인.

그림 2. 키패드 디자인을 위한 동적 프로그래밍
Fig. 2. The proposed dynamic programming for keypad design

4. 제안 방법

4.1 동적 프로그래밍

큰 문제의 답을 구하기 위해 작은 문제의 답이 이용되는 경우 재귀적 알고리즘으로 이를 해결할 수 있다. 재귀호출은 같은 호출을 여러 번 반복하는 경우가 발생하는데 이 재귀적 중복을 해소한 방법이 동적 프로그래밍이다[9]. 본 논문은 누르는 횟수를 최소화하는 키패드를 디자인하기 위해 동적 프로그래밍 방법을 이용했다. 이 동적 프로그래밍은 키를 누르는 횟수를 줄이기 위해서 자주 사용되는 알파벳을 각 키의 앞쪽에 위치시키는 것이 목적이다. 알파벳의 등장 횟수를 빈도(frequency)라고 정의하고 SMS 리스트에서 각 알파벳의 빈도를 구한다. 점화식을 세우는데 그 핵심 원리는 다음과 같다. 26개의 알파벳이 k개의 키에 바인딩된 최적 키패드를 찾으려면 i개의 알파벳이 $k-1$개의 키에 최적으로 바인딩되고 $i+1$부터 'z'까지가 하나의 키에 바인딩될 때의 최적의 i를 찾아야 한다.

제안한 동적 프로그래밍의 흐름은 그림 1과 같다. 동적 프로그래밍은 정의된 배열을 순서대로 채우는 방법으로 최적해를 구할 수 있다. 그러기 위해 우선 초기 값 $M_{0,j}$를 구한다. 그리고 점화식을 이용해 배열을 순서대로 채워 나간다. 모든 배열을 다 채웠을 때 최적 키패드의 키가 눌러진 횟수를 구할 수 있다. 하지만 실험에서 원하는 것은 키패드 디자인이기 때문에 되추적기법을 사용해야 한다. 즉, 배열을 채울 때 최소의 값을 가질 경우 이전 배열 위치를 저장해 키패드 디자인을 찾는 것이다.

그림 2는 위의 흐름을 의사 코드로 자세히 기술한 것이다. $M_{i,j}$는 i개의 알파벳 중 j개의 구분자가 존재할 때 최소로 키가 눌러질 때의 횟수로 동적 프로그래밍 점화식으로 계산된다. $F_{i,j}$는 i부터 j 알파벳까지 빈도의 합으로 $M_{i,j}$를 구하기 위해 사용된다. 1, 2행은 초기 값을 설정하는 부분이다. 3, 4, 5, 6행은 점화식을 이용해 정의된 배열을 채우는 것을 표현한 것이다. 5행은 위에서 설명한 점화식을 표현한 것이고, 6행은 실제 디자인을 구하기 위해 되추적 배열을 채우는 것을 표현한 것이다. 반복문을 이용해 배열을 끝까지 채울 수 있게 된다. 8, 9, 10행은 실제 키패드 디자인을 구하는 부분이다. 최적일 때 배열을 되추적하며 실제 키 배치를 S_i 저장하게 된다.

4.2 유전 알고리즘

유전 알고리즘은 개체의 자연선택 원리에 기반을 둔 최적화 알고리즘이며, 재생산, 교차, 돌연변이 연산자를 사용한다[10,11]. 세대는 특정한 순간의 해의 집합을 의미한다. 각 세대의 해는 교차나 돌연변이를 통해 다음 세대의 해를 만들어 낸다. 인구는 특정한 세대의 해의 개수를 의미한다.

본 논문에서 제안한 유전 알고리즘은 알파벳 순서로 배치하는 키패드 디자인과 알파벳 순서를 지키지 않고 배치하는 키패드 디자인으로 나눠진다. 두 키패드 디자인 모두 Sastry의 유전 알고리즘 툴 박스를 사용했다[12]. 일반적인 목적으로 사용되는 유전 알고리즘을 지원하는 이 라이브러리를 목적에 맞게 하기 위해 두 경우로 나누어 각각 파라미터를 설정했다. 표 2는 알파벳 순서 유지 키패드 디자인을 위한 파라미터 값이다.

