A Proposed Standard for Measuring Crossword Compilation Efficiency

H. BERGHEL* AND R. RANKIN
Department of Computer Science, University of Arkansas, Fayetteville, AR 72701, USA

Recent interest in the automation of crossword compilation has lead to the development of a wide variety of different procedures in use on an even wider range of host systems. Due to the variegation between these procedures, comparisons between them in terms of efficiency is difficult. In this paper we provide a dataset and the output listing for a 5 × 5 puzzle which we suggest may be of use as a standardized method of comparison for such compilers in order to measure relative efficiency.

Received August 1989, revised October 1989

1. INTRODUCTION

Major advances in the effort to automate various aspects of crossword compilation have taken place in the past decade. Several effective solution algorithms have been developed in recent years.[1,5,10,12] In addition, significant work has been done in the mechanization of clue construction,[9,11] the human factors aspect,[4,11] and various search heuristics.[4,6,7] In this paper we are interested in performance measurement. Specifically, we seek to develop a procedure by means of which solution algorithms may be meaningfully compared.

One may begin to appreciate the difficulty in making reasonable comparisons by scanning the recent literature. Mazlacks' original compiler ran on an IBM 360/50;[8] Smith and Steen used an ICL 1904S;[10] Smith used a PDP-11/70;[11] Berghel used IBM microcomputers;[1] Harris and Spring used a 80386-based laptop[5] and Wilson's program ran on a Prime 750.[12] There is no less variety in the range of compilers, specifically including ALGOL,[10] various forms of Pascal,[5,11] Fortran,[5,12] and Prolog.[1,4] Run time comparisons amidst such variety provides little information of enduring value. The same program run on a later model of a machine under a more efficient operating system and with an improved compiler may improve by orders of magnitude!

But the matter doesn't end here. Even if the host computing environment were the same, the lexicons may differ, and as they differ so differ the number and variety of solutions found. Further, run times will be as much a function of lexicon size as program efficiency.

Thus, for really meaningful run time comparisons, a protocol should be established which will control the environment. This protocol should specify input and output conditions and reveal a clear and unambiguous philosophy. We suggest such a protocol in the sections to follow.

2. INPUT AND OUTPUT CONSTRAINTS

For any crossword compilation activity which involves the solution of the puzzle, at least two types of input must be present. First, the format of the puzzle must be precisely specified. Second, one must select an appropriate lexicon or word list. These two forms of input both call for different sorts of constraints.

As we discussed in an earlier paper [1], it is useful to characterize the format of a crossword puzzle in terms of geometry, density and degree of interlocking. Each of these considerations affects the level of difficulty in finding solutions (either by person or computer). For benchmarking purposes, we suggest that the appropriate format should be one which maximizes the degree of difficulty. We can ensure that the difficulty is maximized, *ceteris paribus*, by in turn maximizing the number of fail points in the search space. A fail point is the location of possible conflict during insertion. That is to say, were we to solve the puzzle in Figure 1 by first inserting the across words and then attempting to verify the down words, there would be no fail points at all before verification. On the other hand, when word

A	A	R	O	N
B	R	A	V	A
B	A	C	O	N
A	C	O	I	N
S	A	N	D	Y

Figure 1. 5 × 5 Full Crossword Puzzle

* To whom correspondence should be addressed.

insertion alternates between across and down words, at least one additional fail point arises after each insertion. By dealing with the fail points as much as is possible on the 'front end' we ensure that we fail as high up in the search tree as we can. This translates into fewer, and less costly, false starts.

Visual inspection, and brief reflection, will show that the maximum number of fail points for a puzzle defined over an m by n matrix will be m times n, and that this will result only if the puzzle is 100% dense (no black cells) with complete interlocking (every cell interlocks with its horizontal and vertical neighbors to contribute to a word). Since the lexical requirements are simplified when m = n, a square puzzle geometry is called for.

