Microcomputer compilation and solution of crosswords

R H Davis and E J Juvshol describe their progress on Maxword, a seminal crossword compilation/solution system based on the BBC micro

A system is described which enables crosswords to be automatically compiled or solved, or which alternatively may assist in their manual compilation or solution. Illustrative examples are quoted and the performance of the system, which has been written for an Acorn BBC model B microcomputer with disc pack and printer, is discussed.

microsystems crosswords BBC micro

Crosswords, in one form or another, have existed for several hundred years. More recently, two particular types have been subject to some degree of computation: cryptic crosswords and conventional crosswords. Cryptic crossword puzzles are found mainly in UK publications, and have clues of the type shown in the following examples[1]

GPO pines to break this alternative communications system! (6, 4) (PIGEONPOST)

Its delivery is the occasion of a lot of talk (6) (SPEECH)

Conventional crossword puzzles use clues which are chiefly based on general knowledge, for example

River in France (5) (LOIRE)

Norwegian composer (5) (GREIG)

This type of crossword is widespread throughout the world.

The feasibility of computer-created crosswords has been investigated by Feger[2] and Mazlack[3-5], amongst others. Feger[2] inserted complete words into a diagram which, together with the first word, was supplied by the user. Mazlack[4,5], on the other hand, described a system in which the diagram was filled letter by letter using probabilities derived from a word list. Both of these constructed crosswords were considered too trivial for publication.

In a more recent work by Mazlack[6], a program is described which fills a user supplied diagram slot by slot with complete words. Using this method diagrams can be filled from published crosswords within a short time. This method was extended[7] to incorporate cryptic clue generation, every word used in filling the diagram pointing to a list of predefined clues for that word.

Another approach is described by Williams and Woodhead[8], who developed a method for computer-assisted analysis of cryptic crosswords. Using a specially defined language, LACROSS, the study analysed the different types of cryptic clues given in UK crosswords. Other studies have been made which attempt to apply computers to crossword puzzles[9-11].

The Maxword system

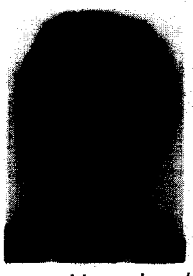
Maxword is a multipurpose package implemented on the Acorn BBC microcomputer. It is intended to automatically compile and solve crosswords. Alternatively it may be used to assist with manual compilation or solution. The system provides a good measure of the level of assistance a common microcomputer is now capable of delivering to this popular pastime.

The manual modules are not very complex from a computing point of view, but do nonetheless gain from use of dictionary and storage facilities as well as from the flexibility provided by the computer.

The automatic compiler includes diagram filling slot by slot and

Department of Computer Science, Heriot-Watt University, Edinburgh, UK

Robert Davis received the degrees of BSc and PhD from the Queens University, Belfast, UK, before working as a systems analyst with Rio Tinto Zinc's subsidiary at Avonmouth, UK. He has since taken up a lecturing appointment at Heriot-Watt University, UK, where he coordinates the MSc course on knowledge-based systems within the Department of Computer Science.



Erik Juvshol joined Heriot-Watt University, UK, in 1982 with computer engineering qualifications from Kongsberg Ingeniorhogskol, Norway. He gained a BSc degree in computer science in July 1984. Since graduation he has taken up a post as systems engineer with Norwegian Telecommunications.

