한글 크로스워드 퍼즐 자동 생성을 위한 알고리즘 개발

(Development of Auto-generation Algorithm for Korean Crossword Puzzle)

이 승 희 † 권 혁 철 †† 조 환 규 †††
(Seung-Hee Lee) (Hyuk-Chul Kwon) (Hwan-Gue Cho)

요 약 크로스워드 퍼즐은 세계적으로 널리 인기가 있는 게임 중의 하나로 주어진 힌트에 적합한 단어를 끼워 넣어 바둑판 모양의 퍼즐 판을 채워 넣는 게임이다. 컴퓨터의 기술이 발달하면서 크로스워드 퍼즐 문제를 자동으로 생성하고 해결하는 프로그램을 만들어보려 하였고, 그 결과 상용화된 프로그램들도 등장을 하였다. 그러나 이런 상용 프로그램들 대부분이 고정된 퍼즐 판의 모양에서 이미 만들어져 있는 문제들을 맞추어나가는 형식을 취하고 있는 것들이다. 왜냐하면, 가로세로 일정한 크기 내에서 나올 수 있는 퍼즐 판의 모양에 맞춘 단어들의 집합을 찾는다는 것은 매우 어려운 문제이기 때문이다. 특히 이런 프로그램들은 영어나 불어 같은 언어들에 국한된 것이고, 아직까지 한글을 기반으로 한 크로스워드 퍼즐 자동 생성 프로그램은 상용화되어 있지 않다. 따라서 본 논문에서는 한글을 기반으로 하는 크로스워드 퍼즐 문제 자동 생성 프로그램을 만들기 위해 어떤 점들이 고려되어야 하는지를 살펴보고 크로스워드 퍼즐 문제를 자동으로 생성해주는 시스템, Korizzle을 설계하였다. 이 시스템에 이용된 알고리즘과 Korizzle을 간단히 소개하고 몇 가지 사항들에 대한 성능 평가를 해보고자 한다.

키워드 : 크로스워드 퍼즐

Abstract A crossword puzzle is one of the popular word games around the world in which you work out the answers and write them in the white squares of a pattern of small black and white squares. As the technology of computers develops, some people worked about making and solving crossword puzzle games, which led them to a commercial use. However, almost all of these commercial programs are ones where you do ready-made puzzles with a fixed size because it is very difficult to make puzzles in a certain size, picking up some among a great number of words to fit for them. Furthermore, these programs are only for a very few languages, such as English, French, not for Korean. Accordingly, we took a look at what should be considered to make an automatic puzzle-generating program for Korean, and in this paper we implemented Korizzle, a system making the puzzles automatically. We introduce the algorithm used for Korizzle and evaluate the its performance.

Key words : crossword puzzle

1. 서 론

크로스워드 퍼즐은 남녀노소를 막론하고 많은 사람들이 즐기는 낱말 맞추기 게임의 하나이다. 보통 글자가 들어갈 하얀색의 빈칸과 글자가 들어가지 않는 검은색의 빈칸에 주어진 가로세로 힌트를 참조하여 단어를 채워 넣는 게임이다. 이때 가로와 세로로 서로 교차하는 지점에는 두 단어에서 공통으로 들어있는 낱글자가 교차하기 때문에 크로스워드 퍼즐이라는 이름을 붙이게 되었다.

크로스워드 퍼즐 게임이 언제부터 시작되었는지는 명확하지 않지만 19세기 중엽, 영국의 어린이 잡지에서 처음으로 그 모습을 찾아볼 수 있다. 이후 1913년 영국에 있던 기자 아서 윈(Arthur Wynne)이 미국으로 건너가

† 정 회 원 : (주)나라 인포테크
sheel@pusan.ac.kr
†† 종신회원 : 부산대학교 전자전기정보컴퓨터공학부 교수
hckwon@pusan.ac.kr
††† 정 회 원 : 부산대학교 전자전기정보컴퓨터공학부 교수
hgcho@pusan.ac.kr
논문접수 : 2005년 2월 22일
심사완료 : 2005년 10월 10일

서 처음으로 뉴욕 월드 지에 성인용으로 개재를 하였다. 이 크로스워드 퍼즐은 아주 많은 인기를 얻어 단행본으로까지 출간되어 75만부가 팔리기도 했다[1]. 호주에서는 잡지에서 Crozzle이라는 puzzle game을 개최하고 있는데, game은 주로 10 X 15 크기의 grid 에 100여 개의 단어 리스트를 주고서 제일 많은 단어들을 일정한 규칙에 따라 배치하여 높은 점수를 얻는 사람이 이기는 경기이다.

초기의 크로스워드 퍼즐 문제는 가로세로 15칸으로 구성된 형태였으며, 사람이 일일이 사전을 뒤져가며 단어와 힌트들을 나열하는 형태로 문제들을 만들었다. 그 이후 크로스워드 퍼즐 문제를 직접 사전을 뒤져서 문제를 만들던 A.F.Ritchie 와 D.S.Macnutt 같은 "Human Crossword Compiler" 들은 자신만의 독특한 방법과 규칙으로 수 만개의 크로스워드 퍼즐 문제를 만들었다. 그러나 지금은 규칙과 모양 면에서 많은 변화를 가져왔고, 또 문제를 만드는 면에서도 자동화 연구가 이루어져 왔다. 크로스워드 퍼즐을 즐기는 사람이 늘어나고 데이타베이스부분이나 탐색기법 등 컴퓨터와 관련된 여러 기술들이 발달함에 따라, 자동으로 크로스워드 퍼즐 문제를 만들거나, 풀 수 있도록 하는 방법은 없는가가 연구가 되었다. 처음으로 크로스워드 퍼즐문제를 자동으로 생성하려고 시도를 한 사람은 L.J.Mazlack이었다. 2,000여 개 정도의 단어를 가진 빈칸이 비교적 적은 퍼즐 문제를, heuristic방법으로 한 단어씩 빈칸을 채울 글자를 찾아나가는 letter-by-letter 방법을 제안하였다. Ginsberg는 90년에 발표한 논문에서 24,000개의 단어를 가진 사전에서 4×4 이상의 크기를 지닌 crossword puzzle에 대해 연구를 하였다. 어느 지점에서 확장을 먼저 할 것인지를 결정하는 조건도 cheapest-first heuristic, connectivity, 그리고 adjacency restriction, 세 가지 정도로 제시를 하였다[2].