표 2. 알파벳 순서 유지 키패드 디자인을 위한 유전 알고리즘 파라미터
Table 2. Genetic parameters for keypad designs with alphabetical constraints

파라미터	값
변수의 개수	25
변수의 값 정의	0 혹은 1
인구 수	100
세대 수	50
교체 확률	0.9
선택 방법	토너먼트 선택: 크기 2
교차 방법	일점 교차
돌연변이 방법	유전자마다 변이
돌연변이 확률	0.1

표 3. 랜덤 키패드 디자인을 위한 유전 알고리즘 파라미터
Table 3. Genetic parameters for keypad designs with non-alphabetical constraints

파라미터	값
변수의 개수	26
변수의 값 정의	1, 2, ..., 26
인구 수	100
세대 수	500
교체 확률	0.9
선택 방법	토너먼트 선택: 크기 2
교차 방법	PMX
돌연변이 방법	랜덤하게 위치 교환
돌연변이 확률	0.02

그림 3. 알파벳 순서를 유지한 키패드 디자인의 염색체 표현
Fig. 3. Representation for keypad designs with alphabetical constraints

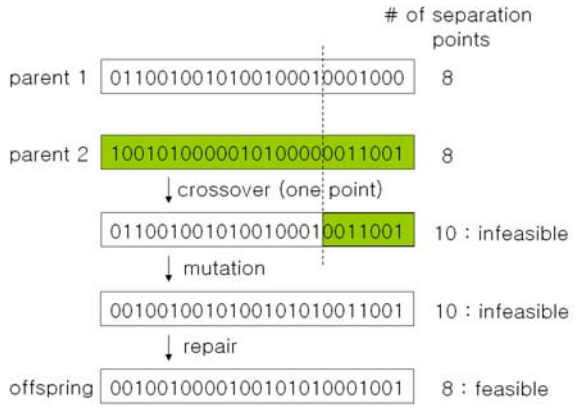


그림 4. 자식 해의 생성 과정
Fig. 4. Genetic procedure to produce an offspring

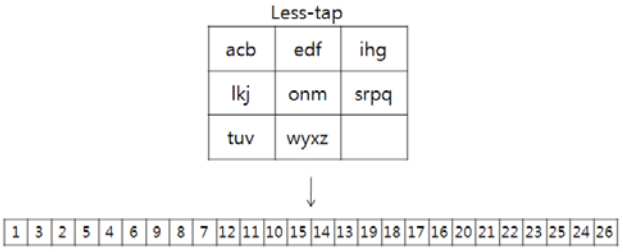


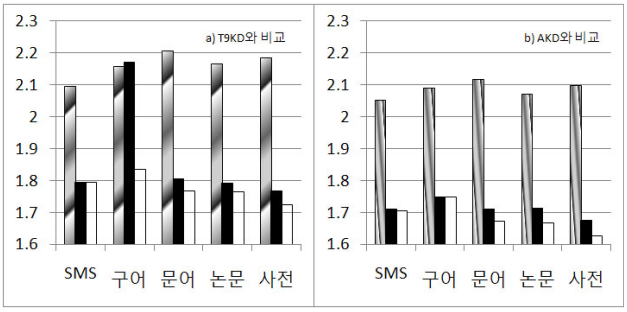
그림 5. 랜덤 순서 키패드 디자인의 염색체 표현
Fig. 5. Representation for keypad designs with non-alphabetical constraints

알파벳 순서의 키패드 디자인의 유전자 표기는 25개의 키 구분자 중에 구분자가 존재하는 부분을 '1'로 하고 나머지는 '0'으로 표현하였다. 총 키의 개수를 m이라 하면 $m-1$개의 '1' 값이 존재한다. Alphabetic 기반의 키패드를 그림으로 표현하면 그림 3과 같다.