The second important input ingredient is a lexicon. With actual crossword compilation, puzzle aesthetics might demand a large lexicon. However, for purposes of benchmarking, the smaller the better. There are three compelling reasons for this. First, the lexicon must be portable if it is to be useful for test purposes. Anything over a hundred words or so might be difficult to place in the public domain. Second, the benchmark must yield predictable output for verification. Usually, the number of solutions increases with the size of the wordlist, so once again, the smaller the lexicon the better. Thirdly, one wants to maintain a small enough lexicon so that accesses to secondary storage are avoided: one wants to avoid the influence of secondary storage overhead in the run time measurements for they may easily eclipse processing time.

Toward this end, we have selected the 134 words in Table 1. They represent the words which were involved in the creation of the first 24 – 5 by 5 puzzles solved by a computer-created crossword compiler[4] using the electronic form of Webster's Second Unabridged Dictionary (in the order in which they appeared in the output stream). The 24 distinct solutions are depicted in Figure 2.

3. BENCHMARK PHILOSOPHY

If we restrict our attention to input and output conditions, we will still allow enough variation to undermine the value of the resulting comparisons. For example, some crossword compilers use filtration techniques like n-gram analysis and bit mapping to reduce

```
aaron    aaron    aaron    aaron
brava    breva    breva    bravo
bacon    badon    eurus    rebut
acoin    arain    alula    itala
sandy    sandy    monal    metal

aaron    aaron    aaron    aaron
bravo    breva    brace    braze
benin    baton    nathe    retan
aegle    arain    emend    inert
skeet    sandy    rudas    marka

aaron    aaron    aaron    aaron
bravo    broma    brose    clite
bosun    betis    repew    alate
amply    anana    anele    notal
sayal    salal    maral    award

aaron    aaron    aaron    aaron
breba    bravo    blore    clite
buret    demit    rabat    alate
alosa    anele    idiot    rotal
sowel    lader    manny    award

aaron    aaron    aaron    aaron
breme    breva    brava    crore
befit    denim    rebut    addie
anent    anele    itala    reges
salay    laser    metal    abele

aaron    aaron    aaron    aaron
breva    breva    bravo    cline
bacon    denim    rebud    abase
acoin    ankle    itala    tates
sandy    layer    metal    enate
```

Figure 2: 24-5 × 5 Solutions

the number of database searches.[5,10] Such techniques are common to approximate string matching procedures generally (see ref. [2]). Since such techniques could be used to complement any algorithm, their presence could distort the emerging profile if disk accesses were allowed. To ameliorate against this problem, we require that the program and data be primary-resident. While this will not entirely avoid the influence of operating system overhead, it will minimize the effect. For similar reasons, we require that once the correctness of the

Table 1: 134 Word Lexicon

AARON	ALBAN	AROMA	BREVA	LAYER	NODAL	RABAT	RERUN
ABASE	ALLOW	ARULO	BROMA	MANNY	NONET	RACON	RETAN
ABBAS	ALOSA	AWARD	BROSE	MARAL	NONYL	RAMED	RIATA
ABDAL	ALULA	BACON	BURET	MARKA	NOTAL	RANGE	ROBIN
ABEAM	AMPLY	BADON	CLINE	METAL	NOTER	RASPY	RODGE
ABELE	ANANA	BATON	CLITE	MONAL	OBESE	RATED	ROPER
ABNER	ANELE	BEFIT	CRORE	NAMER	OCHNA	RATER	ROTAL
ABRAM	ANENT	BENIN	DEMIT	NANNY	OMINA	REBUD	RUDAS
ABRIM	ANKLE	BETIS	DENIM	NASAL	ONSET	REBUT	SALAL
ACANA	ARACA	BLORE	EMEND	NATAL	ORAON	RECON	SALAY
ACARA	ARAIN	BOSUN	ENATE	NATHE	ORIEL	REDAN	SANDY
ACATE	ARAMU	BRACE	EURUS	NEEDS	OSELA	REFEL	SAYAL
ACOIN	ARARA	BRAVA	IDIOT	NEELD	OTTAR	REGES	SKEET
ADDIE	ARDEB	BRAVO	INERT	NEESE	OVILE	RENES	SOWEL
AEGLE	AREEK	BRAZE	ITALA	NENTA	OVOID	RENKY	TATES
ALADA	ARENA	BREBA	LADER	NETTY	OVULA	REPEW	
ALATE	ARETE	BREME	LASER	NEWEL	OZARK	REROW	