Meehan과 Grey는 Ginsberg 와 Mazlack의 두 가지 방법을 비교하여 성능평가를 하기도 하였다. 이때 Prolog와 C를 써서 구현한 경우와 CLP(constraind language)와 C인터페이스를 이용하여 구현한 경우를 비교하면서, crossword puzzle같은 search problem에서는 CLP언어가 필요하다는 것을 결과를 발표하기도 하였다[3]. Harris는 1990년 사전에 있는 글자들의 적절한 배치로 만들 수 있는 모든 crossword puzzle문제를 만들기 위해 dynamic slot table을 제안하기도 하였다. 하지만, 사전과 퍼즐의 크기만 정해진 상태(제약이 없는 상태)에서 가능한 모든 형태(geometry)의 문제를 만들어 내는 것은 NP-complete문제에 해당이 되며 이에 대한 해결책으로 dynamic slot table을 제안하였다. 그는 결론에서 사람이 직접 문제를 생성하는 방법에 비해 자동 생성 방법은 최적의 해결책이 아닌 적당한 정도를 얻어낼 뿐이라고 결론을 짓고 있다[4-6]. 또한 Jo Spring은 Harris, Rankin등이 시도하는 여러 crozzle Solver들을 대상으로 correctiveness, effectiveness, run-time efficiency에 대해 test를 하고 새로운 알고리즘이나 시스템의 test비교 자료로 결과를 제공하였다[7].

이런 연구들의 결과로 자동으로 crossword puzzle문제 또는 답을 생성하는 프로그램들도 많이 제공되고 있으며([1,8-10]), 1997년에 발표된 CrossWord Compiler는 크로스워드 퍼즐 문제를 사람이 만드는 것보다 더 잘 만들고 싶다는 동기에서 개발이 된 프로그램이다[11]. 이 프로그램의 기술문서를 보면 사전의 구성에서부터 힌트를 제공하는 부분까지 단계별로 상세하게 기술을 하고 있다. 글자판을 채워나가는 과정을 'walk'라 이름을 짓고 여러 개의 walk에 따라 단어 채움이 실패할 경우의 backtracking도 제시하고 있다. 또 2005년 6월에 새로운 버전이 소개가 된 FineCrosser 는 여러 조건들과 단어사전들도 선택을 하여 크기 40까지의 crossword puzzle 문제를 자동 또는 수동으로 생성할 수 있도록 하고 있다[12].

우리나라에서도 많은 사람들이 크로스워드 퍼즐을 즐기고 있으며 신문이나 잡지 등에도 크로스워드 퍼즐 문제가 많이 소개가 되고 있다. 인터넷 상에서도 크로스워드 퍼즐을 이용할 수 있는데, (주)가로세로에서 특허출원을 하였던 네트워크 게임 형식의 크로스워드 퍼즐풀이는 현재 제공이 되고 있지 않다. 또 인터넷 사이트인 퍼즐랜드[13]에서는 시사분야나 테마 분야로 나누어서 퍼즐을 제공하고 있으나, 문제를 매번 다르게 자동으로 생성해주는 것이 아니라 이미 만들어진 문제들을 푸는 형태의 서비스를 제공하고 있다.

크로스워드 퍼즐이 단순히 오락적인 면뿐만 아니라, 교육적인 면에서의 효과도 뛰어나다는 점을 감안한다면 한글 기반의 크로스워드 퍼즐 문제를 자동으로 생성하여 주는 시스템이 있다면 아주 유용할 것이다. 그런 이유에서 한글에 적합한 크로스워드 퍼즐 문제를 생성해주는 시스템을 고안해보고자 한다. 먼저 크로스워드 퍼즐의 기반이 되는 사전의 구축에 한글의 특징이 반영되어야 하는 이유를 사전적 분석을 통해 알아보고자 한다. 그 분석 결과를 바탕으로 간단한 알고리즘을 적용한 자동 생성 시스템을 구축하고 가로세로의 크기와 같은 조건들을 변화시킴에 따라 결과가 달라지는 정도를 평가해보고자 한다.

2. Crossword Puzzle system의 구조

이전의 연구들을 살펴보면 크로스워드 퍼즐 문제 자동 생성 시스템의 구축 시 고려해야 하는 점들을 몇 가

지 찾을 수 있다. 사전을 구성할 때 사전의 크기와 글자들의 평균적인 길이를 고려하여야 한다. 사전의 크기에 따라 구성방법이나 검색 방법들이 다양해질 수 있기 때문이다. 두 번째로 퍼즐 판에 제약을 주는 방법을 고려하여 보아야 한다. 예를 들면 가로 세로 크기를 난이도의 척도로 삼아서 난이도가 크기에 비례하도록 할 수도 있다. 그렇지 않다면 한 글자에서부터 연결이 될 수 있는 글자들의 수를 다르게 함으로써 난이도를 다르게 할 수 도 있다. 또는 퍼즐판에 놓이는 빈칸의 개수를 제약조건으로 둘 수도 있다. 퍼즐 판의 크기 등이 미리 결정된 퍼즐 문제를 제약이 있는 퍼즐 문제(constrained) 라고 한다. 그리고 제약이 있는 경우의 퍼즐 문제는 퍼즐 글자판의 단어배치 형상(geometry)이 미리 결정이 되므로 정적으로 구현을 할 수 있다. 그렇지 않은 경우는 제약이 없는(unconstrained) 동적인(dynamic) 경우라고 한다. 마지막으로 해당하는 글자가 없을 경우에 대한 고려가 있어야 한다. 이때 낱 글자를 하나씩 바꾸어가면서 찾는 것을 letter-by-letter 방법이라 하고, 글자 전체를 다른 글자로 바꾸어 넣는 것을 word-by-word 방법이라고 한다.

2.1 크로스워드 퍼즐 시스템의 형식적 정의(Formal Definition)

본 논문에서 제안하는 크로스워드 퍼즐 생성 시스템은 한글을 기반으로 하면서, 단어들 사이에 교차점을 제외하고는 인접을 허용하지 않는 Crozzle이라는 퍼즐의 모양과 유사하므로 이름을 Korizzle이라고 지었다. 다음은 시스템 정의에 필요한 6가지 사항이다.

Korizzle = (D, S, D_a, L, D, C_n)
U: 퍼즐 구성을 위한 사전(universal dictionary)
S: 퍼즐의 가로세로 크기(size)
D_a: 한 개의 글자가 가질 수 있는 최대 자식 수 (degree of node)
L: 퍼즐에서 허용하는 글자의 최대 길이(length)
D: 전체 퍼즐 판의 밀도(Density)
C_n: 허용하는 사이클의 수(number of cycle, n=1,2…n)

U는 퍼즐 판에 들어갈 글자들과 힌트로 주어지는 뜻들로 이루어진 검색의 전체 범위, 즉 사전을 의미한다. 사전은 적절한 구조를 가지고서 추가, 수정, 삭제 등의 연산이 가능해야 한다. 또 사용의 편이를 위해 다른 분야의 사전으로도 교체가 쉬워야 한다. 특히 지금까지 한글을 기반으로 한 크로스워드 퍼즐 생성 문제는 다루어지지 않았으므로 한글에 대한 분석이 먼저 이루어져야 적절한 구조의 사전을 구성할 수 있다.