초기 해는 랜덤으로 $m-1$개의 구분자를 선택한다. 이를 교차와 돌연변이를 통해 새로운 자식을 만들어낸다. 이렇게 생성된 자식의 구분자 개수는 부모와 다를 수 있다. 이와 같은 경우 자식의 구분자 개수를 보정해야 한다. 부모의 구분자 수보다 적다면 랜덤하게 구분자 하나를 선택하여 늘려주고, 크다면 랜덤하게 구분자 하나를 줄여준다. 그림 4는 자식 해를 만드는 일련의 연산 과정을 그림으로 표현한 것이다. 이런 과정을 거쳐 생성된 자식 해의 품질은 3장의 방법대로 평가된다.

랜덤 순서 키패드 디자인을 위한 유전 알고리즘은 다르게 설계되었다. 이를 위한 파라미터 값은 표 3과 같다. 알파벳 순서 유지 키패드 디자인은 유전자 표기를 위해 구분자를 주었지만 랜덤 순서 키패드 디자인은 이것이 불가능하기 때문에 알파벳을 1에서 26 사이의 값의 순열(permutation) 표현 방식을 사용했다. 예를 들면 'a'는 1이고 'z'는 26이다. 유전자의 개수는 총 26개이고 초기 해는 랜덤으로 생성하기 위해 차례대로 되어있는 알파벳을 랜덤한 순서로 섞는다. Less-tap 방식의 키패드 디자인을 표현하면 그림 5와 같다. 이렇게 생성된 부모 해를 교차와 돌연변이 연산을 통해 자식 해를 생성한다. 교차방식은 PMX(partially matched crossover) 방식을 사용했다[10]. PMX는 염색체가 순열로 표시되는 경우에 사용되는 잘 알려진 교차 방식이다. 두 부모 해에 임의의 절단점 두 개를 선택해 첫 번째 부모로부터 절단점 내의 유전자를 상속받는다. 나머지 위치는 두 번째 부모로부터 상속받지만 해당 값이 이미 상속되었다면 첫 번째 부모 해로부터 상속받은 위치를 찾아 두 번째 부모 해의 유전자를 상속받는다. 이렇게 해도 중복될 수 있는데 중복되는 유전자는 첫 번째 부모 해에서 위치를 찾아 두 번째 부모 해에서 다시 상속받는 방식을 계속 반복하여 없애준다. 그림 6은 일련의 연산을 그림으로 표현한 것이다. 돌연변이 연산은 임의의 두 알파벳의 위치를 교환하는 방법을 사용했다. 두 알파벳의 위치를 교환하면 중복된 유전자가 생성되지 않는 장점이 있다. 보정하는 연산을 하지 않아도 된다.

그림 6. PMX 교차 연산의 예
Fig. 6. An example of PMX crossover



(■: DPKD, □: GAKD)
그림 7. 기존 키패드 디자인과 성능 비교
Fig. 7. Performance comparison with existing keypad designs

만들어진 자식 해는 알파벳의 순서만을 가지고 있을 뿐 각 알파벳이 어느 키에 들어가야 하는지는 정해지지 않았다. 본 논문에서는 구분자를 정해주기 위해 4.1절의 동적 프로그래밍 방식을 유전 알고리즘 내부에 적용했다. 자식 해를 동적 프로그래밍을 이용해 배치해주는 것이다. 알파벳 배치를

위해 유전 알고리즘을 사용하는 것보다 빠르게 배치하는 것이 가능하다. 모든 연산을 거친 새로운 키패드 배열을 평가해 최적의 키패드 디자인을 찾게 된다. 평가하는 것은 이전과 마찬가지로 키가 눌리는 총 횟수를 사용했다.