0141-9331/85/10500-07 $03.00 © 1985 Butterworth & Co. (Publishers) Ltd

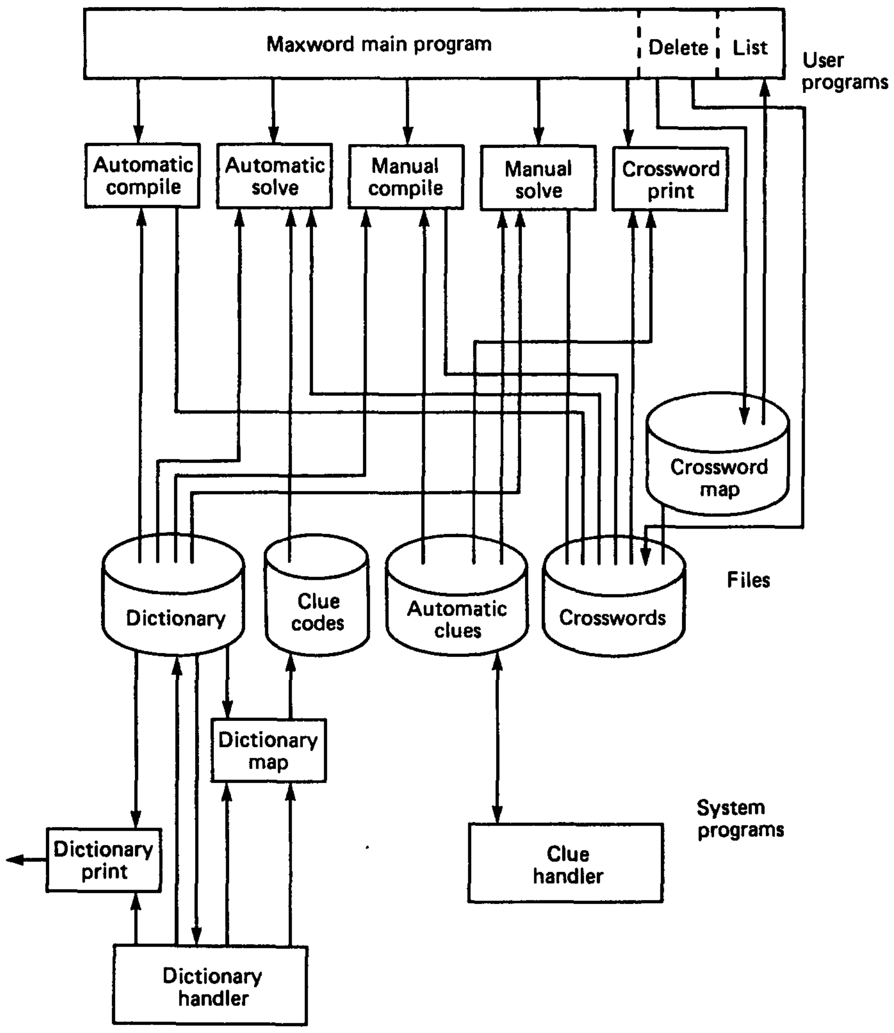


Figure 1. System structure (a link with an arrow shows the direction of both unidirectional and bidirectional flow)

automatic clue generation based on the clue codes of the solution words used in the diagram. Each clue is of a 'general knowledge' type and is defined at the same time as the solution word is chosen.

The Maxword package is completely dependent on a dictionary[12] in which there are one or more clues to correspond to each word. There are two system programs in the package: one handles the dictionary and one the clues. A crossword can also be stored for later retrieval. The system structure is shown in Figure 1.

Menus for users are provided by the main program which controls all the other user programs according to the user's choices.

The dictionary is constructed of words up to a length of 20 characters, but the system is at present limited to a range of 100 different clues. The dictionary handler takes care of everything that has to do with the dictionary; for instance, words together with a clue code can be added to the dictionary, modified, deleted, listed on screen or printed out. The clue handler can add new clues and modify, delete, list or print existing clues.

So far the authors have found no literature describing a computerized crossword solver. Maxword provides an automatic solver which, given both diagram and clues, uses words from the dictionary in an attempt to solve the crossword. The clues given must, however, be of the same type as the recognized internal clues.

Implementation

The modules which make up Maxword are shown in detail in Figure 1. The main program is the 'control block' of the system. All menus concerning user programs are located in the main program so that the menu displayed at any time is dependent on the answers given by the user to previous questions.

The only part of the crossword constructed within the main program is the crossword layout diagram with appropriate 'blocked' (or 'deleted') squares; this can be achieved either automatically or manually.

When any chosen module has finished, the main program is chained in again and the crossword just compiled or solved becomes an entry in the crossword catalogue provided by the main program. Following this the main menu will appear once again.

Main menu

The main menu outlines all the main features provided by the system. The choices provided are

- compile a crossword
- solve a crossword
- delete a crossword
- list all crosswords
- paper copy of a crossword
- exit from program system

Details of further menus and consequent action prompted by each option are to be found in the Maxword user guide, available from the authors.