본 논문에서는 한글 표준국어대사전의 글자 중 길이가 2이상인 글자들의 일부인 64,800 개의 글자에 대해 조사를 하였다. 글자들의 평균 길이, 낱글자들의 개수, 낱글자들의 빈도수, 위치 p에서 볼 수 있는 낱글자들의 수 등에 대해 조사를 해보았다. 조사 결과 다음과 같은 몇 가지 결론을 얻을 수 있었다. 첫째, 전체 낱말의 80%이상이 글자 길이 2~3의 분포를 갖는다. 크로스워드 퍼즐을 구성하기 위한 최소의 글자 길이 2이상인 글자들에 대한 길이 분석 결과를 표 1에서 볼 수 있다. 이에 비해 Ginsberg가 105,897개의 영어단어를 대상으로 길이에 따른 분포를 조사한 결과를 조사를 한 결과는 표 2와 같다[2].

두 표를 비교하여 보면 영어의 경우는 글자의 길이 분포가 거의 고른 분포를 보이는 반면, 한글의 경우 95% 이상이 길이 4이하의 글자들로 구성이 되어있다. 두 번째 특징으로는 영어에 비해 낱글자들의 개수가 많다는 점이다. 한글의 경우, 표준자음 19개와 표준모음 21개, 이중모음 11 개가 초성, 중성, 종성의 자리에 놓임에 따라서 글자를 구성한다. 이들이 이론적으로 조합으로 만들어내는 글자들의 수는 11,172 개이다[14]. 그러나 실지로는 "쬥"처럼 쓰이지 않는 글자들도 있으므로 가능한 글자들의 개수들보다는 작다. 영어의 경우는 알파벳 26개의 낱 글자들로 이루어진다. 세 번째 특징은 낱 글자들이 고르지 않은 분포를 가진다는 점이다. 영어에 비해 아주 많은 개수의 낱 글자들은 사용 빈도에 있어서도 고르지 않은 분포를 보인다. '굉'(1회),'밖'(5회)처

표 1 64,800개의 단어를 가진 한글 사전을 대상으로 조사한 한글의 글자 길이 별 비율

글자길이	2	3	4	5	6	7	8	9	10
해당개수	23,814	29,004	8,589	1,717	522	137	39	7	3
비율	37.31%	45.44%	13.46%	2.69%	0.82%	0.21%	0.06%	0.01%	0.00%

표 2 Ginsberg가 105,897개의 단어를 가진 영어 사전을 대상으로 조사한 영어의 글자 길이 별 비율

글자길이	3	4	5	6	7	8
해당개수	853	3,342	6,894	10,745	14,574	15,901
비율	1%	4%	8%	2%	6%	7%
글자길이	9	10	11	12	13	14
해당개수	1,045	12,705	9,107	6,920	4,635	2,809
비율	1%	14%	1%	0.8%	0.5%	0.3%

럼 낮은 빈도로 쓰인 것부터, '이'(3068회), '기'(2075회) 같이 높은 빈도를 보이는 낱글자들도 있다. 그림 1을 보면 전체 표본 사전 중에서 10번 이내의 빈도를 보이는 낱글자들의 수가 591개로서 전체 낱글자 개수의 42%에 해당한다.

영어, 불어, 그리고 독일어 등을 대상으로 한 이전의 연구들에서는 퍼즐을 위한 사전의 구조는 낱글자가 26개 밖에 되지 않으며, 글자의 길이 분포 또한 고르다는 점에서, 낱글자를 기준으로 사전을 구성하거나, 글자의 길이에 따라 사전을 구성하는 방법을 많이 사용하고 있다. 특히 Cross Compiler에서 쓰인 방법은 글자들의 길이를 최대 30으로 제한한 뒤, 크기 30의 배열을 선언하고, 각각은 해당되는 길이로 구성이 된 단어를 가리킬 수 있도록 다시 26개의 배열로 구성된 노드들의 집합으로 구현을 하고 있다[11].

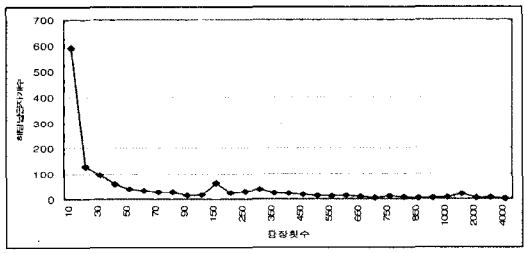


그림 1 낱글자 각각에 대해 한글 사전에서의 등장횟수를 조사한 결과

그러나 위에서 조사된 한글의 특성을 고려하여 본다면, 적절한 구조가 아니다. 먼저 낱글자를 기준으로 사전을 구성하는 방법은 영어의 경우 26개의 낱글자에 대해서만 구성을 하면 되지만, 한글은 무려 1,000개가 넘으므로 1,000개 배열이 필요하다. 또 낱글자들의 빈도가 1부터 3,060까지 다양하지만 전체 낱글자의 45%가 빈도수 600이하이므로 낱글자를 기준으로 배열 또는 링키드 리스트 구조로 사전을 구성하는 것은 공간적인 면에서 비효율적일 수 있다.

두 번째로 글자의 길이 별로 글자들을 연결하여 사전을 구성하는 방법도 한글의 경우는 글자의 길이 2~4에 해당하는 것이 80%나 되므로 나머지 길이 4부터 10까지의 경우는 20%의 단어를 위해 공간적인 면에서의 낭비를 가져오게 된다.

사전이 사용될 crossword puzzle에서는 주로 행해지는 연산이 위치 P_i에 낱 글자 W_i가 들어있는 길이 l인 글자를 찾는 질의연산이다. 그러므로 이 연산이 행해지는 사전 역시 위치 p에 따른 검색과 길이 l에 대한 검색이 같이 이루어 질 수 있는 구조로 만들어지는 것이 가장 효율적이라고 할 수 있다.

이런 문제들을 고려하여 본 논문에서 제시하는 사전의 구조는 다음과 같다.