5. 시뮬레이션 및 결과

DPKD(Dynamic Programming for Keypad Design)와 GAKD(Genetic Algorithms for Keypad Design)는 각각 4.1절과 4.2절에서 제안한 방법들이다. 각 데이터는 사이즈가 일정하지 않다. 결과를 한 눈에 비교하기 위해 총 데이터의 크기로 결과 값을 나누어 정규화 한다.

표 4. 8개에서 12개까지의 키를 사용한 알파벳 순서유지 키패드 디자인 (GAKD의 결과)
Table 4. Keypad designs with alphabetical constraints with 8 to 12 keys (Results of GAKD)

사용된 키 개수	데이터 종류	키패드 디자인
8	SMS 구어	abcd efg hijk lm nopq rs tuv wxyz
	문어 논문	ab cd efg hijk lm nopq rs tuvwxyz
	사전	ab cd efgh ijk lm nopq rs tuvwxyz
9	SMS	ab cd efg hijk lm nopq rs tuv wxyz
	구어	abcd efg hijk lm n opq rs tuv wxyz
	문어 논문	ab cd efg hijk lm n opq rs tuvwxyz
	사전	ab cd efgh ijk lm nopq rs tuvwx yz
10	SMS	ab cd efg hijk lm n opq rs tuvwx yz
	구어 문어 논문	ab cd efg hijk lm n opq rs tuv wxyz
	사전	ab cd efgh ijk lm n opq rs tuvwx yz
11	SMS	ab cd ef gh ijk lm n opq rs tuv wxyz
	구어 문어	ab cd efg h ijk lm n opq rs tuv wxyz
	논문	ab cd ef gh ijk lm n opq rs tu vwxyz
	사전	ab cd ef gh ijk lm n opq rs tuvwx yz
12	SMS 구어	ab cd efg h ijk lm n opq rs tuv wx yz
	문어	ab cd ef gh ijk lm n opq r s tuv wxyz
	논문	ab cd ef gh ijk lm n opq r s tu vwxyz
	사전	ab cd ef gh ijk lm n opq rs t uvwx yz

첫 번째 테스트는 알파벳 순서를 유지한 키패드 디자인을 기존의 알려진 키패드 디자인과 비교했다.

T9KD(T9-based Keypad Design)는 T9 방식의 키패드 디자인으로 8개의 키를 사용한다. AKD(Alphabetic-based Keypad Design)는 우리나라 SKY 휴대폰 등에 쓰이는 디자인으로 9개의 키를 사용한다. 그림 7은 같은 키의 개수를 가지는 키패드 디자인들을 비교한 것이다. y축의 값은 정규화된 결과 값으로 문자 하나를 입력할 때 필요한 평균 키 누름 횟수와 같고 따라서 작을수록 더 좋은 결과가 된다. T9KD와 AKD는 SMS에서 최적의 값을 보여준다. GAKD는 사전 데이터를 사용했을 때 가장 좋은 값을 가지는데 T9KD와 AKD보다 각각 21%, 22% 개선했다. SMS 데이터를 사용할 때에도 T9KD보다 14%, AKD보다 17% 더 높은 성능을 보여준다.