compiler has been verified, the solutions should be counted but not saved.

Inasmuch as it is our intention to emphasize the algorithm and not the compiler, we suggest that a reasonable mix of successful and unsuccessful searches be employed. We endeavor to avoid, to the degree that it is possible, measuring whatever memory management and backtracking characteristics which the executable program might have 'inherited' from the compiler. For this reason, an all-solutions benchmark on a large lexicon would be out of place. On the other hand, a first-solution-found benchmark on a small lexicon with a favorable word ordering would understate worst-case behavior. As a compromise, we offer two tests based upon the same lexicon. For 'average case' behavior, we recommend finding all solutions for the 5 by 5 puzzle using the 134 word list in alphabetical order. For 'worst case' behavior, we recommend attempting to find all solutions on the same word list *sans* 'AARON'. The latter has no solutions and hence will emphasize the performance of the algorithm in detecting unsuccessful paths.

Further, we require that no duplicate words be allowed in any particular solution. Relaxing this condition results in puzzles which are inconsistent with standard practice and will proliferate solutions beyond necessity. Conversely, we do recommend that puzzle transposes be allowed. It is our view that the overhead associated with detecting and eliminating the transposes could easily obfuscate the emerging performance profile. Thus, the actual output of our program would be 48 solutions: the 24 solutions in Figure 2 and their transposes.

Of course, all of the above standardization will be for nought without agreeing to a host system. For this test we recommend the IBM PC which is without a doubt one of the most popular, and most easily accessible computer systems in the world. In addition, all models within the product line have a built in system clock for timestamping and are easy to configure for the tests. Further, all run DOS which is a single tasking operating system which supports a full range of language translators.

4. BENCHMARK RESULTS FOR OUR MODEL

As we stated above, the particular words selected for the lexicon will of course affect performance. We find our wordlist useful for it is easily adaptable to two sorts of measures. The entire wordlist (which will produce 24 solutions not counting transposes) yields, in our view, the more interesting measurement for it emphasizes the algorithm rather than the compiler. While it underemphasizes memory management and backtracking efficiency, these are usually program-independent phenomena, and in our view should probably be discounted anyway. However, for those who wish a more complete measure, shortening the wordlist by one ('AARON') will the maximize the backtracking (now, there will be no solutions) and test the entire spectrum of program and compiler performance.

The crossword compiler that we tested was created by the crossword compiler-compiler described by Berghel and Yi[4] based upon the model of Berghel.[1] We ran our program for both the 133 and 134 word lexicons. The compiler was Arity Prolog 5.1. Since processor type, clock speed, size of main memory and BIOS and DOS versions are all important determinants of benchmark performance, we offer benchmark results for four popular PC configurations for greater portability. These environments are: 1) an IBM/PC (4.77 MHz 8088 w/ 640 k, IBM BIOS), 2) an IBM/PC-AT (6 MHz 80286 w/ 640 k, IBM BIOS), 3) an AT-clone (10 MHz 80286 w/ 512 k, PHOENIX BIOS) and 4) an IBM PS/2-Model 70 (25 MHz 80386 w/2048 k, IBM BIOS). The host operating system was PC-DOS 3.2 for the first three tests, and PC-DOS 4.0 for the Model 70. Operating system overhead was held constant for all four host systems through configuration files (for complete specifications, see ref. [3]). The run times for the four microcomputer systems with respect to both lexicons appear in Table 2.