사전에서 허용하는 글자의 최대 길이를 L라고 한다면, 각 위치 i에 대한 배열 L개로 사전이 이루어진다. 각 배열의 원소는 위치 P_i에 나타나는 낱 글자 W_i와 그 낱글자가 위치 i에 포함이 되어있는 글자들의 리스트로 구성이 된다. 그러므로, 만약 위치 P_i에서 볼 수 있는 낱 글자들이 모두 m개라면 배열 P_i는 m개의 헤드가 있는 링키드 리스트의 구조를 갖게 된다[그림 2]. 이런 구조의 사전은 단어와 뜻으로 구성된 .txt 파일을 읽어 들인 뒤 구성이 되는 사전형태이다. 예를 들면 '가지나물'의 경우 위치 1에서는 '가', 위치 2에는 '지'가 있으므로 위치 1의 "가"를 헤드로 하는 링키드 리스트와 위치 2의 '지'로 시작하는 링키드 리스트에서 각각 보여진다. 이렇게 구성이 된 사전은 별도의 포맷(.dic)으로 출력하여두었다가 다시 읽어 들일 수도 있다. 그렇게 함으로써 다른 분야의 사전도 구성을 하여 필요한 경우 사전만 바꾸면 얼마든지 문제를 만들 수 있는 domain-independent 한 문제로 시스템을 구성하였다.

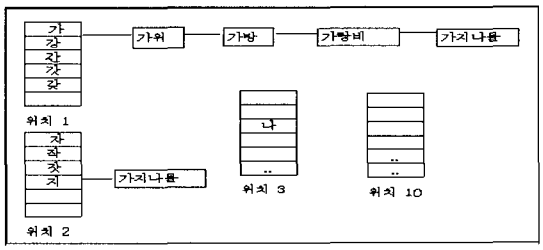


그림 2 최대 허용길이가 10인 경우의 사전의 구성도. 각 위치에 나타나는 글자들을 헤드로 하는 링키드 리스트로 구성된 배열구조이다.

퍼즐의 가로세로크기 S는 시스템이 생성하는 퍼즐 문제의 가로세로 크기에 대한 조건이다. 이 크기는 고정시켜 놓지 않고, 10 × 10부터 50 × 50까지 가능하도록 하였다.

한 글자가 가질 수 있는 자식 노드의 수인 D_a는 다음과 같이 이해를 할 수 있다. 퍼즐 판에서 한 개의 단어가 놓이는 연속의 가로 세로 칸을 각각 가로 프레임, 세로 프레임이라고 이름을 지을 때 프레임이 몇 군데에서 교차를 하는가를 의미한다. 예를 들어서 두 곳에서 교차를 하는 프레임은 한 개의 부모 프레임과 한 개의 자식 프레임을 두고 있는 것으로 생각할 수 있다. 이 경우 D_a은 1이 된다. 세 곳에서 교차를 하는 프레임은 한 개의 부모 프레임과 두 개의 자식 프레임을 갖는 경우이

다(D_a=2). 이것을 트리의 가지 수 개념으로 여겨보면 결국 퍼즐 판이 얼마나 조밀한 글자들을 갖게 되는가의 문제는 가지 수의 제약과도 관계가 있게 된다. 그러므로 퍼즐 판의 조밀도 에 대한 제약을 주기 위해 평균 분지 수라는 제약을 정의하였다.

> 평균 분지 수 = ((out degree가 2인 프레임의 개수 × 2) + (out degree가 1인 프레임의 총 수)) /(out degree가 1이거나 2인 프레임의 총 개수)

부모 노드는 자식 노드를 n개 가지기 위해서 2n-1개의 길이를 지녀야만 한다. 그러므로 자식을 세 명만 갖게 하기 위해서도 부모 프레임에 들어갈 글자의 길이가 5이상이 되어야만 한다. 그러나 글자 길이 5이상인 글자들이 전체 글자의 2%밖에 되지 않으므로 실패할 확률이 높다. 그래서 평균 확장 가능 수를 2로 제한을 한다.

시스템 구성 여섯 가지 중 마지막 남은 평균밀도 D에 대해서는 2.3절에서 설명을 하겠다.

2.2 한글 크로스워드 퍼즐을 위한 알고리즘

가로 세로 S의 크기를 지닌 퍼즐 글자판에서 길이 l의 프레임이 놓일 위치를 결정하는 문제 또한 간단하지 않다. 컴퓨터가 프레임이 가로세로 크기 N인 퍼즐 판에 놓인 위치를 전체적으로 볼 수 있다면 적절한 위치에 놓겠지만, 그렇지 못하기 때문이다. 그림을 참조하여 보면, 위치 (x1, y1)에서 (x2, y2) 의 위치에 글자 W 가 있을 때 다음에 확장이 될 수 있는 위치와 글자를 결정하기 위해 다음과 같은 단계를 거쳐야 한다.

① step 1: Pi = DecidePosin(W) ; // 글자 W에서 확장을 하고자 하는 위치 Pi를 선택한다.

② step 2: Lb = LBound(Pi) ;
Rb = RBound(Pi) ; // 위치 P_i에서 확장이 될 수 있는 좌우(위아래)의 최대 경계 값을 얻어온다.

③ step 3: *len = choose(|LBound - Rbound|)* ;
pos = choosePos(len) ; //경계 값 내에서 적당한 길이(길이>=2) len을 결정한 뒤 자식 노드와의 연결점 pos를 구한다.

④ step 4: W = Search(pos, W_i, len); // pos, W_i, len을 조건으로 하여 사전에서 조건을 만족하는 글자를 찾는다.

위의 매 단계마다 만약 결과를 얻지 못하면 다시 전 단계로 돌아가서 다시 반복을 해야 한다. 예를 들면 단계4에서 길이가 len이고, 위치 pos에 낱 글자 W_i가 있는 단어를 찾지 못한다면 다시 단계3으로 돌아가서 새로운 길이 값을 받아서 다시 단계4를 반복한다. 그리고, 단계2에서 경계 값을 구하는 과정에서는 자식 프레임이 놓인 다음에 확장이 될 수 있는지 없는지를 판단하여야

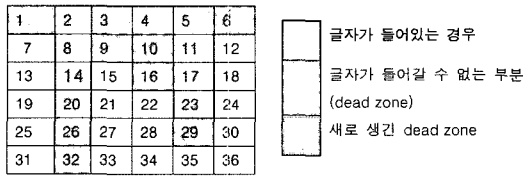


그림 3 가로 프레임을 채우기 전

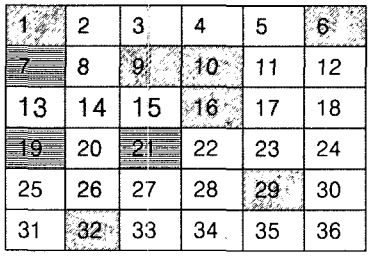


그림 4 14를 교점으로 하여 가로 프레임을 채우고 난 뒤에 dead zone이 새로 생긴 경우(7,19,21)

한다. 이때 dead zone(확장이 될 수 없는 칸)에 해당이 되어서 더 이상 확장 할 수 없는 부분이라면 허용을 하지 않기로 한다. 예를 들어 그림 3과 그림 4에서 14를 교점으로 하는 가로 프레임(13~15)의 경우, 16지점은 확장이 될 수 없는 지점이다. 14를 교점으로 하여 길이 3의 단어를 채운 경우에는 이제 새롭게 dead zone(7, 19, 21)이 생기게 된다.