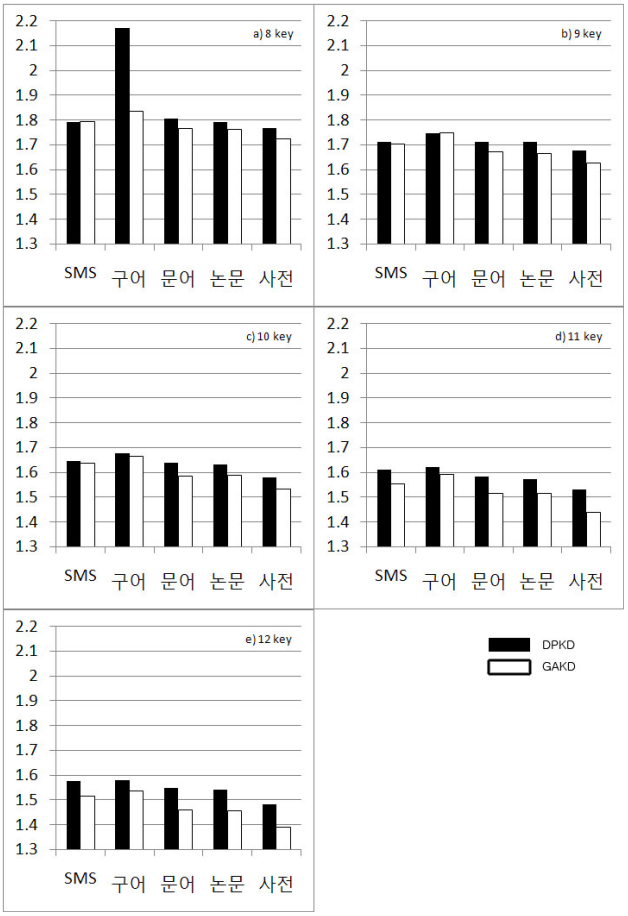


그림 8. 표 4에 주어진 키패드 디자인의 성능 비교
Fig. 8. Performance comparison of keypad designs given in Table 4

두 번째 테스트는 각 데이터를 8개에서 12개까지 키 개수를 바꿔가며 동적 프로그래밍과 유전 알고리즘을 비교했다. 각 조건에서 생성된 키패드 디자인은 표 4와 같고 각 값을 비교한 결과가 그림 8의 차트에 주어진다. 두 방법의 차이는 키의 개수가 증가할수록 커지는 것을 볼 수 있다. SMS 데이터에서 8개 키에서는 같은 값을 보이던 두 방법이 9개 키에서는 0.4%, 10개 키에서는 0.5%, 11개 키에서는 3.6%, 12개 키에서는 4% 결과 값의 차이를 보인다. 빈도를 이용한 동적 프로그래밍과 유전 알고리즘이 결과 차이를 보이는 주된 이유는 단어와 문장 구조 때문이다. 본 논문은 같은 키에 존재하는 알파벳이 연속으로 나올 때 카운트를 하나 추가한다. 대소문자가 변경될 때에도 카운트를 하나 추가해 준다. 이와 같은 데이터는 빈도를 이용한 방법에는 존재하지 않는 정보이기 때문에 동적 프로그래밍으로는 최적의 키패드를 생성할 수는 없게 된다. 유전 알고리즘은 실제 키가 눌러진 횟수를 통해 평가하기 때문에 이것을 반영할 수 있는 것이다.

표 4를 보면 데이터에 따라 알파벳 배열이 조금씩 바뀌는 것을 볼 수 있다. 사용하는 단어, 문장 구조 등이 다르다는 것을 의미한다. 그림 8을 보면 같은 키의 개수를 가지고 있어도 입력 데이터에 따라 결과가 조금씩 다른 것을 볼 수 있다.

세 번째 테스트는 기존의 연구 결과를 그와 같은 키의 개수를 가지는 GAKD와 FKD의 결과와 비교한다. GAKD는 4.2절에서 제안한 방법으로 유전 알고리즘을 통해 랜덤한 순서의 키패드 디자인을 생성하는 방법이다. FKD(Frequency-based

Keypad Design)는 동적 프로그래밍 방법을 사용하지 못하기 때문에 새로 제안한 방법으로 가장 자주 나오는 알파벳을 정렬하여 각 키의 앞쪽부터 차례로 할당한 키패드 디자인이다. 제안한 방법과 비교하기 위한 키패드 디자인은 Less-tap[6], k-gram[5], KOGA (Keyboard Optimization using Genetic Techniques)[13]이다. Less-tap은 빈도를, k-gram과 KOGA는 유전 알고리즘을 사용했다. 그림 9를 통해 각 값을 비교해 볼 수 있다. GAKD는 Less-tap을 SMS 데이터에서 4% 개선하고 있다. 각 데이터별로는 구어에서 5%, 문어는 8%, 논문과 사전에서는 9% 개선한다. GAKD는 k-gram을 9개 키에서 평균 2%, 10개 키에서 평균 5% 개선한다. KOGA는 GAKD보다 20% 성능이 안 좋은 것을 볼 수 있다.