Compile When a new crossword is about to be constructed, the following data items are needed

- number of squares across
- number of squares down
- automatic or manual blocking of squares
- automatic or manual compilation
- type of clues that will be used
- name of the crossword

Since the user is allowed to continue a previous crossword compilation, it is necessary to specify whether a new or an old crossword is to be used. If an old crossword is to be used only its name is of interest to Maxword, since other information is already known.

Once blocking of squares in the layout diagram has been performed by the main program, an entry in the crossword catalogue is created for a new crossword with no existing clues. Module communication parameters are set according to the choices made and the specified compiler is chained.

Solve To solve a crossword, only three questions need to be answered at first: what is the name of the crossword to be solved; whether solving should be done automatically or manually; and what is the name of the solution. When these questions are answered successfully, the communication parameters are set up and the appropriate module is chained.

Delete Given the name of the crossword to be deleted, and a confirmation of the intention to delete it, a crossword may be removed from Maxword's catalogue and the file on disc deleted. If the crossword cannot be found in the catalogue, a message to this effect is presented.

List The crossword catalogue is listed on the screen in pages holding a maximum of ten crosswords; this is merely a file to screen copying the catalogue.

Print The print module is chained when requested.

Words and clues

Crosswords consist only of words and clues. The words used in Maxword are limited in length to 20 characters (which has proved to be a sufficient length — in fact it has been difficult to find suitable words more than 17 characters long). The number of words in the dictionary is increasing all the time; at the time of writing the dictionary contains roughly 3300 words. The words included in the dictionary are all words which, together with many others, can be referred to with only one clue (eg ANDY, JOHN, PETER, ROBERT are all words with a common clue, 'boy's name').

Each clue has a clue code — a number within the range from 00 to 99 — associated with it. Every word in the dictionary is stored with the clue code as a suffix.

All the words are stored in a file ('words'), ordered in ascending word length and alphabetically within each word length. At the beginning of 'words' there is a map with one entry for each word length containing a pointer to the first word of each word length; this is necessary for the random access and retrieval of individual words.

The clues are stored as strings in a file ('aclues') which again has a map with an entry for each clue code.

Diagrams

For practical reasons a limit of 50 × 50 squares is set for layout diagrams, but the real limitation is the number of slots (254). In the manual mode of diagram design, the user is free to choose any pattern. For automatic diagram design, Maxword provides a simple means for quick diagram generation.

If the diagram is smaller than 35 × 20 squares then it will fit onto the screen. For crosswords that exceed the size of the screen, the screen can be used as a window into the diagram.

Slots

When the diagram has been created the next step is to identify the slots (ie horizontal or vertical sequences of free squares into which solution words may be placed). The diagram is scanned twice to establish the number of slots running across and down the diagram and tables of the required size are created. From a further two scans of the diagram the length and start square of each slot is recorded. Information about the squares occupied by each slot is also recorded, to keep track of which slots are interlocking and at which square interlocking takes place.

Because the information about interlocking slots can be quite substantial (5000 byte if the crossword is 50 × 50) it is kept on a file called 'crmap'. The original slot length, the start square of each slot and the number of blank squares in the slot are stored in tables. Further tables are set up according to the needs of specific modules.

Compilation

The way Maxword's dictionary and clues are structured means that automatic compilation and solution of crosswords are quite similar. The main difference is that during compilation the clue code is not known, whilst during solution it is.

When compiling a crossword, tables are needed to keep the sequence in which the slots are filled, the clue code for each word and the address of that word in the dictionary. Additional tables are kept containing pointers to the start of every word length in the dictionary and the number of words for each word length.

Goal slots

Selecting the next slot to try to fill, the 'next goal slot', is important. The first method used in Maxword was very simple: the next goal slot was always taken to be the longest slot, on the theory that longer words would be more difficult to fit in than shorter words as the layout diagram filled up. This method proved unsuccessful because there were usually a number of slots of length two and three, which eventually caused timing difficulties.

A natural development was to take into account the number of alternative words available for each slot, as described by Mazlack[6]. This method involves a formula for calculating the remaining number of alternatives for slots affected by a particular insertion. The number of alternatives for any one slot is given by

$W_n^{(1-k/n)}$

where n is the word length, k is the number of characters already in the slot and W_n is the number of n-letter words in the dictionary.