2.3 평가 기준

퍼즐 생성프로그램의 결과물을 평가함에 있어서 관점을 사용자 관점에서 본다면, 얼마만큼의 난이도를 허용할 것인지가 평가항목이 될 수 있다. 퍼즐의 난이도 또한 사전에서 선택이 되는 어휘의 전문성 여부와 퍼즐을 채우고 있는 white grid의 정도, 교차점의 개수 등이 난이도가 될 수 있다. 그래서 본 논문에서는 퍼즐문제의 난이도 결정 조건을 4가지 제시하고 그 기준들의 값에 변화를 주면서 실험을 해보았다. 우선 퍼즐 판 전체에서의 밀도에 대해 정의를 내려보고자 한다.

평균 밀도(D) = (글자로 채워져 있는 CELL) / (전체 CELL의 개수)

프레임의 개수를 밀도 계산에 고려할 수 도 있지만, 이 경우 교차지점의 중복도 고려하여야 함으로 CELL의 비어있는 정도를 밀도의 계산에 사용하는 것이 더 적절한 방법이다. 대개 50%의 밀도가 나와야 고르게 분포된 것이라고 생각할 수 있다.

사이클이란 그림 5-1과 그림 5-2에서 그 경우를 찾아볼 수 있다. 그림 5-1까지 진행된 경우 다음 단계로 두 가지 방법으로의 확장이 가능하다. '넘'과 '전'이 들어있는 길이 3이상의 단어가 있는 경우 또는 '물'과 '심'이

들어있는 길이 3이상의 단어가 있는 경우에 퍼즐 판을 채울 수 있다. 두 가지 경우에 대해 사전에서 조사를 해 본 결과 '물'과 '심'이 들어가는 단어가 '견물생심'이 존재하므로 그림 5-2와 같이 확장이 될 수 있다.

	천		
돌	연	변	이
	기		심
	념		전
	물		심

그림 5-1 확장을 하면 사이클이 발생할 수 있는 경우

	천		
돌	연	변	이
	기		심
	념		전
견	물	생	심

그림 5-2 사이클의 생긴 경우(천연기념물-돌연변이-이심전심-견물생심)

하지만 이 경우는 사이클이 생긴다. 이와 같은 사이클을 허용하기 위해서는 사전 내의 단어들 중 사이클이 생길 수 있는 조건을 만족하는 단어가 얼마나 되는지가 가장 중요한 변수이다. 사전을 구성하는 단어들 중 어느 정도가 조건을 만족시킬 수 있는지 확인을 하기 위해 다시 사전을 분석하여 보았다.

사이클이 생길 수 있는 조건은 두 개의 낱 글자 X1, X2가 들어있으면서, 두 글자의 간격은 최소한 2이상(d)이어야 한다. 그러므로 글자의 길이는 최소한 3이상이 된다. 인명과 외래어가 모두 포함된 64,000여 개의 글자를 가진 사전에서 x1, x2와 같은 글자 쌍을 가지는 단어의 개수를 조사해 본 결과는 다음과 같다[그림 6]. 가능한 낱 글자의 조합 수 190만개 중 실제로 크로스워드 퍼즐에서 쓰인 낱 글자 쌍은 31,908개이다. 이때 한 개의 조합을 선택을 하였을 때 해당되는 글자가 하나밖에 없는 경우가 18,962개에 해당한다는 결과를 보여주고 있다.

가능한 조합 쌍 190만개 중 3,190 개만의 쌍만이 가능한 글자를 가지고 있다는 것은 전체의 1.6%로서 아주

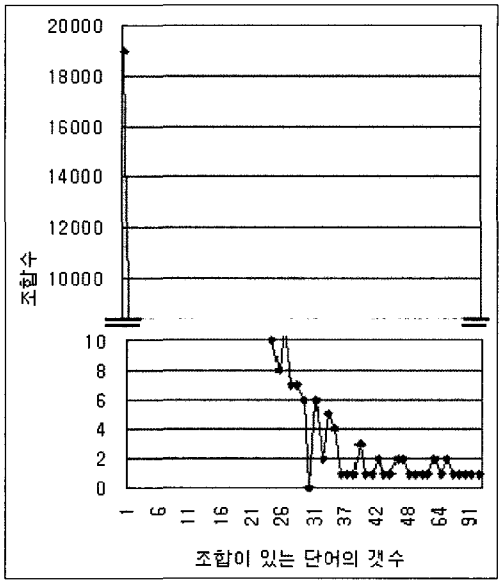


그림 6 단어로 이루어 질 수 있는 가능한 조합에 해당하는 글자 쌍의 수

낮은 확률을 보여준다. 그러므로 한글을 이용한 자동 생성시스템에서는 사이클을 허용하지 않는 것이 바람직하다고 볼 수 있다.

3. 시스템 구현

위에서 설명을 한 점들을 고려하여 개발된 자동 생성 시스템은 퍼즐을 생성하기 위한 사전을 구축하는 부분과 퍼즐을 직접 생성하는 부분으로 나뉜다. 구현된 언어는 Java이며 JDK 2를 사용하여 컴파일을 하였다. 그리고, CPU 1.70GHz, 512MB RAM의 시스템에서 테스트를 하였다.

3.1 사전 부분

사전 부분은 사전의 종류에 따라 text 포맷의 파일을 읽어 들여서 자신의 사전을 구축할 수 있다. 또 이미 만들어진 사전을 따로 저장을 해 두었다가 필요할 때 로드를 할 수 있게 하였다. 그렇게 함으로써 domain independent한 구현이 가능하다. 그리고 퍼즐 생성을 위한 기본 단계로서 길이 len, 특정 위치 p, 낱 글자 w를 적절히 조건으로 주어서 2.2의 단계의 검색조건을 미리 제시할 수 있다.