표 5. 8개에서 12개까지의 키를 사용한 랜덤 순서 키패드 디자인 (GAKD의 결과)
Table 5. Keypad designs with non-alphabetical constraints with 8 to 12 keys (Result of GAKD)

사용된 키 개수	데이터 종류	키패드 디자인
8	SMS	adjx eu hrv iybq nmf ogk slc twpz
	구어	ag euf hmk iyp nrvxzq olj swc tdb
	문어	agj eu ipyqz nhb odk rlw scf tmvx
	논문	auy eg ipx nlfj ohkz rdbq scw tmv
	사전	asx euz ihy lcv npfj ogkq rmw tdb
9	SMS	ag eu hlxq iyj nwpz oc rmv sfb tdk
	구어	agj eu hlv iy nwpq okx rmz scf tdb
	문어	afx eu hl iy npj ogk rmwq scv tdbz
	논문	afz dhx eu ipj nbw ogy rl scv tmkq
	사전	ah euz iyj lcw npv ob rmxq sgf tdk
10	SMS	agx eu hmv iy lc np ob rwj sfz tdkq
	구어	af eu hdxz iyj lm nbp ok rw sc tgvq
	문어	afz eu hbk iyx lm npjq og rw scv td
	논문	afj dhvq eg iu lw npz oyk rb scx tm
	사전	afq cbv eu iy lmw np okj rhx sg tdz
11	SMS	ab eg hfz iy lmxq np ok rv scj td uw
	구어	af euz hkx ib lg nm ov rw scj td ypq
	문어	af dcz eu hb iyx lmq np ok rw svj tg
	논문	ayq dmx ef hv iu lwz npj ok rb sc tg
	사전	af cmj dpz eu iy lh nb ok rwq svx tg
12	SMS	ax dc ej hb iy lm np ok rvz sf tgq uw
	구어	af ej hk ip lm nb ox rv scz td uw ygq
	문어	aj cgq dk eu hf iy lm np ox rwz sv tb
	논문	ax cf dm ej hk iu lw np oy rbq svz tg
	사전	ax cmj dp ez iy lh nb ok rv sf tgq uw

네 번째 테스트는 GAKD와 FKD를 키의 개수와 데이터에 따라 비교한다. GAKD를 통해 생성된 키패드 디자인은 표 5와 같다. 그림 10을 통해 각 조건에서의 성능을 비교할 수 있다. 데이터 종류에 따른 차이를 보면 GAKD는 FKD를 평균적으로 SMS에서 1%, 구어에서 1.6%, 문어에서 1.7%, 논문에서 2.1%, 사전에서 1.6% 개선했다. 키의 개수별로 값을 보면 GAKD가 FKD를 평균적으로 8개 키에서는 2%, 9개 키는 1.9%, 10개 키는 1.8%, 11개 키는 1.2%, 12개 키는 1% 개선했다. 키의 개수가 증가할수록 개선율이 줄어드는데 이는 한 키에 바인딩 되는 알파벳이 줄어들어 같은 키를 반복하여 누르는 횟수가 줄기 때문이다. 12개의 키를 사용한 경우 고작 2개의 키만이 3개의 알파벳을 가지기 때문에 FKD와 GAKD 키패드 디자인의 차이가 크지 않다.