This expression assumes a uniform distribution of letters in word positions, but according to Mazlack[6] and the tests done with Maxword it appears to be satisfactory and has improved the compilation success rate considerably.

Word selection

Having decided the next goal slot, Maxword must find a suitable word to fit. A copy of the goal slot is held in a goal string so that, when selecting a word among all the alternatives in the dictionary, one of the following situations applies.

- If the goal string is blank (ie there are no letters in it), a word is chosen randomly amongst the words of length n.
- If the first letter of the slot is already filled with a letter K, for example, the start address of n-letter words beginning with K is calculated based on a uniform distribution of letters in the first position of a word. When there are more than 80 n-letter words in the dictionary, however, an adjustment is made to avoid starting alphabetically beyond the words of interest.
- If there are letters in the slot other than in the first position, words of length n are searched sequentially.

Since the dictionary is too large for a typical microcomputer memory, searching includes a great deal of wasteful disc accessing.

Matching

A word from the dictionary ('dicword') is selected and compared with the goal string, letter by letter, until all letters match. If a dicword matches the goal string, then Maxword checks whether the dicword has already been used or not. If it has been used or if there are backtracking difficulties (described in the following subsection) the search continues; otherwise the word is accepted as matching.

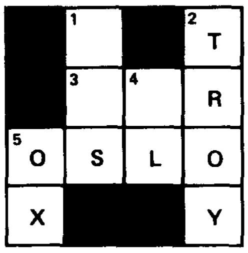


Figure 2. Backtracking illustration

If the dicword is a mismatch, a new word will be fetched from the dictionary. There are three cases in which this fetch sequence will be interrupted: when a match occurs; when the first letter of the dicword is no longer the same letter as the first letter of the goal string; or when there are no more dicwords of the requisite length.

Once matched, a dicword is inserted into the crossword and Maxword goes on to find the next goal slot.

If no matching occurs because there is no word in the dictionary to match the goal string, backtracking must take place.

Backtracking

Backtracking is used to clear a slot already filled, except for any letters also contained in interlocking slots which have been successfully filled. For example, in Figure 2, if the word TROY needed backtracking the letters T, R and Y would be erased but the O would remain, as it is also contained in the slot filled with OSLO.

If there is more than one candidate for backtracking, the compiler first checks the letters in the goal string that failed to match. If any of the letters are found to be particularly difficult to match, ie if a letter appears in less than 3% of the words checked, then the word in which the letter is found is backtracked instead. If no letter is particularly difficult to match the compiler selects the slot which is interlocked with the fewest filled slots. In the case of a tie a candidate is chosen at random.

Solution

In crossword solution, the decision as to which slot should be the next goal slot is similar to that used in the compiler, but is made easier by making use of the fact that the clue code is known. A file called Ccodes contains information about the number of solution words in the dictionary for each clue and word length. Using this information, Maxword sets up a table with the number of alternatives for each slot.

The first slot chosen is the one with fewest alternatives; otherwise a random choice is made. From there the next goal slot chosen continues to be the one with the fewest alternatives which is interlocked with a filled slot. This policy proceeds until the crossword is solved.

Manual operation

Manual compilation and solution are provided as a way of avoiding a limit on the number of words or clues. During both compilation and solution the dictionary is readily accessible.

Manual compilation is implemented as a computerized 'pen and paper' version in which the cursor keys are used to move about the diagram, so that the user can write letters into the diagram, enter clues or ask the dictionary for words. To enter a clue the cursor must be at the start square of the relevant slot. If only one slot starts in that square the slot number is found; otherwise Maxword will ask for the direction of the slot of interest.

To use the dictionary when compiling crosswords manually, the user moves the cursor to the start square of a slot and asks for words. Maxword will then take that slot as a goal string and sequentially search through the words in the dictionary of the requisite length.

Solution is the reverse of compilation with respect to the use of clues; the features provided are similar. Clues are displayed on the screen on request. Entering letters is done in exactly the same way as during compilation, and the dictionary can also be consulted when solving a crossword in the same way as during compilation.