그리고 사전의 실시간 수정, 삭제, 입력도 가능하게 구현되었다. 그림 7은 두 번째 위치에 단어 '강'이 들어있고 길이가 3인 글자들을 검색한 결과이다. 모두 74개의 글자가 조건을 만족한다는 것을 보여주고 있으며, 그 중 한 글자를 선택하면 뜻을 보여준다. 이 사전의 구조에는 사전에 새로운 단어를 등록하거나 삭제할 수 있는

기능도 있어서 다른 응용에서도 이용이 가능할 수 있도록 하였다.

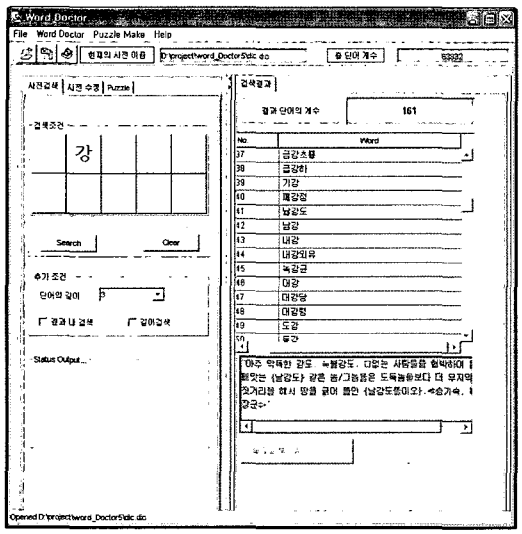


그림 7 두 번째 위치에 '강'이 있는 글자의 검색 결과

3.2 퍼즐 글자판 부분

퍼즐판은 가로세로 크기와 분지 수, 글자의 길이를 조건으로 하고 내부의 geometry는 자동으로 생성을 하여 준다(그림 8). 물론 이미 생성된 퍼즐 문제를 저장하거나 다시 불러올 수 있다. 생성된 퍼즐 판을 만들기 위해서는 만들어진 퍼즐 판의 정보를 저장해두어야 한다(그림 9).

전체 시스템 개요도는 그림 10과 같다.

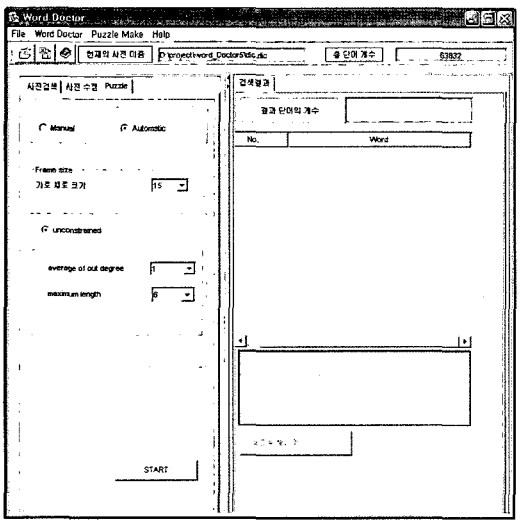


그림 8 퍼즐 판의 여러 조건들을 선택할 수 있는 화면

```
%% Puzzle Board
#BOABD
% 가로와 세로 : 15        15
#-------------------------------------------------------
28
 출애굽기              0      1      3      1
 고혈압증              3      0      6      0
 잡등산이              3      2      6      2
 존슨          0       3      1      3
 .
 .
 .
#================
30
 도출          0       0      0      1
 고기잡이꾼            3      0      3      4
 .
 .
# 전체 프레임개수 : 58
# 퍼즐판의 밀도  : 0.52
```

그림 9 생성 시스템의 결과로 만들어진 퍼즐 판 문제가 저장되는 형태

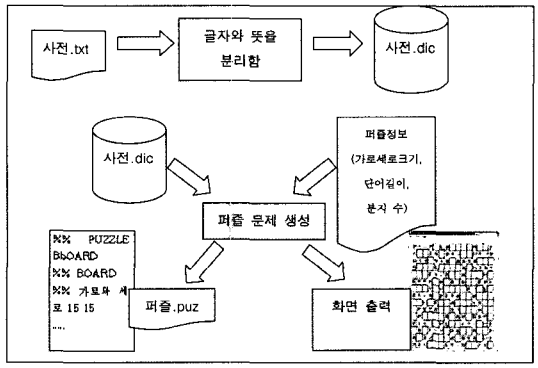


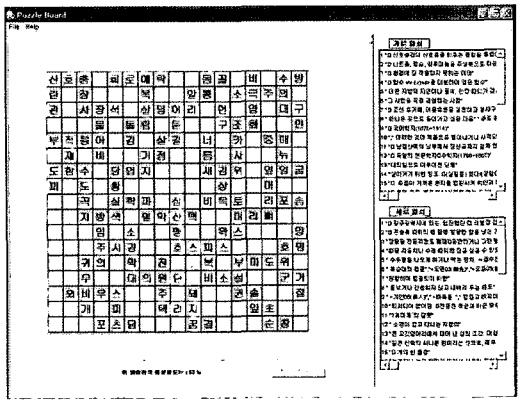
그림 10 시스템 개요도. txt 형태로 된 사전을 읽어서 글자와 뜻을 분리하여 사전을 구축한다. 그 다음 사전 정보와 퍼즐 판의 정보를 읽어서 퍼즐 문제를 생성한 뒤 화면으로 보여주거나 출력을 한다.

3.3 실행 결과

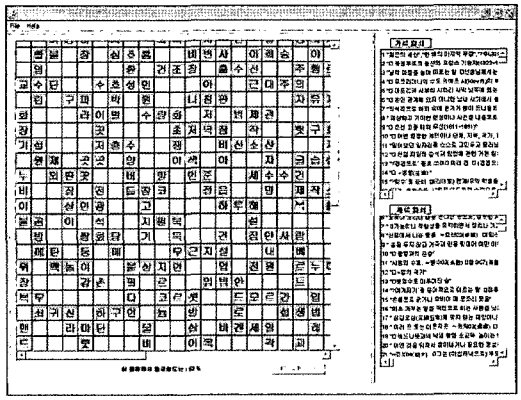
다음은 퍼즐의 난이도를 조정하는 분지 수와 퍼즐의 크기를 다양하게 바꾸어가면서 실행시킨 결과이다(그림 11).