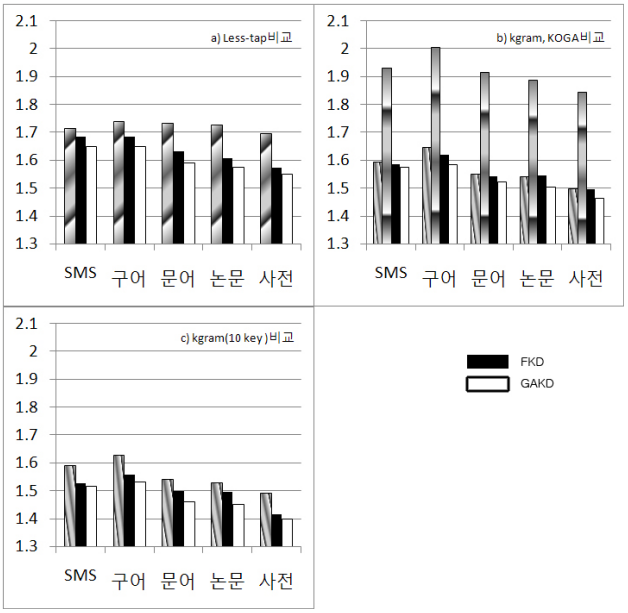


그림 9. 기존 키패드 디자인과 성능 비교
Fig 9. Performance comparison with existing keypad designs

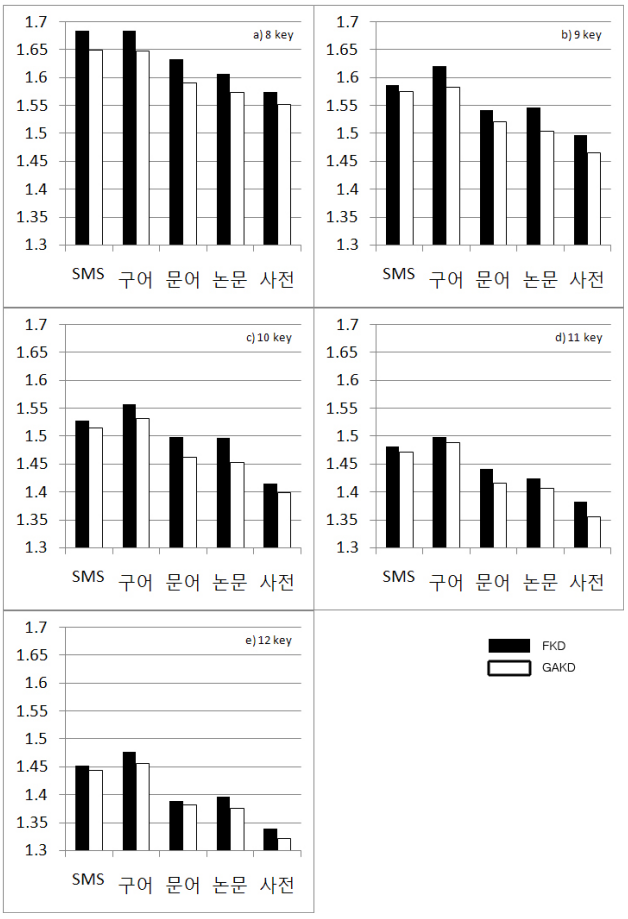


그림 10. 표 5에 주어진 키패드 디자인의 성능 비교
Fig. 10. Performance comparison of key pad designs given in Table 5

6. 결 론

본 논문은 멀티탭 방식을 기반으로 일반적인 영문 데이터가 입력될 때 필요한 키 누름 횟수를 최소화하는 키패드를 디자인했다. 8개에서 12개까지의 키를 사용하고, 알파벳 순서를 유지한 키패드 디자인과 랜덤한 순서의 키패드 디자인을 구했다. 키패드 디자인을 생성하기 위한 방법으로 동적 프로그래밍, 유전 알고리즘, 그리고 그 둘의 결합(combination) 방법을 제안했다. 생성된 키패드 디자인은 SMS, 구어, 문어, 논문, 사전 데이터를 이용해 실제 눌러진 키 횟수를 구해 값을 비교했다. 비교 대상은 기존의 키패드 디자인과 기존 논문에서 설계된 키패드 디자인, 빈도 기반 키패드 디자인, 그리고 본 논문에서 제안한 방법의 결과이다. 실험으로부터 유전 알고리즘을 통해 생성된 키패드 디자인이 모든 경우에 가장 좋은 성능을 보이는 것을 알 수 있었다. 키의 개수가 증가하면 한 키에 바인딩 되는 알파벳의 수가 줄기 때문에 큰 성능 개선을 얻을 수는 없었지만 제안한 유전 알고리즘을 이용하여 각 상황에 따라 최적의 키패드를 디자인할 수 있다는 것이 본 논문의 성과이다.