System details

Maxword is implemented on a BBC model B microcomputer with a Torch Z80 disc pack. The Torch pack contains two 400k dual disc drives. An Epson FX-80 printer is connected to produce paper copies. All programs are written in BASIC and the modules are run in mode 7 so that approximately 25 kbyte of memory is available for program and data.

Module communication

Maxword consists of many separate modules and there must be some sort of communication between them. By using a part of the memory, which is not overwritten when a new module is chained, values of constants can be used in different modules. A constant must be assigned to the same memory location in all modules where it is used. Changing the value of a global constant must therefore only be done in the main program.

Library

Maxword stores all crosswords produced solved or compiled, thus acting as a library. The size of the library is presently limited to the space available on disc drive: 32 crosswords. Crosswords may be listed under the following headings

- NEW, just created (before being controlled by a module)
- CPD, compiled (all words and clues inserted)
- PCD, partially compiled
- SVD, solved (all slots filled, correctly or not)
- PSD, partially solved

UDF is an additional code which is used internally when checking file names. The value is returned if a file name or crossword with a given name does not exist. That crossword is then said to be undefined (UDF).

Clues can be represented in four ways within Maxword, as follows

- INT, internal clue codes within the defined range
- EXT, any external clue defined by the user
- OFF, clues held offline and not handled by the computer
- NON, nonexistent definition (ie the clue type has not yet been defined)

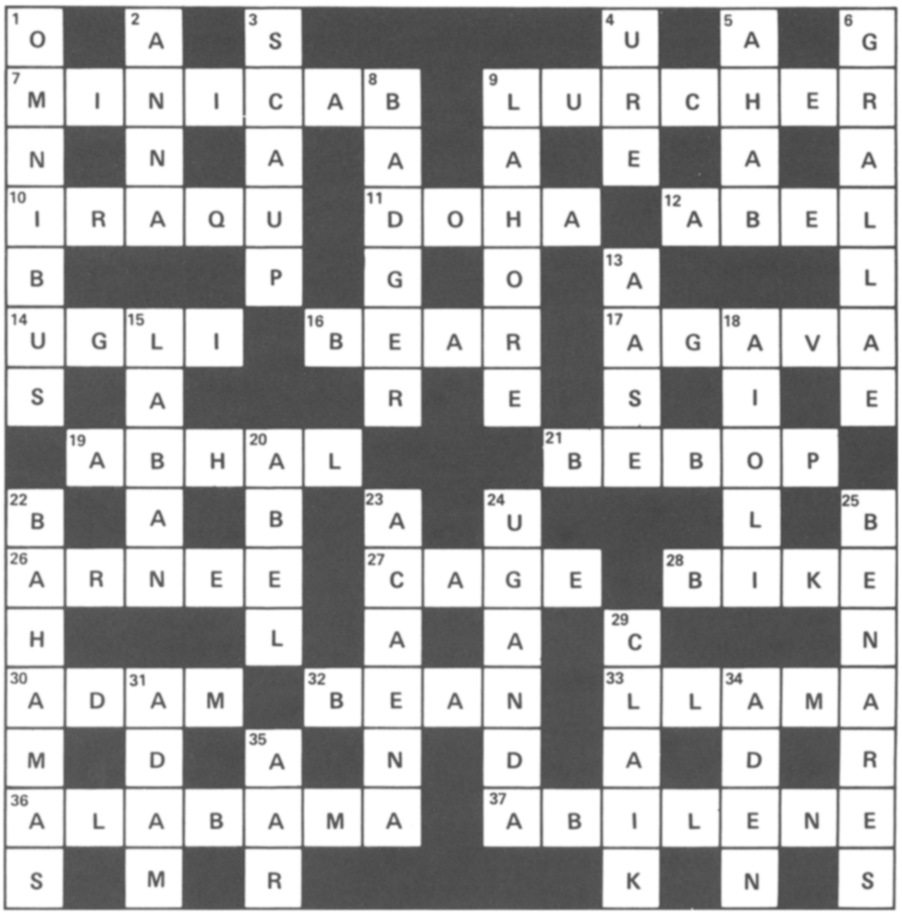
System programs

Maxword provides two system programs, the dictionary handler and the clue handler[8]. The most complex of the two is the dictionary handler, which controls two submodules, dic update and dic range, concerned with updating and merging new words into the dictionary respectively. Controlled by these is a third submodule called clue map, the purpose of which is to prepare information for the automatic solver about the number of words with a given clue code and length. The dictionary handler provides for several options, *viz* add (new words), update (existing words), modify (the clue code of all entries of a given code), list (the dictionary on the screen) and print (the dictionary on an Epson FX-80).