4. 실험

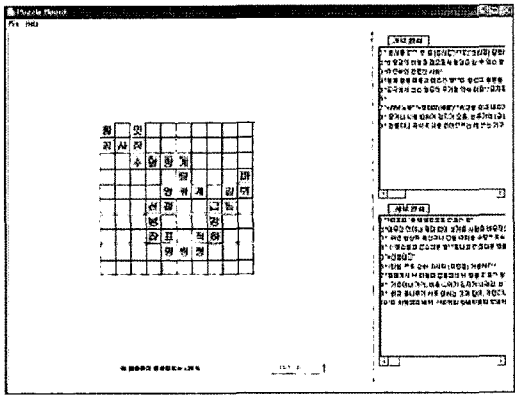
외국에서 만들어진 시스템의 평가는 주로 생성 시간과 노드의 수를 기준으로 평가를 하고 있다[15]. 하지만, 자동 생성시스템에서 문제 생성 시간도 중요하지만 퍼



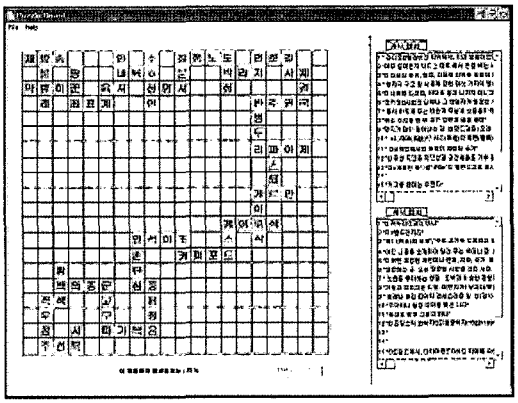
(a) 가로 세로 크기 17, 분지 수(degree) 1.3, 최대 글자 길이 4, 평균밀도 53%



(b) 가로 세로 크기 50, 분지 수(degree) 2.0, 최대 글자 길이 4, 평균밀도 52%



(c) 가로세로 크기 10, 분지 수(degree) 1.0, 최대 글자 길이 4, 평균밀도 29%



(d) 가로세로 크기 20, 분지 수(degree) 1.0, 최대 글자 길이 4, 평균밀도가 24%

그림 11 여러 가지 실행 화면

즐 문제의 난이도를 조절에 따른 시간의 변화 및 조건의 허용 범위들에 대한 평가도 중요하다.

한글의 특성을 반영한 사전을 기본으로 하여 퍼즐 판을 자동으로 생성할 때 입력 조건들을 달리하여 매 조건마다 평균 300회의 실험을 하였다. 조건은 퍼즐 내의 각 프레임들이 가질 수 있는 자식 노드의 수를 나타내는 분지 수와 글자의 최대 길이를 조건으로 하였다.

4.1 분지 수에 대한 밀도의 변화

분지 수를 1부터 2까지로 두었을 때와 가로세로 크기가 달라질 때의 변화를 그림으로 나타내면 다음과 같다(그림 12).

위의 그림에서 볼 수 있듯이 분지 수가 1이나 1.2일 때는 평균 밀도가 낮은 수치를 보이지만, 1.2 이상의 경우는 그다지 큰 차이가 나지 않고 평균적으로 밀도가 50%가까운 결과를 나타내고 있다.

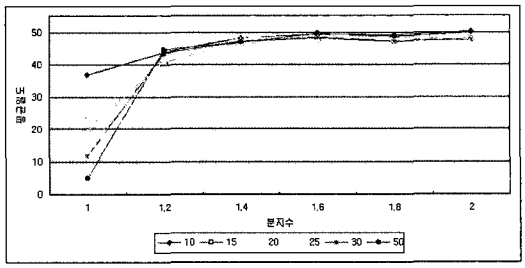


그림 12 분지 수의 변화에 따른 평균 밀도값 변화

4.2 허용하는 글자의 최대 길이에 따른 밀도의 변화

길이 별 밀도의 변화는 분지 수에서의 변화만큼은 민감하지 않은 결과를 보여준다. 퍼즐 크기 가로세로 50이고, 분지 수를 2로 하였을 때 길이에 따른 평균 밀도의 변화는 표 3과 같다.

표 3 길이에 따른 평균 밀도의 변화와 실제 분지 수 (degree) 값의 변화

글자의 길이	실제 분지 수 (degree) 값	평균 밀도
6	1.19	44.7
5	1.24	47.8
4	1.28	50.4
3	1.31	50.8

그리고, 표 4에서는 실제 분지 수를 2로 주었지만, 결과로 나오는 실제 분지 수는 1.2를 넘지 못하는 것을 알 수 있다.

표 4 퍼즐의 크기에 따른 요구 분지 수(degree)간의 관계

퍼즐크기 \ 요구분지 수	2	1.8	1.6	1.4	1.2	1
50	1.24	1.24	1.23	1.21	1.16	1.00
45	1.25	1.24	1.24	1.23	1.18	1.00
40	1.23	1.24	1.25	1.23	1.18	1.00
35	1.23	1.24	1.24	1.25	1.19	1.00
30	1.24	1.23	1.24	1.24	1.19	1.00
25	1.24	1.24	1.23	1.24	1.19	1.00
20	1.23	1.23	1.24	1.24	1.21	1.00
15	1.23	1.23	1.23	1.24	1.20	1.00
10	1.23	1.23	1.23	1.23	1.19	1.00

그리고 분지 수가 고정된 경우 어느 정도의 평균 밀도를 보이는지를 test 해 본 결과 표 5의 결과를 얻었다. 전체 100개의 크로스워드 퍼즐 중 80% 이상이 평균 밀도 45%이상의 평균밀도를 가지는 퍼즐을 한다.

표 5 분지 수1.8일 때의 평균 밀도 별 해당 크로스워드 퍼즐의 개수

평균밀도(%)	~ 25	30	35	40	45	50	55	55 ~
개수	1	4	5	8	17	34	46	34

마지막으로 퍼즐의 생성에 걸리는 시간을 계산해보았다. 가로세로 퍼즐의 크기에 따른 변화와 분지 수에 따른 시간 소요 정도는 다음과 같다. 최대 글자의 길이를 4, 디그리 1로 제약조건을 둔 뒤 퍼즐 판의 크기를 다양하게 바꾸어보았다. 각 100 회의 실험을 한 결과 퍼즐 판의 크기에 따라 그림 13의 결과를 얻을 수 있었다. 전체적으로는 퍼즐 판의 크기가 증가하면 실행시간도 증가하는 그래프를 보여주고 있지만 크기의 변화에 정비례하는 결과는 보이지 않았다. 이는 선택하는 글자와 진행 방향을 랜덤하게 정하는 방식의 알고리즘을 사용했기 때문이다. 다음으로 최대 글자의 길이를 4, 퍼즐의 크기를 20으로 제한을 둔 퍼즐 판의 생성시간을 평가해보았다. 역시 각 100 회의 실험을 하였다. 결과로 나타난 시간 변화는 그림 14와 같이 나타났다. 분지 수 1의 경우를 제외하고는 정비례하지 않고 있다. 이 역시 선택하는 글자와 진행 방향을 랜덤하게 정하는 방식의 알고리즘을 사용했기 때문이다.