참 고 문 헌

[1] W. Chafe and D. Tannen, “The Relation Between Written and Spoken Language,” *Annual Review of Anthropology*, Vol. 16, pp. 383-407, 1987.

[2] 한국인터넷진흥원, “무선인터넷 이용실태조사” (http://www.mobizen.pe.kr/attachment/1303603587.pdf).

[3] L. Graham. “The NPD Group: iPhone 3G Leads U.S. Consumer Mobile Phone Purchases in the Third Quarter of 2008” (http://www.npd.com/press/releases/press_081110.html)

[4] H. Kim and Y.-H. Kim. "Design of Efficient Mobile Keypad Using Genetic Algorithms," *In Proc. KIIS Spring Conference*, Vol. 19. No. 1, pp. 295-298. April 2009. (in Korean)

[5] G. W. Lesher, B. J. Moulton, and D. J. Higginbotham. "Optimal Character Arrangements for Ambiguous Keyboards," *IEEE Transactions on Rehabilitation Engineering*, Vol. 6, No. 4, pp. 415-423, December 1998.

[6] A. Pavlovych and W. Stuerzlinger. "Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones", *In Graphics Interface*, pp. 97-104, 2003.

[7] G. Leech, P. Rayson, and A. Wilson. Word Frequencies in Written and Spoken English: Based on the British National Corpus, Pearson ESL. 2001.

[8] funSMS.net(http://www.funsms.net/sms_messages.htm)

[9] H. Sakoe and S. Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 26, No. 1, pp. 43-49, February 1978.

[10] 문병로, “*쉽게 배우는 유전 알고리즘: 진화적 접근법*”, 한빛미디어, 2008.

[11] H. Bang, J. Park, J. Hong and H. Lee, “Model Predictive Control System Design with Real Number Coding Genetic Algorithm”, *Journal of Korean Institute of Intelligent Systems*, Vol. 16, No. 5, pp. 562-567, October 2006.

[12] K. Sastry. "Single and Multiobjective Genetic Algorithm Toolbox in C++," Technical report, Illinois Genetic Algorithms Laboratory, 2007. Illigal Report No. 2007016, 2007.

[13] B. J. Oommen and J. R. Zgierski, “Keyboard Optimization Using Genetic Techniques,” *In Proc. Tenth Annual International Phoenix Conference on Computers and Communications*, pp. 726-732, March 1991.

저 자 소 개

김현민(Hyunmin Kim)
2007년 : 숭실대학교 컴퓨터학부 (학사)
2008년～현재 : 광운대학교 임베디드SW 공학과 (석사과정)

관심분야 : 유전 알고리즘, 임베디드SW
Phone : 02) 940-5212
E-mail : wbaim@kw.ac.kr

김용혁(Yong-Hyuk Kim)
1999년 : 서울대학교 전산과학 전공(학사)
2001년 : 서울대학교 컴퓨터공학부(석사)
2005년 : 서울대학교 컴퓨터공학부(박사)
2005년～2007년 : 서울대학교 반도체 공동연구소 연구원
2007년～현재 : 광운대학교 컴퓨터소프트웨어학과 조교수

관심분야 : 최적화, 진화연산, 지식공학
Phone : 02) 940-5212
E-mail : yhdfly@kw.ac.kr