The clue handler copies the file in which all internal clues are held into memory, at which time both additions and modifications can be made.

Results

Since Maxword is implemented on a BBC microcomputer there is only about 25 kbyte available for program and data. This is sufficient for the compiler and its workspace but, because there is considerable disc accessing, the time consumed is an important factor which affects every crossword.



1 O 2 A 3 S 4 U 5 A 6 G
7 M I N I C A 8 B 9 L U R C H E R
N N A A A E A A
10 I R A Q U 11 D O H A 12 A B E L
B P G O 13 A L
14 U G 15 L I 16 B E A R 17 A G 18 A V A
S A R E S I E
19 A B H 20 A L 21 B E B O P
22 B A B 23 A 24 U L 25 B
26 A R N E E 27 C A G E 28 B I K E
H L A A 29 C N
30 A D 31 A M 32 B E A N 33 L L 34 A M A
M D 35 A N D A D R
36 A L A B A M A 37 A B I L E N E
S M R K N S

Figure 3. Automatically compiled crossword

A second important factor is the dictionary richness, ie how many alternatives there are for each slot in the diagram. The impact that the lack of alternatives can have may be seen by comparing the effort needed from the compiler in compiling the crosswords shown in Figures 3 and 4. The slot distributions and compilation times for these crosswords are shown in Tables 1 and 2 respectively.

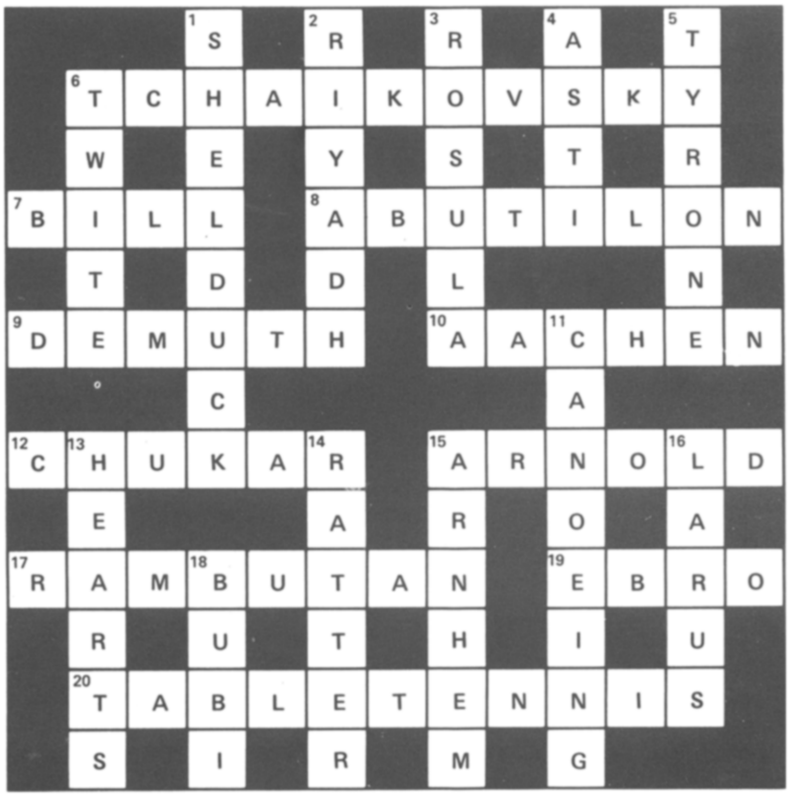
Figure 4 causes the more serious problems of the two, because of the two slots of length 11. Since they are so long they interlock with several other slots, and therefore account to a great extent for the long time needed to fill the diagram.

The filling of Figure 4 is also affected by the problem of unknown start letters. When start letters are not given by interlocking slots, words have to be found by a sequential search. Consequently the slots tend to be matched by words starting with A, B or C. This undesirable feature was avoided by introducing a randomized start point for partly filled slots without any start letter, but only at the cost of an unacceptable increase in the number of backtracks needed to compile a crossword. This problem is yet to be overcome.