Table 2: Run Times (seconds)

System	134 word lexicon	133 word lexicon
IBM PC/XT	319.6	250.1
IBM PC/AT	110.1	86.3
10 MHz AT Clone	64.7	50.6
Model 70 PS-2	15.6	12.3

5. CONCLUSION

The philosophy behind our benchmark has the following tenets:

(1) The puzzle format should be as difficult as possible,
(2) Both input and output should be fixed and portable,
(3) Executable programs and data should be primary-resident and should avoid disk accesses,
(4) The tests should emphasize the performance of the algorithm rather than the performance of the compiler,
(5) The benchmark should not proliferate solutions beyond necessity except with respect to puzzle transposes, and
(6) The host system should be widely available.

We emphasize that a slight *caveat* is in order for those who wish to make comparisons based upon this test. In all of the tests we use the wordlist in exactly the same order that it appears in Table 1 (alphabetical). This is important, for word order affects the degree of backtracking. We claim no orthodoxy for this approach beyond the intuitive appeal of a natural word order. We should mention that our run times were nearly 20% faster than reported when we reversed the word order.

For the reasons given, we feel that this is a useful first step toward the standardization of crossword compiler performance measurements. On balance, it seems to be an economical and effective benchmark.

Acknowledgments

We wish to thank D. Roach, C. Yi and the anonymous referees of *The Computer Journal* for invaluable comments on earlier drafts of this paper.

REFERENCES

1. H. Berghel, Crossword Compilation with Horn Clauses. *The Computer Journal*, **30** (2), 183–188 (1987).
2. H. Berghel, A Logical Framework for the Correction of Spelling Errors in Electronic Documents. *Information Processing and Management*, **23** (5), 477–494 (1987).
3. H. Berghel and R. Rankin, A Proposed Benchmark for Crossword Compilation, Technical Report CSAS-TR-89-08, Department of Computer Science, University of Arkansas, August, 1989.
4. H. Berghel and C. Yi, Crossword Compiler Compilation. *The Computer Journal*, **32** (3), 276–280 (1989).
5. G. Harris and J. Spring, Automation of Crossword Puzzle Solutions, Division of CAD, Griffith University, Nathan, Australia, 1989 [manuscript].
6. L. Mazlack, The Use of Applied Probability in the Computer Construction of Crossword Puzzles. *Proceedings of the IEEE Conference on Decision and Control*, pp. 497–506 (1973).
7. L. Mazlack, Computer Construction of Crossword Puzzles Using Precedence Relationships. *Artificial Intelligence*, **7** (1), 1–19 (1976).
8. L. Mazlack, Machine Selection of Elements in Crossword Puzzles: An Application of Computational Linguistics. *SIAM Journal on Computing*, **5** (1), 51–72 (1976).
9. G. Smith and J. duBoulay, The Generation of Cryptic Crossword Clues. *The Computer Journal*, **29** (3), 282–284 (1986).
10. P. Smith and S. Steen, A Prototype Crossword Compiler. *The Computer Journal*, **24** (2), 107–111 (1981).
11. P. Smith, XENO: Computer-Assisted Compilation of Crossword Puzzles. *The Computer Journal*, **26** (4), 296–301 (1983).
12. J. Wilson, Crossword Compilation Using Integer Programming. *The Computer Journal*, **32** (3), 273–275 (1989).

Announcements

2–6 JULY 1990

PARIS, FRANCE

2nd International Conference Economics and Artificial Intelligence

Scope

Economics was based for a long time on the assumption of homo œconomicus. But one has more and more to tackle with the representations that economic agents use when reasoning about the problems they perceive.

The purpose of Artifical Intelligence and Cognitive Sciences is precisely to understand how homo cogitans is building his mental models and his reasoning about his knowledge.