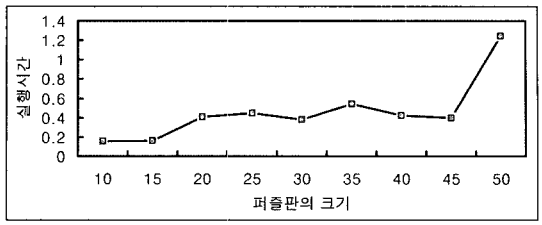


그림 13 퍼즐 판의 크기에 따른 실행시간 변화

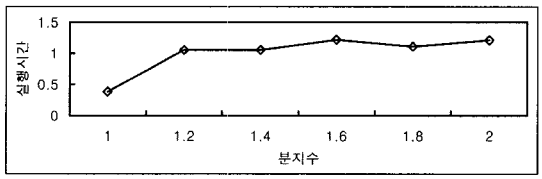


그림 14 분지 수에 따른 실행시간

5. 결론 및 향후 연구

영어나 불어 같은 언어와는 다른 특징을 가지고 있는 한글에 적합한 크로스워드 퍼즐 문제 자동 생성 시스템을 위해서 총 64,000 여 개의 단어들을 가진 한글 사전을 분석하고, 그 결과를 바탕으로 크로스워드 퍼즐 문제 자동 생성 시스템에 적합한 사전을 구축하였다. 그리고, 크로스워드 퍼즐 문제를 생성하는 자동 생성 시스템을 구현하여 보았다. 문제를 생성하는 매 단계마다 단어를 선택하고 진행 방향을 결정함에 있어서 랜덤한 방법을 선택하도록 알고리즘을 적용하여 보았다. 그리고, 퍼즐의 난이도에 영향을 줄 수 있는 여러 가지 조건에 변화를 주어서 test를 해본 결과 다음과 같은 결론을 얻을 수 있었다.

- 분지 수를 1이상으로 한다면 퍼즐의 가로세로 크기에 무관하게 평균밀도 40~50%정도를 가지는 퍼즐 문제를 생성할 수 있다.
- 퍼즐판에 놓여질 수 있는 글자의 최대 길이는 퍼즐 판의 평균밀도와는 상관관계가 없다.
- 분지 수 1.4 이상의 조건만 만족한다면 크기와 무관하게 80%는 40~50%의 평균밀도를 갖는 퍼즐 문제를 생성할 수 있다.
- 퍼즐을 생성하는데 걸리는 시간은 퍼즐 판의 크기가 50보다 작다면 큰 차이가 없다.

추후에 한글에 기반한 크로스워드 퍼즐 문제 생성 시

스템에 대해 다음과 같은 점에서의 보완이 이루어져야 한다.

- 확장지점에서 단어를 선택할 때 낱글자의 빈도수 등을 고려한 알고리즘의 개발이 필요하다. 최악의 경우 분지 수가 1인 조건에서 한번 밖에 쓰이지 않는 낱글자가 처음에 선택이 된다면 더 이상의 확장이 일어나지 않을 수도 있기 때문이다.
- 진행 방향과 확장 지점을 결정할 때 이미 구성되어있는 퍼즐 판의 모양에 대한 정보를 이용하여 퍼즐 판 전체에 고르게 분포를 할 수 있도록 하여야 한다.
- 생성된 퍼즐 문제를 실제로 풀 수 있는 인터페이스도 제공되어야 하며 이는 웹 상에서의 서비스가 가능하도록 개발되어야 한다.
- 특정 분야의 용어로 구성된 사전에 적용하였을 경우의 성능평가가 이루어져야 한다.

참 고 문 헌

[1] http://www.crossword-compiler.com/
[2] http://www.crosswordkit.com/
[3] http://www.crosswordweaver.com
[4] http://www.crosswordtornument.co.kr/
[5] http://www.puzzleland.co.kr/about_puzzleland/news_07.asp.
[6] http://urimal.cs.pusan.ac.kr/urimal_new/
[7] http://www.imada.sdu.dk/~sik/
[8] M.L.Ginsberg et al., "Search Lessons Learned From Crossword Puzzles," TR Dept. of CS, Standford Univ., pp. 210-215, 1990.
[9] Gary Meehan, Peter Gray, "Constructing Crossword Grids: use of Heuristics vs Constraints," SGES Publications, British Computer Society, pp 159-174, 1997.
[10] Geoff Harris 외 2인, "Basic Blocks in Unconstrained Crossword Puzzles," ACM Symposium on Applied Computing, pp. 257-262, 1993.
[11] 이 승선 외 4인, "Compact TRIE Index(CompTI) : 한국어 전자사전을 위한 데이타베이스 색인 구조," 정보과학회 논문지, pp. 3-12, 1995.1
[12] http://home.pusan.ac.kr/~purmi/paper.htm
[13] http://finecrosser.com/en/index.html
[14] G.H.Harris, "Generation of Solution sets for unconstrained Crossword Puzzles," ACM Symposium on Applied Computing, Volume I. pp.214-219, 1990.
[15] G.H.Harris 외 3인, "Dynamic Crossword slot table implementation," ACM/SIGAPP Symposium on Applied Computing, Volume I. pp. 95-98, 1992.
[16] J.J.H.Forster 외 2인, "The Crozzle-A Problem For Automation," ACM/SIGAPP Symposium on Applied Computing,Volume I. 1992.
[17] Jo Spring, "Benchmarking Automated Solution Generators for the Crozzle," ACM Symposium on Applied Computing, pp. 247-250, 1993.

이 승 희

1991년 이화여자대학교전자계산학과 졸업(학사). 2004년 부산대학교 산업대학원 전산학전공(공학석사). 2004년~현재 (주) 나라인포테크. 관심분야는 생물정보학, 한국어 정보처리

권 혁 철

1982년 서울대학교 공과대학 컴퓨터공학과 졸업(학사). 1984년 서울대학교 공과대학 컴퓨터공학과 졸업(공학석사). 1987년 서울대학교 공과대학 컴퓨터공학과 졸업(공학박사). 1988년~현재 부산대학교 컴퓨터공학과 정교수. 관심분야는 한국어 정보처리, 대용량 정보검색, HCI와 인터넷 에이전트, 시맨틱 웹

조 환 규

1984년 서울대학교 계산통계학과 졸업(학사). 1986년 한국과학기술원 전자계산학과 졸업(공학석사). 1990년 한국과학기술원 전자계산학과 졸업(공학박사). 1990년~현재 부산대학교 전지전자정보컴퓨터공학부 정교수. 관심분야는 그래프이론, 생물정보학, 그래픽스