The crossword in Figure 4, which took 3334 s to compile, was found to be solved in only 358 s (less than 11% of the compilation time). This is ascribed largely to the difficulties that the two 11-letter words caused during compilation, a fact which turned out to be an advantage for the solver as there were fewer words from which to choose. The solver never came close to solving the crossword shown in Figure 3 within a reasonable time.

Conclusion

Maxword was developed to enable crosswords to be compiled or solved automatically and, if required, to assist in their manual compilation or solution. Whilst both the size of the dictionary involved and the capacity of the space available on a personal microcomputer (such as



1 S 2 R 3 R 4 A 5 T
6 T C H A I K O V S K Y
W E Y S T R
7 B I L L 8 A B U T I L O N
T D D L N
9 D E M U T H 10 A A 11 C H E N
C A
12 C 13 H U K A 14 R 15 A R N O 16 L D
E A R O A
17 R A M 18 B U T A N 19 E B R O
R U T H I U
20 T A B L E T E N N I S
S I R M G

Figure 4. Automatically compiled crossword (source: Daily Record *6 April 1984)*

Table 1. Number of words in the Maxword dictionary for each slot length, and frequency of each slot length occurring in the crosswords shown in Figures 3 and 4

Slot length	Number of possible words	Frequency of occurence	
		Figure 3	Figure 4
2	7	–	–
3	180	2	–
4	553	14	4
5	573	10	2
6	589	4	10
7	614	8	–
8	468	–	4
9	199	–	–
10	106	–	–
11	54	–	2

Table 2. Compilation figures for crosswords shown in Figures 3 and 4

	Crossword shown in Figure 3	Crossword shown in Figure 4
Dimensions	15 × 15	13 × 13
Blocked squares, %	36.0	38.4
Blank noninterlocking squares, %	42.6	40.2
Blank interlocking squares, %	21.3	21.3
Number of backtrackings required	8	134
Time taken to fill diagrams, s	823	3334

the BBC model B) deter from the performance of the system, Maxword nonetheless forms a test bed for further investigation into such areas of interest as the efficient production of crossword layout diagrams, alternative search methods for partially known words and the issues involved in extending the crossword to include a greater range of clue types.

A full program listing and user guide for Maxword is available from the authors on request.

References

1 Source: *The Scotsman* (16 April 1984)

2 **Feger, O** 'A program for the construction of crossword puzzles' *Angew. Informatik* Vol 17 No 5 (1975) pp 189–195

3 **Mazlack, L J** 'The use of applied probability in the computer construction of crossword puzzles' *Proc. IEEE Conf. Decision and Control, San Diego, CA, USA* (December 1973) pp 497–506

4 **Mazlack, L J** 'Data structures required for crossword puzzle construction' *Proc. 36th Annual Meeting of the American Society for Information Science, Los Angeles, CA, USA* (October 1973) pp 141–142

5 **Mazlack, L J** 'Computer construction of crossword puzzles using precedence relationships' *Artif. Intell.* Vol 7 No 1 (1976) pp 1–19

6 **Mazlack, L J** 'Machine selection of elements in crossword puzzles — an application of computational linguistics' *SIAM J. Comput.* Vol 5 No 1 (1976) pp 51–72

7 **Smith, P D** 'XENO: computer-assisted compilation of crossword puzzles' *Comput. J.* Vol 26 No 4 (1983) pp 296–303

8 **Williams, P W and Woodhead, D** 'Computer assisted analysis of cryptic crosswords' *Comput. J.* Vol 22 No 1 (1979) pp 63–66

9 **Smith, P D and Stein, S Y** 'A prototype crossword compiler' *Comput. J.* Vol 24 No 2 (1981) pp 107–111

10 **Cox, J** 'Crossword compiling puzzles the programmer' *Computer Weekly* (30 August 1979)

11 **Juvshol, E J** 'Automatic crossword compilation and solution' *BSc dissertation* Department of Computer Science, Heriot-Watt University, UK (June 1984)

12 **Bailie, J** (Ed.) *The Newnes Pocket Crossword Dictionary* Hamlyn, Feltham, UK, (1983)