Thus, should not homo œconomicus be conceived as an homo cogitans? Recent economic work is heading in that direction by taking into account the agent's mental expectations, beliefs and effective calculations.

Conversely, should not homo cogitans be conceived as an homo œconomicus? Cannot Artificial Intelligence enhance its methodology by considering the economics of cognition?

The aim of the conference is to explore the cross-fertilization of social sciences (economics, management, organization theory) and cognitive sciences (artifical intelligence, logic, cognitivie psychology, neurosciences).

Topics

1. *Theories and methods*

1.1. *Economic models and knowledge*
- Rational expectations models
- Economy of cognition, value of information, transparence of information
- Risk perception and uncertain knowledge
- Knowledge in economics and non classic logics (epistemic, non monotonic, fuzzy)
- Cooperative and non cooperative games, multiagent models: negotiation, bargaining, contracts
- Knowledge in economic models: asymmetric information, credibility
- Common knowledge and emergence of institutions and conventions

1.2. *Cognition, communication and learning*
- Reasoning with and about limited resources
- Microstructure of markets
- Implicit knowledge, implicit contracts
- Cognitive process of discovery
- Parallel distributed processing and connectionism
- Knowledge, planning and action

1.3. *New modeling methods for complex systems*
- Conceptual modeling in knowledge-based systems
- Metamodeling: the study of the methods of modeling
- Specialized hardware and software for modeling complex systems

2. *Experimentation and simulation*

2.1. *Decision Support System (D.S.S.) for management and organization*
- Intelligent information systems and computer-aided decision making
- Validation procedure for D.S.S. and expert systems
- Connection between large data base and knowledge base
- Strategic and planning game

2.2. *Intelligent forecasting models*
- Planning and control models
- Macro and micro-economic models
- Manufacturing and distribution models
- Expert sytems for financial analysis. Risk assessment and portfolio theory

2.3. *Organizational intelligence*
- Human Resource management
- Legal and fiscal systems
- Office automation
- Economic and social impact of Al
- Innovation and technological change

2.4. *Large systems planning*
- Regional and urban planning
- Environmental resources management
- National health organization
- Transportation and communication

Papers Presentation and Selection

The Program Committee will favor papers which will highlight the cross-fertilization of Artificial Intelligence and related fields with areas of application in Economics, Management and Organization Theory.

Although the area under study is rather new, expected contributions will exhibit openness and rigor, in order to maintain a high level of quality in the presentations.

The proceeding of the first International Conference on Economics and Artifical Intelligence are published by Pergamon Press, IFAC Proceedings series, Oxford, UK, 1987.

Conference Secretariat

Claire VAN HIEU/Corinne SWEENEY. AFCET - CECOIA, 156, boulevard Péreire, 75017 Paris, France. Tél.: (33) 1 47.66.24.19. Fax: (33) 1 42.67.93.12. Télex: 283 155 F ELITA Code 235 E.

29–31 OCTOBER 1990

BOURNEMOUTH, UK

International Conference on Information Technology and People

The ways in which decision makers should be taking account of emerging technology, and of people's reactions to it, will be the subject of an International Conference being organised jointly by Britain's two foremost institutions representing Information Technology professionals – the Institution of Electrical Engineers (IEE) and the British Computer Society (BCS).

The Conference, to be held in Bournemouth's prestigious International Conference Centre from 29–31 October 1990, will take the bold step of bringing together professionals from sociological and economic disciplines as well as the technologists, in order to inject the greatest possible degree of realism into visions of a future information society, answering questions such as:
- is the technology really practicable?
- do people really want it?
- if so, how rapidly is it likely to develop?

Topics

IT and the Customer
IT in Industry and Commerce
IT and the Employer
IT and Communications
IT and Government

Anyone interested in attending the Conference, should contact the IEE Conference Services Department, Savoy Place, London WC2R 0BL. Tel: 01-240 1871. Ext 222.